Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Mineração de Dados de ações com Indicadores Fundamentalistas

Lucas Gomes Almeida
Rafael da Silva Rocha

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientador
Prof. Dr. Jan Mendonça Corrêa

Brasília
2019

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Mineração de Dados de ações com Indicadores Fundamentalistas

Lucas Gomes Almeida
Rafael da Silva Rocha

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof. Dr. Jan Mendonça Corrêa (Orientador)
CIC/UnB

Prof. Dr. Marcos Fagundes Caetano
CIC/UnB

Prof. Dr. Pedro Antônio Dourado de Rezende
CIC/UnB

do Bacharelado em Ciência da Computação

Brasília, 12 de julho de 2019

Dedicatória

Dedico este trabalho primeiramente a Deus, aos meus pais, Mildrede da Silva Rocha e Josafá Xavier Rocha, meu irmão Anderson e minha irmã Giovanna, minha noiva e companheira Bianca e sua família, aos meus avós, tios e tias, primos e primas e a todos meus amigos, por todo apoio e compreensão.

Rafael da Silva Rocha

Dedicatória

Dedico este trabalho à minha família, meus pais, Eveline Maria Gomes Almeida e José Almeida da Silva e minha irmã Mariana Gomes Almeida, por todo o apoio, paciência, carinho, orações ao longo de toda minha formação e por acreditarem em mim, principalmente nos momentos em que nem eu mesmo acreditava.

Lucas Gomes Almeida

Agradecimentos

Agradeço primeiramente a Deus por seu amor e cuidado, pois, sem Ele, nada seria possível. "Porque Deus amou o mundo de tal maneira que deu o seu Filho unigênito, para que todo aquele que nele crê não pereça, mas tenha a vida eterna." João 3:16

Agradeço aos meus pais Mildrede e Josafá, por terem me criado com carinho e amor, por nunca deixarem a desejar como pais, por me darem todo o suporte emocional, financeiro e espiritual, por terem me apoiado na minha trajetória acadêmica até aqui, sempre me incentivando e me ajudando a alcançar meus objetivos e meus irmãos Anderson e Giovanna, por serem irmãos sensacionais e me apoiarem em tudo.

Agradeço a minha noiva Bianca, por ser a melhor companheira que eu poderia escolher, por sempre estar presente nos momentos em que precisei, por me incentivar, me suportar e me ajudar em todas as coisas, por sempre acreditar em mim, por todo amor e carinho que foram cruciais para eu chegar até aqui.

Agradeço à minha família: avós paternos Ana e Pedro e Maternos (In Memoriam) Rosa e Valdemiro, tios e tias, primos e primas em especial ao meu primo Kleberson que sempre me apoiou e me ajudou.

Agradeço à meus sogros Elizabeth e Evanil por todo apoio e à minha cunhada Beatriz que, além de me apoiar e acreditar em mim, me ajudou na execução deste trabalho me dando consultoria sobre termos contábeis.

Agradeço a todos meus amigos que estiveram presentes na minha vida no decorrer dessa jornada.

Agradeço ao professor Jan Mendonça Corrêa pela orientação, por ser um ótimo professor e uma excelente pessoa e por tornar possível a conclusão dessa etapa e aos componentes da nossa banca, professores Marcos Fagundes Caetano e Pedro Antônio Dourado de Rezende, por terem nos dado a honra de fazerem parte da conclusão deste processo.

Agradeço à UnB e o seu corpo docente por fazerem parte dessa trajetória e a Empresa Júnior de Computação a UnB (CJR) por ter tornado a minha passagem na UnB algo muito mais gratificante, enriquecedor e desafiante, e por ter me feito uma pessoa melhor, além de me ter dado a oportunidade de fazer vários amigos e conhecer profissionais excepcionais.

Rafael da Silva Rocha

Agradecimentos

Agradeço primeiramente a Deus, pois sem Ele nada disso seria possível. Ele me deu forças e esteve comigo em todos os momentos.

Agradeço aos meus pais, Eveline Maria Gomes Almeida e José Almeida da Silva, por todo o apoio, suporte, amor e orações, que me moveram pelo curso e, principalmente, pela reta final.

Agradeço à minha irmã, Mariana Gomes Almeida, por todo carinho, amor, noites em claro (me fazendo companhia enquanto eu surtava em meio a trabalhos para entregar e provas para estudar) e pelo apoio de sempre.

Agradeço aos meus avós, Sônia Maria dos Santos Gomes e José Elzimário Alves Gomes, que estão presentes em minha vida desde o meu nascimento, por todo amor e todas as orações que, certamente, me ajudaram a chegar até aqui.

Agradeço aos meus tios, tias, primos, primas e demais familiares por todo apoio e paciência.

Agradeço a todos os amigos que estiveram presentes ao longo deste percurso. Um agradecimento especial às minhas amigas Letícia Gaspar, Ísis Oliveira e Nayara Isabel por serem pessoas tão incríveis e por me transmitirem tanta segurança para que eu seja sempre minha melhor e mais verdadeira versão de mim mesmo, além de todos os incentivos e ajuda que me permitiram suportar a pressão do TCC.

Agradeço ao professor Jan Corrêa pela orientação e por tornar possível a conclusão dessa etapa.

Agradeço aos professores Marcos Fagundes Caetano e Pedro Antônio Dourado de Rezende, por terem aceitado fazer parte de nossa banca e por participarem da conclusão deste ciclo.

Agradeço a esta universidade e seu corpo docente por todo conhecimento que me proporcionaram.

Por último, mas não menos importante, agradeço a todas as pessoas que estiveram presentes no meu processo de formação e que me auxiliaram, direta ou indiretamente, no caminho até aqui.

Lucas Gomes Almeida

Resumo

Este trabalho tem como objetivo encontrar e analisar padrões no conjunto de dados de empresas de setores da BM&F Bovespa utilizando sua cotação e indicadores fundamentalistas.

Além destes, foram adicionados indicadores socioeconômicos. Todos os dados são públicos e foram tratados de forma a serem utilizados nos diversos processos de mineração abordados ao longo do trabalho. Os resultados mostram que é possível identificar padrões não triviais a partir dos dados utilizados.

Palavras-chave: Mineração de dados, Balanço Patrimonial, BM&F Bovespa, Cotação, Indicadores Fundamentalistas, Indicadores Socioeconômicos

Abstract

This work aims to find and analyze patterns in the set of data from BM&F Bovespa's sector companies using their quotation and fundamentalist indicators.

In addition, socioeconomic indicators were added. All data is public and were treated to be used in the various data mining processes addressed throughout the work. The results show that it is possible to identify non-standard from the data used.

Keywords: Data Mining, Balance Sheet, BM&F Bovespa, Quotation, Fundamental Indicators, Socioeconomic Indicators

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Atualmente, grandes quantidades de dados são geradas a cada instante. Tais dados, se considerados de forma individual, não são úteis no processo de obtenção de resultados. Desta forma, torna-se de grande importância obter informações claras a partir destes dados. Tal obtenção se dá através do uso de técnicas que almejam facilitar o entendimento destes dados.

Uma das formas de transformação dos dados em informação é através da mineração de dados. Esta, de acordo com Cios, Pedrycz e Kurgan [2], é uma técnica que tem por objetivo dar sentido a grandes conjuntos de dados em determinado domínio específico, ou seja, a mineração de dados é a extração de conhecimento em grandes quantidades de dados.

Com este trabalho, pretende-se encontrar informações úteis em conjuntos de dados complexos fazendo uso de técnicas de mineração de dados, conforme especificado no Capitulo 3. Tais técnicas têm por objetivo automatizar e agilizar a descoberta de padrões, de forma que o tempo gasto no processamento dos dados possa ser reduzido, bem como a chance de equívocos na análise.

O objeto de pesquisa deste trabalho são conjuntos de dados complexos, públicos e de fácil acesso. A análise fará uso dos dados relativos a empresas com ações na bolsa de valores, suas cotações e indicadores fundamentalistas. Tais dados foram obtidos através de duas ferramentas (sites): Fundamentus e Yahoo! Finanças.

O presente trabalho busca, também, a descoberta de padrões e correlações significativas sobre dados de cotações da bolsa de valores. Além de análises mais generalizadas, foram feitas, também, análises por grupos de ações. Os grupos utilizados foram: Carnes e Derivados; Petróleo, Gás e Biocombustíveis; Tecidos e Vestuário.

1.1 Problema

Os conjuntos de dados relacionados ao mercado financeiro disponibilizados *online* são de grande valia, mas, sozinhos, não têm utilidade para estudos mais complexos. Desta forma, há a necessidade da busca por padrões sobre estes dados, de forma que sejam possíveis análises mais aprofundadas.

1.2 Hipóteses

É possível descobrir padrões úteis em dados sobre o mercado de ações fazendo uso de ferramentas de mineração de dados.

1.3 Objetivo

O objetivo deste trabalho de conclusão de curso é a descoberta de padrões em conjuntos complexos de dados sobre ações do mercado financeiro.

1.4 Objetivos Específicos

Para alcançar o objetivo geral citado acima, foram definidos os seguintes objetivos específicos:

- Extrair os dados financeiros das empresas através de ferramentas *online*.
- Realizar o tratamento desses dados.
- Aplicar técnicas e algoritmos de mineração de dados nos dados tratados.
- Analisar os resultados obtidos após a aplicação das técnicas de mineração.

1.5 Visão geral do trabalho

Este trabalho está dividido nos seguintes capítulos:

- Capítulo 1: Introdução
  - Capítulo onde são listados os objetivos do trabalho, bem como a metodologia utilizada e os resultados esperados.
- Capítulo 2: Referencial Teórico

  - Capítulo que tem por objetivo suprir o leitor de conceitos básicos sobre mineração de dados, mercado de ações e ferramentas utilizadas ao longo do trabalho.

- Capítulo 3: Análise dos dados

  - Capítulo que mostra o desenvolvimento do trabalho propriamente dito. É aqui que são mostrados os resultados obtidos, bem como a análise destes.

- Capítulo 4: Conclusão e trabalhos futuros.

  - Capítulo que trata das conclusões que puderam ser construídas com base no processo de análise, bem como sugestões de trabalhos futuros.

1.6 Metodologia

Obter os conjuntos de dados a partir dos sites Fundamentus[3] e Yahoo! Finanças[4], trata-los de forma a serem compatíveis com as ferramentas a serem utilizadas (*Weka*, *Orange*, *Python* e *Pandas*) garantindo sua qualidade, fazer uso de técnicas de mineração de dados para a descoberta de padrões existentes e, por fim, realizar a análise dos resultados obtidos.

Capítulo 2

Referêncial Teórico

Nesse capítulo são apresentados os principais conceitos e ferramentas utilizados ao longo deste trabalho. Tais conceitos são de suma importância para o entendimento do projeto e estão divididos por seções. Trataremos da explicação e diferenciação no que diz respeito aos conceitos de dado, informação e conhecimento, conceituação e explicação sobre mineração de dados, bem como da descrição das ferramentas utilizadas, mercado de ações e suas especificidades, análise fundamentalista e suas especificidades.

2.1 Dado, Informação e Conhecimento

O desenvolvimento tecnológico trouxe consigo diversas mudanças à forma de funcionamento da sociedade como um todo. Cícero Caiçara [5] nomeia o resultado dessa mudança como sociedade da informação. Para ele, esta pode ser definida como: "um estágio de desenvolvimento social caracterizado pela capacidade de seus membros (cidadãos, empresas e administração pública) de obter e compartilhar qualquer informação, instantaneamente, de qualquer lugar e da maneira mais adequada"[5]. Tal evolução provoca mudança de paradigmas e hábitos e, desta forma, juntamente com a necessidade do armazenamento e posterior tratamento desta informação, surge, também, a necessidade de definição de termos e conceitos que, anteriormente, não eram especificados.

Davenport e Prusak [1] definem a diferença entre dado, informação e conhecimento. De acordo sua conceituação, dados são fatos brutos, simples observações sobre o estado do mundo; estes se caracterizam por sua fácil estruturação, obtenção por máquinas, quantificação e transferência. Informação são os dados após sofrerem uma transformação significativa; em outras palavras, são os dados dotados de relevância e propósito; sua obtenção requer certa unidade de análise, consenso com relação ao significado (para que uma mesma informação não esteja relacionada a dois significados, por exemplo) e mediação humana, tendo em vista que seu processamento requer mais do que a linha de

raciocínio de um computador pode proporcionar. Já o conhecimento é definido como a informação presente na mente humana e que tem valor; para que um dado se transforme em conhecimento, é necessário que este passe por um processo que inclua reflexão, síntese e contexto. Este se caracteriza por ser de difícil estruturação, captura em máquinas e transferência, ao passo que é frequentemente tácito, isto é, que não está declarado, mas que se subentende, implícito, subentendido. O conhecimento pode ser descoberto através do processo chamado Knowledge Discovery in Databases (KDD) que, de acordo com Fayyad [6], é o processo não trivial de identificar padrões nos dados que sejam válidos, potencialmente úteis e minimamente entendíveis. A Tabela 2.1 [1] abaixo resume os três conceitos e explicita suas relações:

Dado	Informação	Conhecimento
• Facilmente estruturado; • Facilmente obtido por máquinas; • Facilmente obtido por máquinas; • Facilmente transferível;	• Requer unidade de análise; • Exige consenso em relação ao significado; • Exige necessariamente a mediação humana;	• De difícil estruturação; • De difícil captura em máquinas; • Frequentemente tácito; • De difícil transferência;

Tabela 2.1: A relação entre dado, informação e conhecimento [1]

2.1.1 Dado

Para Rezende [7], "o dado é entendido como um elemento da informação, um conjunto de letras, números ou dígitos, que, tomado isoladamente, não transmite nenhum conhecimento, ou seja, não contém um significado claro". Para O'Brien [8], "dados são fatos ou observações cruas, normalmente sobre fenômenos físicos ou transações de negócios". Uma outra definição, segundo Valdemar W. Setzer [9], diz que dados são definidos como uma sequência de símbolos quantificados ou quantificáveis (onde quantificável significa algo que pode ter sua quantidade determinada e depois reproduzido sem que se perceba a diferença entre este e o original). Em síntese, é um elemento que, quando tomado de forma isolada, não gera qualquer tipo de compreensão acerca da realidade.

Davenport e Prusak [1] definem dado como "observações sobre o estado do mundo". Um exemplo básico seria dizer: "existem 123 unidades dentro desta caixa". A observação destes fatos brutos podem ser realizadas por pessoas ou por algum tipo de tecnologia apropriada, não sendo necessário nenhum tipo de interpretação. Tal constatação pode, ainda, ser facilmente capturada e armazenada.

Algumas das características que mais se destacam ao se falar de dados são sua facilidade de estruturação, de obtenção por máquinas, de transferência e armazenamento e de quantificação.

2.1.2 Informação

Drucker [10] define informação como *"dados dotados de relevância e propósito"*. Tal relevância e propósito são conferidos aos dados pelo ser humano e, em oposição aos dados, a informação exige análise.

De acordo com Setzer [11], informação é uma abstração informal, ou seja, não há como descrevê-la de modo formal através de uma teoria lógica ou matemática. Ou seja, a informação está presente na mente das pessoas e é representada de forma diferente para cada indivíduo, a depender do que é significativo para cada um. Para Setzer [11], não se trata de uma definição, mas sim de uma caracterização, ao passo que "significativo" e "indivíduos" são termos que não têm uma definição muito clara.

Citando Setzer [11]:

> "Um entendimento intuitivo desses termos. Por exemplo, a frase 'Paris é uma cidade fascinante' é um exemplo de informação – desde que seja lida ou ouvida por alguém, desde que 'Paris' signifique para essa pessoa a capital da França (supondo-se que o autor da frase queria referir-se a essa cidade) e 'fascinante' tenha a qualidade usual e intuitiva associada com essa palavra."

Desta forma, para que uma informação seja armazenada em um computador, esta tem que ser convertida em um conjunto de dados (estruturas mais simples e livres de significado), tendo em vista que o fator humano presente no processo citado acima impede o processamento pela máquina.

Um ponto fundamental que diferencia dado de informação é o fato de que o primeiro é puramente sintático, enquanto o segundo possui semântica. Searle [12] explana este conceito da seguinte maneira: considere uma tabela que possui nomes de cidades em uma coluna, meses enumerados de 1 a 12 em outra e a temperatura média anual para cada cidade no mês correspondente em uma terceira coluna. No entanto, a tabela está escrita em Chinês. Para uma pessoa brasileira que não possui conhecimento sobre a língua chinesa, a tabela é constituída apenas por dados. Ainda que houvesse ordenação dos destes obedecendo algum tipo de ordem pré-determinada, tal processo seria apenas sintático, não contribuindo para o entendimento por parte do indivíduo.

Todavia, se a mesma tabela fosse traduzida para o Português, os elementos da tabela deixariam de expressar apenas a sintaxe e passariam a significar algo para essa mesma pessoa descrita acima.

2.1.3 Conhecimento

Davenport e Prusak [1] chamam atenção para o fato de que a importância do envolvimento humano aumenta conforme evoluímos pelo processo dados-informação-conhecimento. Para eles, o conhecimento é a parte mais valiosa desta "tríade" e, consequentemente, a mais difícil de gerenciar. Seu valor se baseia no fato de que alguém deu à informação um significado, um contexto, uma interpretação. De forma mais geral, pode-se considerar o conhecimento uma grande síntese de múltiplas fontes de informação.

Setzer [11] caracteriza conhecimento como sendo algo pessoal, uma abstração interior de algo que foi experimentado, vivenciado, por alguém.

Davenport e Prusak [1] evidenciam, ainda, algumas das principais características do conhecimento. Este é de difícil estruturação, tendo em vista sua componente livre (participação humana), de difícil captura em máquinas, é frequentemente implícito, subentendido e de difícil transferência.

2.2 Mineração de Dados

Com o advento da tecnologia e consequente avanço do compartilhamento e armazenamento de conhecimento, surgiu a necessidade do tratamento de volumes cada vez maiores de dados. Cios, Pedrycz e Kurgan [2] descrevem que, neste contexto, surge a mineração de dados, um ramo da Ciência da Computação que tem atraído muita atenção da indústria da informação e da sociedade como um todo. Isto se dá em parte por conta da disponibilidade de quantidades extremamente grandes de dados e a crescente necessidade de transformar estes em informação e conhecimento.

Para Cios, Pedrycz e Kurgan [2], as aplicações desta vertente da Computação no âmbito prático são as mais diversas, tais como análise de mercado, controle de produção, detecção de fraude, exploração científica, entre outros. Han e Kamber [13] acrescentam, ainda, que a mineração de dados pode ser vista, então, como a evolução da tecnologia da informação.

O avanço de aplicações que possuem como base o processamento de estruturas de dados heterogêneos, isto é, estruturas compostas por elementos de tipos diferentes (sistemas de bancos de dados, por exemplo), foi, também, um fator decisivo no que diz respeito à ascensão do *data mining* (Mineração de Dados). O aumento da quantidade de informação armazenada demandou formas melhores, mais rápidas e mais baratas de manipulação, tendo em vista que, sem mecanismos eficientes de extração de conhecimento, de nada adianta o armazenamento massivo de dados (Han e Kamber)[13].

Do ponto de vista dos detentores de dados, estes têm como principal objetivo não somente entender o que já possuem, mas adquirir conhecimento a partir disso, de forma

que seja possível solucionar problemas reais das melhores formas possíveis. Este conhecimento deve ser de fácil compreensão, válido, novo e útil (Cios, Pedrycz e Kurgan)[2]. O processo para a obtenção deste conhecimento consiste em uma sequência iterativa de passos, os quais vão desde o pré-processamento dos dados até a mineração propriamente dita, sendo esta seguida da apresentação da informação obtida ao usuário.

A série de passos descrita pelo manual CRISP-DM produzido pela IBM [14] é a seguinte:

- *Business Understanding* (Compreensão do negócio);
- *Data Understanding* (Compreensão dos dados);
- *Data Preparation* (Preparação dos dados);
- *Modeling* (Modelagem) – Onde ocorre a mineração dos dados;
- *Evaluation* (Avaliação);
- *Deployment* (Implantação);

É possível notar que a parte descrita como mineração de dados está contida em apenas um dos passos apresentados acima; ainda assim, é o passo essencial à análise como um todo, tendo em vista que é a mineração que tem o potencial de descobrir padrões antes escondidos no conjunto de dados para que, então, estes possam ser avaliados. Os passos citados acima serão aprofundados a seguir.

2.2.1 *Business Understanding* (Compreensão do negócio)

O passo da compreensão do negócio se caracteriza por explorar o que a organização ou cliente espera como resultado da mineração de dados. Por mais que esta pesquisa pareça dispensável, esta é uma parte importante do processo, tendo em vista que saber as razões pelas quais a mineração está sendo realizada ajuda a equipe envolvida a estar em sintonia ao passo que impede que recursos sejam desperdiçados. [14]

2.2.2 *Data Understanding* (Compreensão dos dados)

A fase de compreensão dos dados envolve um olhar mais aprofundado nos dados disponíveis para análise. Este passo é crucial no que diz respeito a evitar problemas na fase seguinte (preparação dos dados). A compreensão dos dados envolve acessar estes dados e explorá-los (com o auxílio de tabelas e gráficos, por exemplo). Esta etapa permite a determinação da qualidade dos dados a serem utilizados. [14]

2.2.3 *Data Preparation* (Preparação dos dados)

A preparação dos dados é um dos aspectos mais importantes e mais demorados da mineração de dados. Dependendo da organização ou cliente e seus objetivos, as seguintes subfases podem estar presentes neste ponto do processo: fusão de conjuntos de dados, seleção de amostras de subconjuntos de dados, junção de registros, derivação de novos atributos, classificação dos dados para modelagem, remoção ou substituição de valores em branco ou ausentes e divisão do conjunto de dados em subconjuntos de treinamento e teste. [14]

2.2.4 *Modeling* (Modelagem)

A etapa de modelagem é onde a mineração dos dados propriamente dita ocorre. Os dados que foram preparados nas etapas anteriores podem, agora, ser combinados de forma a revelar padrões e soluções para os problemas apontados na fase de compreensão do negócio. A modelagem é, geralmente, conduzida em iterações múltiplas. Os mineradores de dados rodam uma série de modelos utilizando parâmetros variados (podendo ser refinados ou mesmo substituídos no decorrer do processo). É raro uma questão de mineração ser resolvida de forma satisfatória com um único modelo e uma única execução. [14]

2.2.5 *Evaluation* (Avaliação)

Nesta fase, grande parte do processo de mineração de dados já foi executado. Os modelos determinados na fase anterior são avaliados de forma a verificar se são tecnicamente corretos e eficazes de acordo com os critérios definidos anteriormente. É necessário, ainda, avaliar os resultados de acordo com o que foi discutido no início do projeto. Este passo é necessário para assegurar que a organização ou cliente possa fazer uso dos resultados obtidos. Dois tipos de resultado são produzidos pela mineração dos dados: os modelos finais selecionados na fase anterior (modelagem) e quaisquer conclusões ou inferências desenvolvidas a partir dos modelos, do processo de mineração em si e dos dados. [14]

2.2.6 *Deployment* (Implantação)

A etapa de desenvolvimento é o momento do processo em que o conhecimento adquirido nas fases anteriores é utilizado para realizar melhorias na organização ou cliente. As melhorias podem ser tanto físicas, regimentais ou mesmo comportamentais. Em geral, a fase de desenvolvimento envolve dois tipos de atividades: planejamento e monitoramento do desenvolvimento de resultados e compleição de ajustes finais (tais como produção de

relatórios finais e condução de projetos de revisão). A execução destas atividades se dará de acordo com as necessidades da organização ou cliente. [14]

2.3 Técnicas e algoritmos de Mineração de Dados

Como já citado anteriormente [2][13], a mineração de dados consiste em um conjunto de técnicas multidisciplinares que engloba uma grande quantidade de tecnologias no âmbito da tecnologia da informação. Esta seção apresenta as principais técnicas que serão utilizadas no decorrer deste projeto. Para cada técnicas existem vários algoritmos desenvolvidos.

2.3.1 Classificação

A classificação [15] é uma técnica que tem suas bases no aprendizado supervisionado, ou seja, o método aproxima funções-alvo de valor discreto, onde a função aprendida é, então, representada por uma árvore de decisão [16]. Possui algoritmos de *data mining* que determinam a saída de uma nova instância de dados através da criação de um guia passo a passo.

A árvore de decisão [16] é uma estrutura que consiste de nós internos e externos organizados de forma hierárquica e conectados por ramos. O nó interno (também chamado nó decisório ou nó intermediário) é a unidade de tomada de decisão que realiza a avaliação (através de teste lógico) de qual será o próximo nó descendente ou filho. Em oposição, um nó externo (também conhecido como folha ou nó terminal) está associado a um rótulo ou a um valor.

Árvores de decisão usadas para problemas de classificação são chamadas de Árvores de Classificação. Neste caso, as folhas são as classes e os nós são as decisões [16].

A classificação utiliza-se do conceito de 'conjunto de treinamento' para produzir um modelo. Utiliza-se um conjunto de dados com valores de saída conhecidos e, a partir deste, é produzido um modelo. Assim, quando houver um novo ponto de dados (com um valor de saída desconhecido) utiliza-se o modelo e, a partir deste, tenta-se prever a classe [17].

2.3.2 Clusterização

A clusterização (ou agrupamento) é uma técnica que se baseia no aprendizado não supervisionado, ou seja, diferentemente da classificação, esta técnica não necessita que os registros sejam previamente categorizados [15].

A clusterização visa identificar e aproximar os registros similares. Um agrupamento (ou *cluster*) é uma coleção de registros similares entre si, porém diferentes dos outros registros nos demais agrupamentos. Não há a pretensão de classificar, estimar ou predizer o valor de uma variável, ela apenas identifica os grupos de dados similares [15].

Em geral, as medidas utilizadas para indicar a similaridade entre os elementos são as medidas de distâncias tradicionais (Euclidiana, Manhattan, etc) [15].

Pode-se criar um número específico de grupos, a depender das necessidades. As aplicações das tarefas de agrupamento são as mais diversas: pesquisa de mercado, reconhecimento de padrões, processamento de imagens, análise de dados, segmentação de mercado, taxonomia de plantas e animais, pesquisas geográficas, classificação de documentos da Web, detecção de comportamentos atípicos (fraudes), entre outras [15].

Clusterização Hierárquica

A ideia básica do método hierárquico é a criação de agrupamentos através da aglomeração ou da divisão dos elementos que compõem o conjunto. Este método gera um dendrograma (gráfico em formato de árvore que será melhor descrito a seguir) [15].

O método hierárquico pode ser dividido em dois tipos básicos: aglomerativo e divisivo.

- Aglomerativo: Faz uso de estratégia bottom-up onde, inicialmente, cada um dos objetos é considerado um agrupamento. A similaridade é, então, calculada entre um agrupamento específico e os demais agrupamentos. Os agrupamentos mais similares são unidos e formam novos agrupamentos. O processo continua, até que exista apenas um agrupamento principal [15][18].
- Divisivo: Faz uso de estratégia top-down, onde, inicialmente, todos os objetos são parte de um mesmo agrupamento. Os agrupamentos vão sendo divididos até que cada objeto represente um agrupamento [15][18].

Dendrogramas

O dendrograma [19] é um diagrama de árvore que mostra grupos que foram formados por agrupamentos de observações em cada um dos passos e por níveis de similaridade. Tais níveis de similaridade são medidos ao longo do eixo vertical e as diferentes observações são listadas ao longo do eixo horizontal.

Distance Maps

Os *distance maps* [20] (ou mapas de distância) são gráficos que se caracterizam por promover a visualização da distância euclidiana (distância entre dois pontos) calculada

com base na distância entre os atributos numéricos que estão sob análise. Tal visualização é a mesma que obteríamos se fosse impressa uma tabela numérica. A diferença é que, no caso dos *distance maps*, os números são representados por cadências de cores, por exemplo, o branco significa distância zero e o preto significa distância máxima. As cores que aparecem entre o preto e o branco são as distâncias intermediárias.

2.3.3 Algoritmo *k-Nearest-Neighbor*

Para Camilo [15] boa parte das técnicas de classificação fazem uso de um conjunto de dados de treinamento para, assim, aprender a classificar um novo registro. Desta forma, quando são submetidas a um novo registro elas já se encontram prontas, isto é, já aprenderam.

Camilo [15] diz ainda que o algoritmo *k-Nearest-Neighbor* encontra-se em uma categoria diferente de métodos. Tal categoria somente realiza esse aprendizado quando solicitado para a classificação de um novo registro. Assim, o aprendizado é considerado tardio (*lazy*). Tais métodos necessitam de um tempo relativamente menor de treinamento, mas, em contrapartida, são muito dispendiosos computacionalmente falando, ao passo que necessitam de técnicas e recursos para armazenar e recuperar os dados utilizados no treinamento.

O algoritmo (*k-Nearest-Neighbor*) foi inicialmente descrito na década de 50, mas apenas se tornou-se popular na década de 60 quando do aumento da capacidade computacional. De forma suscinta, o algoritmo realiza o armazenamento dos dados utilizados para o treinamento e, no momento em que um novo objeto é submetido para classificação, o algoritmo faz uma busca pelos k registros mais próximos do novo registro (medida de distância). Este recebe sua classificação de acordo com a classe mais comum entre os k registros mais próximos [15].

Uma segunda definição, dada por Abernethy [21], diz que o algoritmo *k-Nearest-Neighbor* (também conhecido como Filtragem Colaborativa ou Aprendizado Baseado em Instância) é um técnica de mineração de dados que permite usar instâncias de dados anteriores, com valores de saída já conhecidos, para realizar a previsão de um valor de saída desconhecido em uma nova instância de dados.

No Weka (recurso citado posteriormente), o algoritmo encontra-se sob o nome *lazy.IBk*, onde o IB significa *Instance Based* (Baseado em Instâncias) e o k é o número de vizinhos que o algoritmo examinará.

O algoritmo foi utilizado no trabalho por tratar de valores numéricos, fato este de extrema importância, tendo em vista que trataremos de dados relacionados ao mercado financeiro.

2.3.4 Correlação

Segundo o dicionário Aurélio, correlação significa relação mútua entre dois termos, qualidade de correlativo, correspondência. Correlacionar, significa estabelecer relação ou correlação entre partes; ter correlação.

Gennari [22] afirma que "os recursos são relevantes quando seus valores variam sistematicamente com a categoria". Em outras palavras, de acordo com Hall [23], "um recurso é útil se estiver correlacionado ao preditivo da classe; caso contrário, é irrelevante".

Deste modo, temos que a correlação auxilia no processo de eliminação de dados irrelevantes bem como de dados redundantes [23].

É importante ressaltar que correlação não implica em causa [24][25], ou seja, não é possível afirmar que determinado fato é consequência de outro tendo observado apenas a correlação entre ambos. De toda forma, para o trabalho em questão, tais dados (correlações) podem ser usados para auxiliar na tentativa de previsão do comportamento no que diz respeito a valores de ações de empresas da bolsa de valores.

2.4 *Python*

Segundo Borges [26]:

> "*Python* é uma linguagem de altíssimo nível (em inglês, *Very High Level Language*) orientada a objeto, de tipagem dinâmica e forte, interpretada e interativa.
>
> Possui uma sintaxe clara e concisa, que favorece a legibilidade do código fonte, tornando a linguagem mais produtiva.
>
> A linguagem inclui diversas estruturas de alto nível (listas, dicionários, data / hora, complexos e outras) e uma vasta coleção de módulos prontos para uso. Multiparadigma, a linguagem suporta programação modular e funcional, além da orientação a objetos. Mesmo os tipos básicos no *Python* são objetos. A linguagem é interpretada através de *bytecode* pela máquina virtual *Python*, tornando o código portável. Com isso é possível compilar aplicações em uma plataforma e rodar em outros sistemas ou executar direto do código fonte."

Foi escolhida a linguagem *Python* para o tratamento e manipulação dos dados por, além do descrito anteriomente, a mesma ser *open source*, gratuita e que contém diversas bibliotecas que auxiliam nos processos de mineração de dados.

2.4.1 Pandas

É uma biblioteca open source, licenciada pelo BSD[1] a qual fornece uma estruturas de dados de alto desempenho, fáceis de usar além de prover um gama de ferramentas de análise de dados que podem ser utilizadas em scripts escritos em Python.[28]

[1] Licença de código aberto.[27]

2.5 *Weka*

Weka é um conjunto de algoritmos relacionados à *Machine Learning* que têm por principal objetivo a realização de atividades de mineração de dados [29]. De acordo com Witten et al [30], a sigla é uma abreviação da expressão *Waikato Environment for Knowledge Analisys*. Como o nome sugere, foi desenvolvido pela Universidade de Waikato, localizada na Nova Zelândia. É composto por implementações de algoritmos de aprendizagem de máquina em conjunto com uma série de recursos que realizam pré-processamento, regressão, classificação, clusterização, aplicação de regras de visualização de dados e apresentação de resultados [29].

O *Weka* foi desenvolvido através da linguagem de programação Java e sua distribuição segue os termos da GNU (*GNU General Public License version 3.0 (GPLv3)*, ou Licença Pública Geral). Sua GUI (*Graphical User Interface*) é voltada para a interação com arquivos de dados e produção de resultados visuais. Por possuir uma API geral, é possível incorporá-lo aos aplicativos que realizam tarefas de mineração de dados como uma biblioteca.

A ferramenta apresenta, também, uma variedade de recursos, como, por exemplo, o suporte às etapas do processo experimental de mineração de dados, desde a preparação dos dados de entrada, análise estatística de esquemas de aprendizagem, até a visualização dos dados e apresentação dos resultados. Possibilita, ainda, a integração direta com bancos de dados. Tal funcionalidade permite ao usuário obter os dados diretamente da base e salvá-los em um formato adequado para uso posterior no *Weka*.

O *Weka* faz uso do formato ARFF (*Attribute-Relation File Format*), um arquivo de texto ASCII que descreve uma lista de instâncias que compartilham um conjunto de atributos [31]. Arquivos .arff possuem duas subdivisões:

- *Header*: Parte do arquivo que contém o nome da relação, uma lista dos atributos (as colunas nos dados) e seus tipos.
- *Data*: Apresenta os dados de cada atributo para sua determinada classe.

Um dos motivos pelos quais a ferramenta apresenta destaque entre as demais disponíveis no mercado é o fato de possuir distribuição livre e multiplataforma (por ser criado em linguagem Java, como mencionado anteriormente) o que a torna adaptável a diferentes sistemas operacionais (como Windows, GNU/Linux e Mac OS).

Segundo Witten et al. [30], o *Weka* apresenta ao usuário quatro tipos diferentes de interfaces gráficas, além de uma interface mais simplificada, por linha de comando:

- *Explorer*: Disponibiliza ao usuário o acesso à barra de menu e suas opções, bem como a possibilidade de carregar os dados que serão utilizados e verificar os resultados

gerados pelos algoritmos de mineração. Uma das desvantagens do modo *Explorer* é que o conjunto de dados utilizado no processo é mantido em memória, limitando o uso a problemas de pequeno e médio porte.

- *Experimenter*: Tem como objetivo facilitar a identificação dos métodos e parâmetros nas técnicas de classificação e regressão mais adequados para determinado problema. Essa interface torna o processo experimental automático, o armazenamento das estatísticas pode ser realizado no formato ARFF e os arquivos gerados podem ser objeto de uma nova exploração de dados.

- *KnowledgeFlow*: Os usuários selecionam componentes *Weka* a partir de uma barra de ferramentas, colocando-os em uma tela de *layout* que os conectam a um gráfico responsável pelo processamento e análise dos dados. Esta interface fornece uma alternativa ao *Explorer*, pois analisa como os dados fluem através do sistema, além de permitir o *design* e a execução de configurações para processamento de dados em fluxo por componentes conectados - que representam as fontes de dados - ferramentas de pré-processamento, algoritmos de mineração, métodos de avaliação e módulos de visualização.

- *Workbench*: Ambiente que combina todas as interfaces GUI (*Graphical User Interface*) em uma única interface. É útil se o usuário precisa alternar entre duas ou mais interfaces distintas com frequência.

- *Simple CLI*: Apresenta orientações de como utilizar o *Weka* através da linha de comando (via Terminal no GNU/Linux/Mac OS ou *Prompt* de Comando no Windows) e permite ao usuário informar os comandos desejados na mesma janela. Tal funcionalidade se diferencia por conferir a possibilidade de escrever *shell scripts* chamadas de linha de comando com parâmetros. Dessa forma, o usuário é capaz de criar modelos, executar experimentos e realizar previsões sem uma GUI (*Graphical User Interface*).

2.6 *Orange*

O *Orange* [32] é um conjunto de ferramentas de visualização de dados, aprendizado de máquina e mineração de dados. Foi criado na Universidade de Ljubljana, Eslovênia. O início de seu desenvolvimento se deu em 1997 por Janez Demšar and Blaž Zupan. Possui uma interface de programação visual para análise explorativa e visualização interativa de dados. Também pode ser usado como uma biblioteca *Python*. Foi escolhida para realizar clusterização dos dados neste trabalho por ser *open source* e facilitar a visualização dos resultados.

2.7 Mercado de ações

2.7.1 O que é uma ação?

De acordo com Rassier e Hilgert [33], *"ações são títulos nominativos, negociáveis, que representam uma fração do capital social de uma empresa. Ao comprar uma ação, o investidor se torna sócio da empresa, ou seja, de um negócio. Passa a correr o risco dos lucros e prejuízos como qualquer empresário"*.

Feitosa [34] explica que o capital social pode ser definido como o valor que os sócios ou acionistas da empresa estabelecem para o negócio, quando este é iniciado. Pode ser definido como a quantia inicial bruta investida, capital este que é necessário para o início das atividades da nova empresa, tendo em vista que seus lucros iniciais não serão suficientes para a sustentação do negócio.

Ações são, então, subdivisões de determinada empresa que são negociadas no mercado.

2.7.2 O mercado de ações

Rassier e Hilgert [33] definem que o indivíduo que compra uma ação na bolsa de valores pode ser chamado de acionista minoritário, visto que a porcentagem da empresa que lhe diz respeito não lhe garante poder de controle sobre as decisões a serem tomadas.

Uma característica do mercado de ações que o torna atrativo é a ausência de um nível alto de burocracia no que diz respeito à compra e venda; este fato é uma vantagem clara, tendo em vista que a entrada ou saída de uma empresa limitada ou de capital fechado exige todo um processo de alteração de contratos sociais. Esta facilidade de compra e venda configura a liquidez do mercado de ações. Tal liquidez permite ao investidor migrar entre negócios de forma simples e fácil, dependendo da atratividade destes (Rassier e Hilgert)[33]. A busca por segurança de investimentos faz com que empresas ditas "bem geridas" sejam mais atrativas, dado que estas, geralmente, apresentam lucros sólidos e crescentes.

2.7.3 Tipos de ações

De forma geral, as ações comercializadas no mercado podem ser divididas em dois tipos: ações ordinárias (ON) e ações preferenciais (PN).

De acordo com Rassier e Hilgert [33], uma ação ordinária é *"o tipo de ação que concede o direto de voto nas assembleias da companhia ao seu detentor. [...] na maioria das vezes ele não tem poder de veto"*, enquanto que as ações preferenciais são aquelas que dão ao acionista a preferência no recebimento dos dividendos. Entretanto, segundo Rassier e Hilgert [33] *"são aquelas que menos protegem o acionista minoritário, porque não lhe dão*

o direito de votar em assembleia e, ainda, em caso de venda da empresa, a lei não lhe garante o direito de participar do prêmio de controle, também chamado de tag along, que é a diferença entre os preços de uma ação de controle (ações pertencentes aos acionistas que comandam a empresa) e uma ação comum (ações pertecentes a acionistas minoritários) [35]. *O prêmio de controle é pago em caso de mudanças no controle da companhia. Se caracteriza como um mecanismo de proteção aos acionistas minoritários.".*

No caso das ações ON, o direito de voto que é conferido aos acionistas minoritários ganha maior relevância em casos de divergências entre os acionistas majoritários. Em situações como esta, o detentor de ações ON pode, dependendo da circunstância e juntamente com outros acionistas minoritários (também detentores de ações ON), vir a ter o direito a veto.

No que diz respeito às ações do tipo PN, apesar dos contras que as acompanham, estas também apresentam determinadas vantagens ao acionista preferencialista (como são chamados os detentores de ações PN). Estes têm preferência no recebimento dos dividendos quando a empresa apresenta lucro e, no caso do não pagamento de dividendos por três anos consecutivos, adquirem direito a voto. Ações do tipo PN são típicas do mercado brasileiro.

Ações ordinárias são representadas pelo número "3"após o código do ativo (PETR3, por exemplo) enquanto que as ações preferenciais são representadas pelo número "4"(PETR4, por exemplo).

2.7.4 Bovespa

Bolsas de valores são instituições administradoras de mercados. De acordo com Rassier e Hilgert [33], a Bolsa de Valores de São Paulo (Bovespa) caracteriza-se como o único centro de negociação de ações do Brasil e o maior da América Latina. Foi fundada em 1890 e tem sua sede no centro de São Paulo.

O Brasil já chegou a ter, no passado, nove bolsas de valores, mas hoje, as duas principais são a Bovespa (Bolsa de Valores de São Paulo) e a BM&F (Bolsa de Mercadorias & Futuros). Na Bovespa são negociadas ações de companhias abertas e títulos privados de renda fixa, entre outros valores mobiliários. A BM&F tem destaque por se tratar de uma das maiores bolsas de mercadorias e futuros do mundo. Nela são negociados contratos derivativos agropecuários (*commodities*) e derivativos financeiros.

O principal índice no que diz respeito às ações do mercado brasileiro é o Ibovespa, constituído em 1968 pela própria Bovespa. Este índice tem por objetivo medir o desempenho das ações com maior volume negociado num período de 12 meses.

2.8 Análise Fundamentalista

De acordo Miltersteiner [36], a análise fundamentalista é uma ferramenta de suma importância no que diz respeito à análise de investimentos em ações. É baseada no cálculo de relações entre os fundamentos econômico-financeiros e o desempenho da empresa. Tal relação tenta determinar o valor real da empresa.

A análise fundamentalista faz uso de uma série de índices ou indicadores. Cada empresa tem seu valor de mercado representado pelo valor da cotação de suas ações da Bolsa de Valores multiplicado pelo número total de ações que compõe seu capital.

2.8.1 Principais Campos do Balanço Patrimonial e Demonstrativo Financeiro

Ativo

Segundo Silva e Rodrigues(2018)[37] é definido como os recursos que agregam algum valor futuro para uma determinada entidade.

Ativo Circulante

Segundo Silva e Rodrigues(2018)[37] são aqueles recursos que irão se transformar em dinheiro e poderão ser usados dentro de um período anual.

Ativo Não Circulante

Segundo Silva e Rodrigues(2018)[37] são aqueles recursos que não irão se transformar em dinheiro e não poderão ser usados dentro de um período anual.

Ativo Realizável a Longo Prazo

Segundo Silva e Rodrigues(2018)[37] é um tipo de ativo não circulante, que serão tranformados em dinheiro após um ano.

Intangíveis

Segundo Silva e Rodrigues(2018) [37] é um dos tipos de ativos utilizados pelas entidades no exercício de suas atividades operacionais, como por exemplo: patentes, gastos com pesquisa e desenvolvimento.

Ativo Total

Segundo Silva e Rodrigues(2018)[37] é a composição dos Ativos Circulante e Não Circulante.

Lucro Líquido

De Acordo com Silva e Rodrigues (2018) [37] é o resultado apurado após computadas as receitas e despesas financeiras, o resultado dos impostos sobre os lucros e as participações no lucro. É o resultado que efetivamente pertence aos proprietários da empresa.

Patrimônio Líquido

Segundo Silva e Rodrigues(2018)[37] são todos os recursos que estão atrelados aos prorietários ou acionistas de uma entidade.

Dividendos

Segundo Silva e Rodrigues(2018) [37] dividendos é a distribuição de lucros para os proprietários ou acionistas de uma entidade.

Passivo

É definido por Silva e Rodrigues(2018) [37] como as obrigações das entidades com terceiros.

Passivo Circulante

Segundo Silva e Rodrigues(2018)[37] são as obrigações de curto prazo da entidade, que até fim do exercicío social(período de doze meses) deverão ser quitadas.

Passivo Não Circulante

Segundo Silva e Rodrigues(2018)[37] são as que serão quitadas após o fim do exercício social(período de doze meses) de uma entidade.

Passivo Total

Segundo Silva e Rodrigues(2018)[37] é a composição dos Passivos Circulantes e Não Circulantes.

Receita Financeira

Segundo Silva e Rodrigues(2018)[37] representa os valores recebidos através de transações financeiras, tais como juros de uma aplicação, ou descontos obtidos na compra de um produto ou serviço.

Resultado antes das tributações e participações

Segundo Silva e Rodrigues(2018)[37] é o lucro ou prejuízo obtido antes do pagamento das tributações(impostos) e debêntures, empregados e administradores, partes beneficiárias e de fundos de assistência e previdência aos empregados.

Empréstimo e Financimantos

De acordo com Silva e Rodrigues(2018) [37] os empréstimos e financiamentos correspondem a dívidas que a entidade assumiu com uma instituição financeira, é uma conta que faz parte do passivo circulante.

2.8.2 Principais Indicadores Fundamentalistas de Mercado

Preço/Lucro (P/L)

Segundo Gitman [38] indica o quanto especialistas financeiros estão dispostos a pagar a cada dolár de lucro da entidade, expressando a convicção que os especuladores colocam no desempenho a longo prazo da empresa. Quanto maior o P/L maior a confiança, desta forma, é utilizado para graduar o prazo de empreendimentos em conexão com a estima da ação.

$$Preço/Lucro = \frac{Preço\ de\ mercado\ da\ ação}{Lucro\ por\ ação} \tag{2.1}$$

Preço/Valor Patrimonial (P/VPA)

Segundo Gitman [38] relaciona o valor de mercado ao contábil das ações dando uma visão de como os especialistas financeiros veem a execução da entidade.

$$P/VPA = \frac{Preço\ de\ mercado\ da\ ação}{Valor\ patrimônial\ da\ ação} \tag{2.2}$$

Taxa de distribuição de lucros (PayOut)

Segundo Speranzini [39] para um período especificado indica razão entre a participação dos dividendos e ao lucro líquido da empresa.

$$Payout = \frac{Dividendos\ período}{Lucro\ Líquido\ período} \tag{2.3}$$

Dividend Yield (taxa de distribuição de dividendos)

Segundo Speranzini [39] é a rentabilidade dos dividendos de determinada empresa em relação ao preço de suas ações.

$$Div\ Yield = \frac{Dividendo\ pago\ por\ ação}{Preço\ de\ mercado\ da\ ação} \tag{2.4}$$

Liquidez Corrente

Segundo Gitman [38] no curto prazo, mede o quanto a empresa é capaz de saldar seus compromissos financeiros. O valor do índice é diretamento proporcional à liquidez da entidade, sendo assim, quanto maior o valor, maior a liquidez.

$$Liquidez\ Corr = \frac{Ativo\ Circulante}{Passivo\ Circulante} \tag{2.5}$$

Liquidez Seca

Segundo Gitman [38] para os casos em que é díficil a conversão de estoque em dinheiro, este índice oferece uma medida melhor de liquidez, em comparação ao índice de liquidez corrente.

$$Liquidez\ Seca = \frac{Ativo\ Circulante - Estoques}{Passivo\ Circulante} \tag{2.6}$$

Rentabilidade do Capital Próprio (ROE)

Segundo Gitman [38] quanto maior esse índice melhor financeiramente para os acionistas, pois o mesmo mensura o quanto do investimento do capital dos acionistas ordinários da empresa é recuperado por eles.

$$ROE = \frac{Lucro\ Líquido}{Patrimônio\ Líquido} \tag{2.7}$$

Endividamento Geral

Segundo Gitman [38] o valor deste índice é diretamente proporcional ao volume relativo de capital de outros investidores usado para gerar lucros, e também ao envididamento geral

da empresa. Pois ele mensura a proporção dos ativos totais financiada pelos credores da empresa.

$$Endividamento\ geral = \frac{Passivo\ Exigível\ Total}{Ativo\ Total} \tag{2.8}$$

Giro do Ativo Total

Segundo Gitman [38] indica o quanto a empresa é capaz de fazer com que os seus ativos gerem receitas líquidas.

$$Giro\ Ativo\ Total = \frac{Receita\ Líquida}{Ativo\ Total} \tag{2.9}$$

Retorno do Ativo Total (ROA)

Segundo Gitman [38] mensura o quão eficaz está administração de uma empresa quanto a capacidade de gerar lucro atrávez dos ativos disponíveis. É chamado também de retorno de investimento (ROI).

$$ROA = \frac{Lucro\ Líquido}{Ativo\ Total} \tag{2.10}$$

Relação Preço Venda (PSR)

Segundo Vruwink [40] indica o quanto uma empresa está propensa a obter retornos maiores que a média devido a baixa avaliação do mercado em relação às vendas.

$$PSR = \frac{Venda\ dos\ últimos\ doze\ meses}{Total\ Ações} \tag{2.11}$$

2.8.3 Fundamentus

Dentre os recursos de pesquisa utilizados, o site Fundamentus [3] foi um dos principais provedores de dados do projeto. Este é um sistema online que tem como principal objetivo a disponibilização de informações financeiras e fundamentalistas de empresas que têm suas ações listadas na Bovespa. É um site voltado para investidores, ao passo que disponibiliza opções de investimentos. O acesso aos dados se dá através da pesquisa (utilizando-se o nome da empresa na barra de pesquisa do site) e consulta por meio das seções presentes nas abas localizadas na parte superior da tela.

Ao utilizar o sistema, é possível ao usuário obter os principais indicadores fundamentalistas e financeiros das empresas (o que lhes permite uma leitura do estado econômico-financeiro em que a empresa se encontra assim como uma análise do nível do preço de mercado desta), comparar empresas levando em consideração seus setores e subsetores de atuação, realizar o *download* de planilhas Excel de balanços históricos de determinadas empresas, visualizar gráficos das cotações e de alguns dos indicadores.

A contribuição da plataforma para o trabalho consistiu especificamente no acesso aos balanços históricos das empresas escolhidas para análise. Como parte do projeto consiste na observação dos indicadores fundamentalistas, o critério para a decisão da utilização ou não do balanço de determinada empresa baseava-se na presença do cálculo destes indicadores, ou seja, se o balanço de determinada empresa tem os indicadores fundamentalistas calculados, seu balanço histórico segue o padrão procurado para os dados.

2.8.4 Yahoo Finanças

O Yahoo Finanças [4] é uma vertente do Yahoo Brasil que tem como objetivo a disponibilização de informações acerca do mercado financeiro (foco principal nas cotações das ações de empresas).

O acesso e posterior uso dos dados disponibilizados se dá da seguinte forma: pesquisa-se pelo nome da empresa de interesse na barra de pesquisa do site e, então, consulta-se os dados que estão divididos em seções nas abas localizadas abaixo do nome da empresa. Dentre as informações trazidas pela plataforma encontram-se receitas e ganhos das empresas, tendências de recomendação (uma espécie de termômetro para a compra, retenção ou venda de ações), classificação destas recomendações, informações acerca de elevações e rebaixamentos de ações no mercado, entre outras.

A contribuição da plataforma para a pesquisa consistiu na disponibilização e acesso aos dados de cotação de ações das empresas escolhidas. Tais dados localizam-se na aba dados históricos. Por meio deste recurso é possível acessar dados de períodos específicos em regularidade diária, semanal ou mensal. Para o uso no trabalho o período dos dados a serem obtidos foi formatado da seguinte maneira: 01/02/2014 até 31/07/2017. A regularidade utilizada para os dados foi mensal. Desta forma, as datas dos dados obtidos estiveram de acordo com as datas dos dados aos quais foram relacionados (do balanço histórico).

2.9 Trabalhos Correlatos

Existem vários trabalhos na área de Mineração de Dados que lidam com dados da bolsa de valores e correlações.

Berestein [41] apresenta uma tentativa de descoberta de padrões e correlações significativas sobre dados da bolsa de valores utilizando-se de técnicas de mineração de dados (mais especificamente, a classificação) e apresenta resultados obtidos ao fazer uso do algoritmo J48.

Branco e Barroso [42] propõem o uso do software *Mining Stock* para a utilização de técnicas de mineração de dados aliadas à análise de sentimento com o objetivo de definir

tendências para determinadas ações no dia de negociação de forma a entender o nível de influência de uma notícia no preço acionário.

Leite [43] também apresenta uma tentativa de previsão do comportamento de ações fazendo uso de dados históricos, tendo por motivação maior uma eventual previsão dos valores de venda ou compra de determinado volume de ações. O foco principal é sobre as empresas do ramo de distribuição de energia elétrica.

O presente trabalho busca, também, a descoberta de padrões e correlações significativas sobre dados de cotações da bolsa de valores, mas, diferentemente dos citados nesta seção, além dos dados históricos e balanços das empresas presentes na bolsa de valores, nosso estudo também faz uso de indicadores fundamentalistas e econômicos para uma análise mais robusta utilizando técnicas de classificação e clusterização para descoberta de padrões. Além de análises mais generalizadas, foram feitas, também, análises por grupos de ações. Os grupos utilizados foram: Carnes e Derivados; Petróleo, Gás e Biocombustíveis; Tecidos e Vestuário.

Capítulo 3

Análise dos dados

A obtenção, tratamento e posterior análise dos dados para o trabalho se deu a partir do método proposto no manual CRISP-DM produzido pela IBM [14] (Seção 2.2). O desenvolvimento será descrito a seguir.

3.1 *Business Understanding* (Compreensão do Negócio)

O processo de *Business Understanding*, também conhecido como *Compreensão do Negócio* começou quando os objetivos do trabalho foram definidos. Sabia-se que seriam utilizados balanços de empresas presentes na bolsa de valores. Então, ao estudar possíveis resultados relevantes para a pesquisa, chegou-se à conclusão de que seriam buscados indicadores que auxiliassem na previsão dos valores das ações das empresas citadas. Como tal objetivo ainda era bastante abrangente, restrigimo-nos aos seguintes aspectos: estudo das correlações entre as empresas, suas respectivas ações e seus indicadores fundamentalistas, classificação destas utilizando-nos do *software Weka* e clusterização a partir do *software Orange*.

3.2 *Data Understanding* (Compreensão dos Dados)

O processo de *Data Understanding*, também conhecido como *Compreensão dos Dados* teve início a partir do levantamento dos dados que seriam necessários para a realização da análise. Era sabido que tais dados estariam disponíveis a partir de fontes diferentes e que haveria um trabalho de reunião e curadoria para que pudéssemos seguir para a etapa de preparação. Desta forma, foi-se ao encontro dos dados e estes foram coletados como descrito a seguir.

3.2.1 Balanços e Demonstrativos das Empresas

Os balanços com os dados das empresas utilizadas no trabalho foram obtidos através do site Fundamentus [3]. A condição principal para que a empresa fosse escolhida para fazer parte do grupo a ser estudado foi a presença de todos os indicadores fundamentalistas em seu balanço (tendo em vista que tais indicadores foram um dos grandes focos de análise na pesquisa) e, ainda, que a data da última cotação apresentada pela empresa fosse maior ou igual a 30/06/2017 (limite traçado para a análise). Caso o balanço de uma das empresas selecionadas possuísse dados para além da data limite (30/06/2017), as colunas seriam apagadas, de forma a manter a padronização dos dados utilizados.

Os arquivos de balanço das empresas estavam, por padrão, divididos em duas abas. As informações acerca das ações deveriam ser adicionadas na aba cujo nome era "Bal. Patrim.", na célula A2. Outra observação acerca dos arquivos de balanço é que as informações eram apresentadas de forma trimestral.

Como os dados do balanço estavam todos divididos por 1000 (por questões de otimização da visualização dos dados) o valor das ações também tiveram que ser divididos por 1000. Desta forma, para que fossem obtidos os valores reais (para fins de análise), teria-se que multiplicar os valores por 1000.

3.2.2 Cotações Mensais

As cotações das empresas utilizadas foram obtidas através do site Yahoo! Finanças [4]. Como o período definido como escopo do trabalho foi de 01/02/2014 à 31/07/2017, o período compreendido pelos dados das cotações deveriam estar de acordo. Desta forma, foi realizada a verificação do retorno dos dados para confimar se o período estava correto, isto é, se o espaço de tempo compreendido pelos dados era de 01/02/2014 à 31/07/2017. Felizmente o site Yahoo! Finanças [4] apresenta uma ferramenta de filtro, que foi de grande ajuda na seleção e obtenção do conjunto correto de dados.

Uma observação acerca das cotações é que estes dados eram apresentados de forma mensal. As planilhas obtidas no site estavam no formato *.csv* e, no ato do download, deveriam ser colocadas em uma pasta chamada *cotação*, com o nome igual ao do balanço correspondente.

3.2.3 Cotações Diárias

Parte da análise dos dados requisitou a presença dos dados diários das cotações das empresas estudadas. Tal requisição foi atendida pelo *script* em *python* I.4 em anexo, que teve como função realizar o *download* de todas as cotações do período compreendido entre

2014 até 2017 (anos completos) de todas as empresas presentes no Ibovespa e que tivessem seus dados contidos no Yahoo! Finanças [4].

3.2.4 Taxa de Juros

As informações mensais acerca da *Taxa de Juros Selic* foram obtidas através do site da Receita Federal [44]. Para cada mês, calculou-se a taxa de juros reais fazendo uso da fórmula 3.1 (a seguir):

$$(\frac{1 + selic}{1 + Inflaçao} - 1) * JurosReais = \frac{1 + JurosNominais}{1 + Inflação} - 1 \tag{3.1}$$

Segundo Dornbush e Fischer (1991)[45]:

> *"o investimento é endógeno a taxa de juros. A taxa planejada ou desejada de investimento é menor quanto mais alta for a taxa de juros. Por um outro lado, uma taxa de juros baixa torna os gastos com investimentos lucrativos, e, portanto reflete em um alto nível de investimento planejado"*

. Portanto a taxa de juros é uma das varíaveis que têm influência direta sobre o mercado, por isso a mesma foi incluída na análise.

3.2.5 IGP-M

O IGP-M/FGV (Índice Geral de Preços – Mercado) é um índice calculado mensalmente pela FGV e é divulgado no final de cada mês de referência. Segundo a metodologia divulgada pela FGV IBRE (Instituto Brasileiro de Economia) [46] este índice bastante utilizado por refletir a evolução dos preços de atividades produtivas passíveis de serem sistematicamente pesquisados. No presente trabalho, este foi utilizado para mensurar a inflação e deflacionar indicadores e cotações. Isto foi feito com o objetivo deixar os dados mais reais, tendo em vista que, se tal medida não tivesse sido tomada, os valores das cotações pareceriam melhores do que realmente são.

3.2.6 Outros Indicadores

Além dos indicadores fundamentalistas e os demais indicadores citados anteriormente, para compor a análise dos dados também foram utilizados alguns indicadores sociais e econômicos. Tais indicadores serviram para, em conjunto com os demais, promover um entendimento maior do cenário no qual as empresas estudadas estavam inseridas e, assim, proporcionar um número maior de possibilidades de análise.

Os indicadores sociais utilizados foram obtidos através do site do IBGE (Instituto Brasileiro de Geografia e Estatística) [47]. Já os indicadores econômicos foram retirados do site do Banco Central do Brasil [48].

Taxa de desocupação

Mede a proporção de pessoas que estão sem trabalhar em relação as pessoas que estão efetivamente trabalhando em um período [49].

Indicadores da conjuntura econômica

São indicadores que mostram como se comporta a economia do país relacionada a política fiscal, emprego e renda, inflação, juros, crédito, setor externo, além de muitos outros [50].

Expectativa Consumidor

Segundo o Portal da Indústria [51] a expectativa do consumidor é um indicador de confiança do consumidor e é utilizado para identificar a tendência do consumo das famílias.

3.3 *Data Preparation* (Preparação dos Dados)

O processo de *Data Preparation*, também conhecido como *Preparação dos Dados* teve seu início após o processo de coleta. Tal preparação, como citado na descrição do método CRISP-DM [14], é um dos aspectos mais importantes e mais demorados da mineração de dados. Foi aqui que os diversos conjuntos de dados coletados foram fundidos (de forma a facilitar o processo de análise), selecionados (descartando-se o que não seria útil ao processo) e preparados para as próximas fases do trabalho. Tais processos serão descritos a seguir.

3.3.1 Processo de alteração e estruturação dos arquivos *.xls* e *.csv*

Os arquivos obtidos através do site Fundamentus (chamados de balanços históricos) se encontravam no formato .xls (formato de arquivo do software Microsoft Office - Excel). A conversão para *.csv* foi necessária para que estes arquivos pudessem ser lidos no *Weka* e no *Orange*.

O processamento das planilhas se deu através de *scripts* em linguagem de programação *python*.

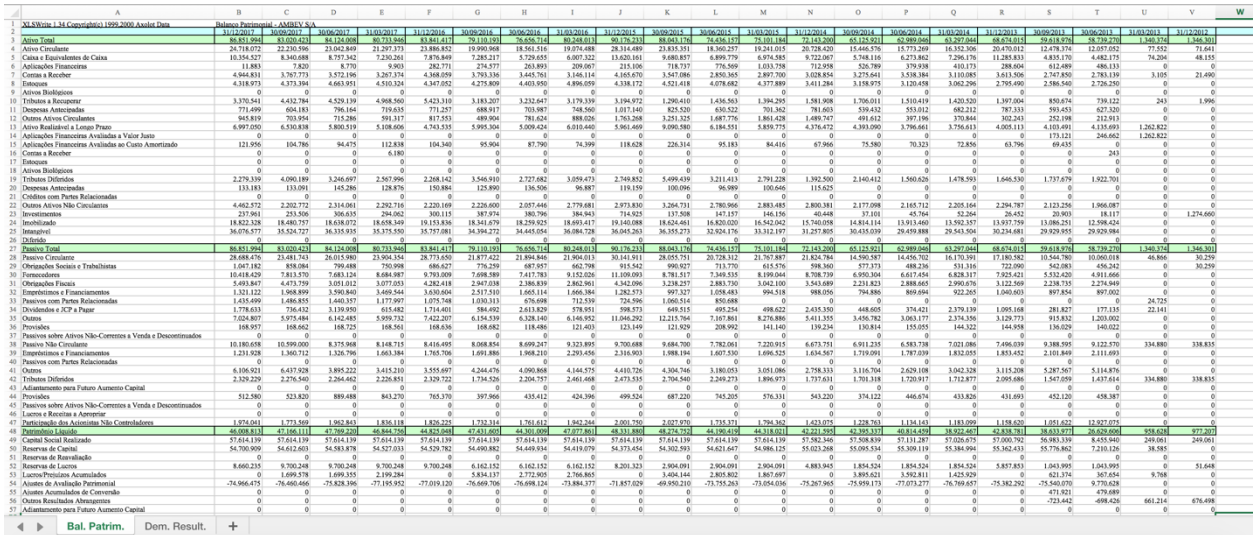
Os balanços obtidos foram programados para conter os dados de 2013 até 2017. Os dados que, de fato, foram utilizados para a pesquisa vão de 2014 até 2017; os dados de 2013 serviram para o cálculo dos indicadores fundamentalistas de 2014 (estes requerem informações do ano anterior para serem calculados).

Os arquivos são compostos por duas planilhas: Balanço e Demonstrativos. Para o uso no projeto foram necessárias algumas alterações na estruturação das planilhas.

Primeiramente, os dados contidos no arquivo foram unificados em uma só planilha, de forma que todos estivessem disponíveis na mesma aba. Nesta planilha única encontravam-se os dados das planilhas 'balanço', 'demonstrativos', bem como os dados históricos das cotações da empresa em questão.

Após isso, foi acrescentada na primeira célula da planilha (coluna A, linha 1) o número de ações (para cada uma das empresas). A alteração foi realizada de forma manual. A escolha da célula se deu pelo fato dela estar em branco em todos os arquivos obtidos. O processo foi necessário pois o número de ações é fornecido pelo site Fundamentus[3], mas não está contida nas planilhas (balanço patrimonial e demonstrativo financeiro) obtidas neste site.

Outra alteração realizada foi a inversão da disposição dos dados. As Figuras 3.1 e 3.2 abaixo mostram um exemplo do formato inicial das planilhas (no momento de sua obtenção):



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	XLSWrite 1.34 Copyright(c) 1999-2000 Axolot Data	Balanco Patrimonial - AMBEV S/A																					
2		31/12/2017	30/09/2017	30/06/2017	31/03/2017	31/12/2016	30/09/2016	30/06/2016	31/03/2016	31/12/2015	30/09/2015	30/06/2015	31/03/2015	31/12/2014	30/09/2014	30/06/2014	31/03/2014	31/12/2013	30/09/2013	30/06/2013	31/03/2013	31/12/2012	
3	Ativo Total	86.851.994	83.020.423	84.124.008	80.733.946	83.841.417	79.110.193	76.656.714	80.248.013	90.176.233	88.043.176	74.436.157	75.101.184	72.143.200	65.125.921	62.989.046	63.297.044	68.674.015	59.618.976	58.739.270	1.340.374	1.346.301	
4	Ativo Circulante	24.718.072	22.230.596	23.042.849	21.297.373	23.886.852	19.990.968	18.561.516	19.074.488	28.314.489	23.835.351	18.360.257	19.241.015	20.728.420	15.446.576	15.773.269	16.352.306	20.470.012	12.478.374	12.057.052	77.552	71.641	
5	Caixa e Equivalentes de Caixa	10.354.527	8.340.688	8.757.342	7.230.261	7.878.849	7.285.217	5.729.655	6.007.322	13.620.161	9.680.857	6.899.779	6.974.585	9.722.087	5.748.116	6.273.862	7.296.178	11.285.833	4.835.170	4.482.175	74.204	48.155	
6	Aplicações Financeiras	11.883	7.820	8.770	9.903	282.771	274.577	263.893	209.067	215.106	718.737	776.569	1.033.758	712.958	526.789	379.938	410.173	288.604	612.489	480.133	0	0	
7	Contas a Receber	4.944.831	3.767.773	3.572.196	3.267.374	4.368.059	3.793.336	3.445.761	3.146.114	4.165.670	3.547.086	2.850.365	2.897.700	3.028.854	3.275.641	3.538.384	3.110.085	3.613.506	2.747.850	2.783.139	3.105	21.490	
8	Estoques	4.318.973	4.373.394	4.663.951	4.510.324	4.347.052	4.275.809	4.403.950	4.896.059	4.338.172	4.521.418	4.078.682	4.377.889	3.411.284	3.158.975	3.120.458	3.062.296	2.795.490	2.586.540	2.726.250	0	0	
9	Ativos Biológicos	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
10	Tributos a Recuperar	3.370.541	4.432.784	4.529.139	4.968.560	5.423.310	3.183.207	3.232.647	3.179.339	3.194.972	1.290.410	1.436.563	1.394.295	1.581.908	1.706.011	1.310.419	1.420.520	1.397.004	850.674	739.122	243	1.996	
11	Despesas Antecipadas	771.499	604.183	796.164	719.635	771.257	688.917	703.987	748.560	1.017.140	825.520	630.522	701.362	781.603	539.432	553.012	682.212	787.333	593.453	627.320	0	0	
12	Outros Ativos Circulantes	945.819	703.954	715.286	591.317	817.553	489.904	781.624	888.026	1.763.268	3.251.325	1.687.776	1.861.428	1.489.747	491.612	397.196	370.844	302.243	252.198	212.913	0	0	
13	Ativo Realizável a Longo Prazo	6.997.050	6.530.838	5.800.519	5.108.606	4.743.535	5.995.304	5.009.424	6.010.440	5.961.460	9.090.580	6.184.615	5.859.775	4.376.472	4.393.090	3.796.661	3.756.613	4.005.113	4.103.491	4.135.693	1.262.822	0	
14	Aplicações Financeiras Avaliadas a Valor Justo	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	173.121	246.662	1.262.822	0	
15	Aplicações Financeiras Avaliadas ao Custo Amortizado	121.956	104.786	94.475	112.838	104.340	95.904	87.790	74.399	118.628	226.314	95.183	84.416	67.966	75.580	70.323	72.856	63.796	69.435	0	0	0	
16	Contas a Receber	0	0	0	6.180	0	0	0	0	0	0	0	0	0	0	0	0	0	0	243	0	0	
17	Estoques	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
18	Ativos Biológicos	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
19	Tributos Diferidos	2.279.339	4.090.189	3.246.697	2.567.996	2.268.142	3.546.910	2.727.682	3.059.473	2.749.852	5.499.439	3.211.413	2.791.228	1.392.500	2.140.412	1.560.626	1.478.593	1.646.530	1.737.679	1.922.701	0	0	
20	Despesas Antecipadas	133.183	133.091	145.286	128.876	150.884	125.890	136.506	96.887	119.159	100.096	96.989	100.646	115.625	0	0	0	0	0	0	0	0	
21	Créditos com Partes Relacionadas	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
22	Outros Ativos Não Circulantes	4.462.572	2.202.772	2.314.061	2.292.716	2.220.169	2.226.600	2.057.446	2.779.681	2.973.830	3.264.731	2.780.966	2.883.485	2.800.381	2.177.098	2.165.712	2.205.164	2.294.787	2.123.256	1.966.087	0	0	
23	Investimentos	237.961	253.506	306.635	294.062	300.115	387.974	380.796	384.943	714.925	137.508	147.157	146.156	40.448	37.101	45.764	52.264	26.452	20.903	18.117	0	1.274.660	
24	Imobilizado	18.822.328	18.480.757	18.638.072	18.658.349	19.153.836	18.541.679	18.259.925	18.693.417	19.140.088	18.624.681	16.820.020	16.542.042	15.740.058	14.814.114	13.913.460	13.592.357	13.937.759	13.086.251	12.598.424	0	0	
25	Intangível	36.076.577	35.524.727	36.335.935	35.375.550	35.757.081	34.394.272	34.445.054	36.084.728	36.045.263	36.355.273	32.924.176	33.312.197	31.257.805	30.435.039	29.459.888	29.543.504	30.234.681	29.929.955	29.929.984	0	0	
26	Diferido	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
27	Passivo Total	86.851.994	83.020.423	84.124.008	80.733.946	83.841.417	79.110.193	76.656.714	80.248.013	90.176.233	88.043.176	74.436.157	75.101.184	72.143.200	65.125.921	62.989.046	63.297.044	68.674.015	59.618.976	58.739.270	1.340.374	1.346.301	
28	Passivo Circulante	28.688.476	23.481.743	26.015.980	23.904.354	28.773.650	21.877.422	21.894.846	21.904.013	30.141.911	28.055.751	20.728.312	21.767.887	21.824.784	14.590.587	14.456.702	16.170.391	17.180.582	10.544.780	10.060.018	46.866	30.259	
29	Obrigações Sociais e Trabalhistas	1.047.182	858.084	799.488	750.998	686.627	776.259	687.957	662.798	915.542	990.927	713.770	615.576	598.360	577.373	488.236	531.316	722.090	542.083	456.242	0	30.259	
30	Fornecedores	10.418.429	7.813.570	7.683.124	8.684.987	9.793.009	7.698.589	7.417.783	9.152.026	11.109.093	8.781.517	7.349.535	8.199.044	8.708.739	6.950.304	6.617.454	6.828.317	7.925.421	5.532.420	4.911.666	0	0	
31	Obrigações Fiscais	5.493.847	4.473.759	3.051.012	3.077.053	4.282.418	2.947.038	2.386.839	2.862.961	4.342.096	3.238.257	2.883.730	3.042.100	3.543.689	2.231.823	2.888.665	2.990.676	3.122.569	2.238.735	2.274.949	0	0	
32	Empréstimos e Financiamentos	1.321.122	1.988.899	3.590.840	3.469.544	3.630.604	2.517.510	1.665.114	1.666.384	1.282.573	997.327	1.058.483	994.518	988.056	794.886	869.694	922.265	1.060.603	897.854	897.002	0	0	
33	Passivos com Partes Relacionadas	1.435.499	1.486.855	1.440.357	1.177.997	1.075.748	1.030.313	676.698	712.539	724.596	1.060.514	850.688	0	0	0	0	0	0	0	0	24.725	0	
34	Dividendos e JCP a Pagar	1.778.633	736.432	3.139.950	655.482	1.714.401	584.492	2.613.829	578.951	598.573	649.515	495.254	498.622	2.435.350	448.605	374.421	2.379.139	1.095.168	281.827	177.135	22.141	0	
35	Outros	7.024.807	5.975.484	6.162.485	5.959.732	7.422.207	6.154.539	6.328.140	6.166.952	11.046.292	12.215.764	7.167.861	8.276.886	5.411.355	3.456.782	3.063.177	2.374.356	3.129.773	915.832	1.203.002	0	0	
36	Provisões	168.957	168.662	168.725	168.561	168.636	168.682	118.486	121.403	123.149	121.929	208.992	141.140	139.234	130.814	155.055	144.322	144.958	136.029	140.022	0	0	
37	Passivos sobre Ativos Não-Correntes a Venda e Descontinuados	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
38	Passivo Não Circulante	10.180.658	10.599.000	8.375.968	8.148.715	8.416.495	8.068.854	8.699.247	9.323.895	9.700.688	9.684.700	7.782.061	7.220.915	6.673.751	6.911.235	6.583.738	7.021.086	7.496.039	9.388.595	9.122.570	334.880	338.835	
39	Empréstimos e Financiamentos	1.231.928	1.360.712	1.326.796	1.663.384	1.765.706	1.691.886	1.968.210	2.293.456	2.316.903	1.988.194	1.607.530	1.696.525	1.634.567	1.719.091	1.787.039	1.832.055	1.853.452	2.101.849	2.111.693	0	0	
40	Passivos com Partes Relacionadas	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
41	Outros	6.106.921	6.437.928	3.895.222	3.415.210	3.555.697	4.244.476	4.090.868	4.144.575	4.410.726	4.304.746	3.180.053	3.051.086	2.758.333	3.116.704	2.629.108	3.042.328	3.115.208	5.287.567	5.114.876	0	0	
42	Tributos Diferidos	2.329.229	2.276.540	2.264.462	2.226.851	2.329.722	1.734.529	2.204.757	2.461.468	2.473.515	2.704.540	2.249.273	1.896.973	1.737.631	1.701.318	1.720.917	1.712.877	2.095.686	1.547.059	1.437.614	334.880	338.835	
43	Adiantamento para Futuro Aumento Capital	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
44	Provisões	512.580	523.820	889.488	843.270	765.370	397.966	435.412	424.396	499.524	687.220	745.205	576.331	543.220	374.122	446.674	433.826	431.693	452.120	458.387	0	0	
45	Passivos sobre Ativos Não-Correntes a Venda e Descontinuados	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
46	Lucros e Receitas a Apropriar	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
47	Participação dos Acionistas Não Controladores	1.974.041	1.773.569	1.962.843	1.836.118	1.826.225	1.732.314	1.761.612	1.942.266	2.001.750	2.027.970	1.735.371	1.794.362	1.423.075	1.228.763	1.134.143	1.183.099	1.158.520	1.051.622	12.927.075	0	0	
48	Patrimônio Líquido	46.008.813	47.166.111	47.769.220	46.844.756	44.823.048	47.431.605	44.301.099	47.077.801	48.331.880	48.274.752	44.190.419	44.318.031	42.221.595	42.395.337	40.814.459	38.922.467	42.838.781	38.633.977	26.029.696	958.628	977.207	
49	Capital Social Realizado	57.614.139	57.614.139	57.614.139	57.614.139	57.614.139	57.614.139	57.614.139	57.614.139	57.614.139	57.614.139	57.614.139	57.614.139	57.582.346	57.508.839	57.131.287	57.026.675	57.000.792	56.983.339	8.455.940	249.061	249.061	
50	Reservas de Capital	54.700.909	54.612.603	54.583.878	54.527.033	54.529.782	54.490.882	54.449.934	54.419.079	54.373.454	54.302.593	54.621.647	54.986.125	55.023.268	55.095.534	55.309.119	55.384.994	55.362.433	55.776.862	7.210.126	38.585	0	
51	Reservas de Reavaliação	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
52	Reservas de Lucros	8.660.235	9.700.248	9.700.248	9.700.248	9.700.248	6.162.152	6.162.152	6.162.152	8.201.323	2.904.091	2.904.091	2.904.091	4.883.945	1.854.524	1.854.524	1.854.524	5.857.853	1.043.995	1.043.995	0	51.648	
53	Lucros/Prejuízos Acumulados	0	1.699.578	1.699.355	2.199.284	0	5.834.137	2.772.905	2.766.865	0	3.404.144	2.805.802	1.867.697	0	3.895.621	3.592.811	1.425.929	0	621.374	367.654	9.768	0	
54	Ajustes de Avaliação Patrimonial	-74.966.475	-76.460.466	-75.828.396	-77.195.952	-77.019.120	-76.669.706	-76.698.124	-73.884.377	-71.857.029	-69.950.210	-73.755.263	-73.054.036	-75.267.965	-75.959.173	-77.073.277	-76.769.657	-75.382.292	-75.540.070	9.770.628	0	0	
55	Ajustes Acumulados de Conversão	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	471.921	479.689	0	0	
56	Outros Resultados Abrangentes	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-723.442	-698.426	661.214	676.698	
57	Adiantamento para Futuro Aumento Capital	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Bal. Patrim. | Dem. Result. | +

Figura 3.1: Balanço Patrimonial

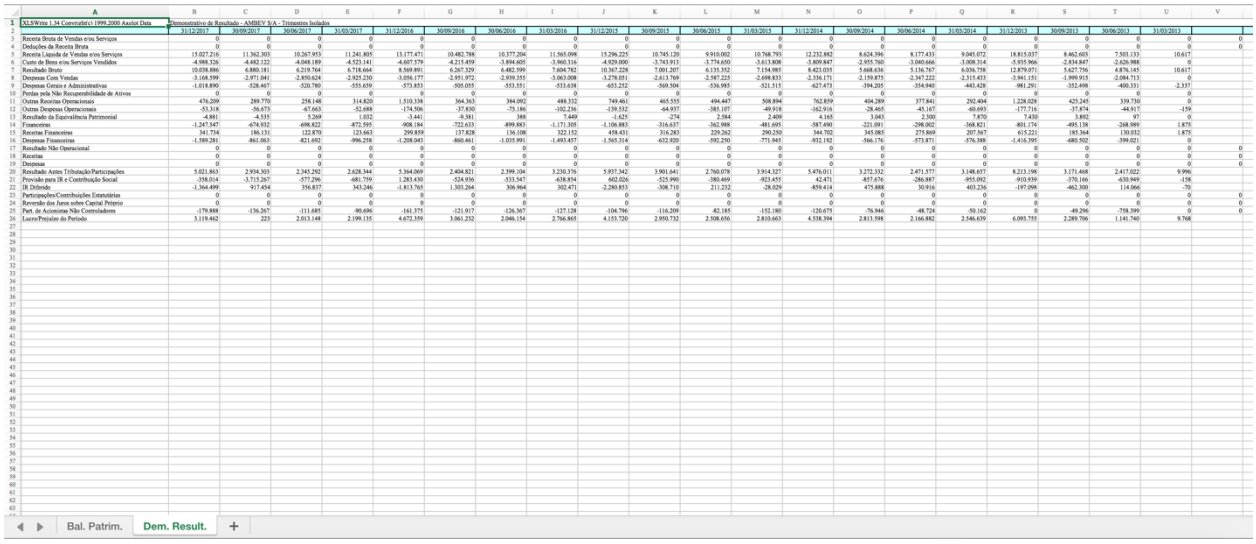


Figura 3.2: Demonstração de Resultado

As datas estão posicionadas na primeira linha, enquanto os descritores dos dados estão localizados na primeira coluna (Figuras 3.3 e 3.4). A inversão se deu de forma que datas e descritores trocassem de lugar; estes agora seriam colocados na primeira linha enquanto aqueles seriam posicionados na primeira coluna.

Por fim, cada uma das datas teve sua posição triplicada na planilha. Isto se deu pelo fato de que as informações no balanço se encontravam em periodicidade trimestral, ou seja, as datas saltavam de três em três meses enquanto os dados relacionados à cotação das ações de cada empresa eram mensais. Era necessário que todos os dados seguissem o mesmo regime. Assim, tal alteração ocasionou, também, a triplicação de todos os dados.

O processamento realizado entre planilhas é feito por posição no arquivo (e não por data). Assim, todas as planilhas utilizadas passaram pelo mesmo processo de formatação, de forma que a disposição dos dados fosse igual para todas.

O fato da organização não ser feita por data se deve à formatação dos campos presentes no arquivo obtido; estes nem sempre seguem o mesmo padrão (na planilha de cotações, por exemplo, a data 01/06/17 equivale a data 31/05/17 na planilha de balanço). Desta forma, foi utilizado o processamento por posição para evitar uma possível reformatação desnecessária dos arquivos.

Ao final do processo foram geradas planilhas nos formatos *.xls* e *.csv*.

	A	B	C	D	E	F	G	H	I	J
1	data	Ativo Total	Ativo Circulante	Caixa e Equivalentes de Caixa	Aplicações Financeiras	Contas a Receber	Estoques	Ativos Biológicos	Tributos a Recuperar	Despesas Antecipada
2	31/12/2016	0.36824303407	0.447613821348	0.236130129394	0.0662540071414	0.920886067481	0.475029133414	0	1.0	0.273305315526
3	30/11/2016	0.36824303407	0.447613821348	0.236130129394	0.0662540071414	0.920886067481	0.475029133414	0	1.0	0.273305315526
4	31/10/2016	0.36824303407	0.447613821348	0.236130129394	0.0662540071414	0.920886067481	0.475029133414	0	1.0	0.273305315526
5	30/09/2016	0.182093501672	0.126127629928	0.17445718176	0.0597916651277	0.47358856782	0.445227221246	0	0.443298175427	0.0962690453594
6	31/08/2016	0.182093501672	0.126127629928	0.17445718176	0.0597916651277	0.47358856782	0.445227221246	0	0.443298175427	0.0962690453594
7	31/07/2016	0.182093501672	0.126127629928	0.17445718176	0.0597916651277	0.47358856782	0.445227221246	0	0.443298175427	0.0962690453594
8	30/06/2016	0.0907967985479	0.0116252046442	0.0	0.0502849912558	0.203647226114	0.552022260613	0	0.460121459041	0.139398026226
9	31/05/2016	0.0907967985479	0.0116252046442	0.0	0.0502849912558	0.203647226114	0.552022260613	0	0.460121459041	0.139398026226
10	30/04/2016	0.0907967985479	0.0116252046442	0.0	0.0502849912558	0.203647226114	0.552022260613	0	0.460121459041	0.139398026226
11	31/03/2016	0.354870266208	0.103211587297	0.0517026196351	0.0	0.03090717986	1.0	0	0.469692305323	0.291545657336
12	29/02/2016	0.354870266208	0.103211587297	0.0517026196351	0.0	0.03090717986	1.0	0	0.469692305323	0.291545657336
13	31/01/2016	0.354870266208	0.103211587297	0.0517026196351	0.0	0.03090717986	1.0	0	0.469692305323	0.291545657336
14	31/12/2015	0.936313129185	1.0	1.0	0.0131545509473	1.0	0.687209733996	0	0.498537379063	1.0
15	30/11/2015	0.936313129185	1.0	1.0	0.0131545509473	1.0	0.687209733996	0	0.498537379063	1.0
16	31/10/2015	0.936313129185	1.0	1.0	0.0131545509473	1.0	0.687209733996	0	0.498537379063	1.0
17	30/09/2015	1.0	0.675835553783	0.566730562115	0.593252825306	0.582209724115	0.958607458608	0	0.0	0.611655463194
18	31/08/2015	1.0	0.675835553783	0.566730562115	0.593252825306	0.582209724115	0.958607458608	0	0.0	0.611655463194
19	31/07/2015	1.0	0.675835553783	0.566730562115	0.593252825306	0.582209724115	0.958607458608	0	0.0	0.611655463194
20	30/06/2015	0.399129884021	0.188258881143	0.231510975253	0.675801099062	0.0	0.674685819556	0	0.0486143557239	0.147651906116
21	31/05/2015	0.399129884021	0.188258881143	0.231510975253	0.675801099062	0.0	0.674685819556	0	0.0486143557239	0.147651906116
22	30/04/2015	0.399129884021	0.188258881143	0.231510975253	0.675801099062	0.0	0.674685819556	0	0.0486143557239	0.147651906116
23	31/03/2015	0.518226161815	0.316051204757	0.261554151734	1.0	0.104489510345	0.991454150034	0	0.0455706057638	0.372021472367
24	28/02/2015	0.518226161815	0.316051204757	0.261554151734	1.0	0.104489510345	0.991454150034	0	0.0455706057638	0.372021472367
25	31/01/2015	0.518226161815	0.316051204757	0.261554151734	1.0	0.104489510345	0.991454150034	0	0.0455706057638	0.372021472367
26	31/12/2014	0.441068103335	0.50529267558	0.650083773242	0.638003131508	0.286463763415	0.261555433582	0	0.109565904036	0.626124823215

Figura 3.3: ABEV3 Normalizada numérica deslocada 6 meses

	A	B	C	D	E	F	G	H	I	J
1	data	Ativo Total	Ativo Circulante	Caixa e Equivalentes de Caixa	Aplicações Financeiras	Contas a Receber	Estoques	Ativos Biológicos	Tributos a Recuperar	Despesas Antecipada
2	31/12/2016	-0.263513931861	-0.104772357305	-0.527739741212	-0.867491985717	0.841772134963	-0.0499417331711	-1	1.0	-0.453389368948
3	30/11/2016	-0.263513931861	-0.104772357305	-0.527739741212	-0.867491985717	0.841772134963	-0.0499417331711	-1	1.0	-0.453389368948
4	31/10/2016	-0.263513931861	-0.104772357305	-0.527739741212	-0.867491985717	0.841772134963	-0.0499417331711	-1	1.0	-0.453389368948
5	30/09/2016	-0.635812996656	-0.747744740144	-0.651085636481	-0.880416669745	-0.05282286436	-0.109545557509	-1	-0.113403649147	-0.807461909281
6	31/08/2016	-0.635812996656	-0.747744740144	-0.651085636481	-0.880416669745	-0.05282286436	-0.109545557509	-1	-0.113403649147	-0.807461909281
7	31/07/2016	-0.635812996656	-0.747744740144	-0.651085636481	-0.880416669745	-0.05282286436	-0.109545557509	-1	-0.113403649147	-0.807461909281
8	30/06/2016	-0.818406402904	-0.976749590712	-1.0	-0.899430017488	-0.592705547771	0.104044521225	-1	-0.0797570819176	-0.721203947549
9	31/05/2016	-0.818406402904	-0.976749590712	-1.0	-0.899430017488	-0.592705547771	0.104044521225	-1	-0.0797570819176	-0.721203947549
10	30/04/2016	-0.818406402904	-0.976749590712	-1.0	-0.899430017488	-0.592705547771	0.104044521225	-1	-0.0797570819176	-0.721203947549
11	31/03/2016	-0.290259467584	-0.793576825405	-0.89659476073	-1.0	-0.93818564028	1.0	-1	-0.0606153893549	-0.416908685329
12	29/02/2016	-0.290259467584	-0.793576825405	-0.89659476073	-1.0	-0.93818564028	1.0	-1	-0.0606153893549	-0.416908685329
13	31/01/2016	-0.290259467584	-0.793576825405	-0.89659476073	-1.0	-0.93818564028	1.0	-1	-0.0606153893549	-0.416908685329
14	31/12/2015	0.872626258371	1.0	1.0	-0.973690898105	1.0	0.374419467992	-1	-0.00292524187319	1.0
15	30/11/2015	0.872626258371	1.0	1.0	-0.973690898105	1.0	0.374419467992	-1	-0.00292524187319	1.0
16	31/10/2015	0.872626258371	1.0	1.0	-0.973690898105	1.0	0.374419467992	-1	-0.00292524187319	1.0
17	30/09/2015	1.0	0.351671107565	0.133461124231	0.186505650613	0.16441944823	0.917214917216	-1	-1.0	0.223310926389
18	31/08/2015	1.0	0.351671107565	0.133461124231	0.186505650613	0.16441944823	0.917214917216	-1	-1.0	0.223310926389
19	31/07/2015	1.0	0.351671107565	0.133461124231	0.186505650613	0.16441944823	0.917214917216	-1	-1.0	0.223310926389
20	30/06/2015	-0.201740231957	-0.623482237714	-0.536978049494	0.351602198124	-1.0	0.349371639112	-1	-0.902771288552	-0.704696187769
21	31/05/2015	-0.201740231957	-0.623482237714	-0.536978049494	0.351602198124	-1.0	0.349371639112	-1	-0.902771288552	-0.704696187769
22	30/04/2015	-0.201740231957	-0.623482237714	-0.536978049494	0.351602198124	-1.0	0.349371639112	-1	-0.902771288552	-0.704696187769
23	31/03/2015	0.0364523236298	-0.367897590486	-0.476891696532	1.0	-0.79102097931	0.982908300068	-1	-0.908858788472	-0.255957055265
24	28/02/2015	0.0364523236298	-0.367897590486	-0.476891696532	1.0	-0.79102097931	0.982908300068	-1	-0.908858788472	-0.255957055265
25	31/01/2015	0.0364523236298	-0.367897590486	-0.476891696532	1.0	-0.79102097931	0.982908300068	-1	-0.908858788472	-0.255957055265
26	31/12/2014	-0.11786379333	0.0105853511591	0.300167546485	0.276006263016	-0.427072473171	-0.476889132837	-1	-0.780868191928	0.252249646431

Figura 3.4: ABEV3 Normalizada percentual de aumento deslocada 6 meses

As alterações realizadas nos arquivos (explicitadas acima) foram realizadas pelos seguintes trechos de código:

Combinação das planilhas 'balanço', 'demonstrativos', 'cotações' , 'indicadores fundamentalistas' e 'indicadores socioeconômico'

Utilizando o *script* em anexo I.11 foi adicionado para cada empresa uma matriz os dados do balanço patrimônial, do demonstrativo financeiro, esses dados são fornecidos trimestralmente, a adição foi feita mensalmente. Para isso triplicou-se os dados deflacionadaos de cada trimestre para os meses contidos nesse trimestre. Há também a adição das cotação mensal deflacionada de cada empresa. O programa também calcula os indicadores fundamentalistas com base nos dados listados anteriormente e os adiciona a mesma. Além disso são adicionados alguns indicadores socioeconomicos, ibovespa e juro real deflacionados. Ao final do processamento é gerada uma matriz onde as colunas são os indicadores e as linhas seus valores por data. A partir desta matriz tem-se a saída de uma planilha de extensão ".xlsx"e um arquivo de extensão ".csv".

3.3.2 Deflação de Balanço, Demonstrativo e Cotações

Os dados dos balanço patrimonial, do demonstrativo, bem como os valores das cotações foram deflacionados usando o índice IGPM. Tal medida foi necessária para que obtivéssemos dados livres da influência da inflação, de forma que a análise pudesse ser realizada da melhor forma possível. O problema com os dados inflacionados era o seguinte: havia casos em que determinada empresa apresentava crescimento, mas este era apenas reflexo da inflação do período. Como nosso estudo tinha por objetivo a descoberta de padrões de acordo com o comportamento das empresas a partir dos indicadores citados anteriormente, foi necessária a eliminação de agentes externos que não contribuiriam com o objetivo proposto, permitindo, assim, uma avalição mais realista das ações.

A deflação foi realizada tomando como base o mês de dezembro do ano de 2012. Com essa data fixada, foi obtido o IGP-M mês a mês e então calculou-se a inflação acumulada de cada mês em relação à data base como mostrado na Figura 3.5.

	A	B
	Mês/Ano	Deflator Base Dez 2012
2	jun/17	1,2708823
3	mar/17	1,3058294
4	dez/16	1,2963584
5	set/16	1,2877223
6	jun/16	1,2809215
7	mar/16	1,2452792
8	dez/15	1,2093944
9	set/15	1,1634881
10	jun/15	1,1414449
11	mar/15	1,1161592
12	dez/14	1,094036
13	set/14	1,0737363
14	jun/14	1,0810888
15	mar/14	1,0821257
16	dez/13	1,0552566
17	set/13	1,0370113
18	jun/13	1,0175103
19	mar/13	1,0084231
20	dez/12	1

Figura 3.5: Índice de Inflação Acumulada Trimestral em relação à Dez/2012

3.3.3 Eliminação de colunas nulas

Foi realizada uma vistoria nas planilhas do grupo de dados. Caso uma coluna estivesse com todos os seus valores zerados em alguma das planilhas do conjunto, essa coluna seria eliminada de todas as planilhas (até mesmo nas planilhas em que tal coluna possuísse valor diferente de zero). Tal medida foi tomada para que os resultados obtidos não sofressem influência de dados que seriam pouco relevantes para comparações entre as ações.

3.3.4 Produção de planilhas com deslocamento de dados

Determinadas partes da análise dos dados (que serão explanadas mais à frente) requisitaram um modelo diferente de planilha de dados. Tal modelo se baseava no deslocamento dos dados. Isto ocorria da seguinte forma: para determinada empresa, todas as colunas eram deixadas como estavam e apenas a coluna que seria usada como classe seria deslocada (em nosso caso, a coluna *cotação*). Tal alteração baseava-se no princípio de que as informacões presentes no arquivo seriam utilizadas para prever os dados do mês seguinte. Por exemplo, o deslocamento de 1 mês funcionaria da seguinte maneira: se o mês é janeiro, a cotação deve ser de fevereiro; no mês de fevereiro, a cotação seria de março; e assim sucessivamente até que acabassem os dados de cotação e sobrassem os restantes; então as linhas no final do arquivo que não tivessem dados deslocados deveriam ser eliminadas.

Isso foi feito de forma que tivéssemos cinco grupos de dados: com o deslocamento de 1 mês de dados (último mês disponível), com o deslocamento de 3 meses de dados, com o deslocamento de 6 meses de dados, com o deslocamento de 9 meses de dados e com o deslocamento de 12 meses de dados).

Os arquivos foram produzidos em formato numérico e, após isso, foram normalizados, utilizando a seguinte formula:

$$Valor\ Normalizado = \frac{Valor\ a\ ser\ Normalizado\ -\ Valor\ Mínimo\ Coluna}{Valor\ Máximo\ Coluna\ -\ Valor\ Mínimo\ Coluna} \quad (3.2)$$

A razão para a normalização vem do fato de que o crescimento de determinadas empresas no conjunto de dados foi muito mais expressivo do que o crescimento das demais empresas do grupo no mesmo período.

3.3.5 Planilha trimestral - Empresa por Indicadores

Um outro passo no processo de preparação dos dados foi a produção de uma planilha trimestral (diferente da citada anteriormente) onde cada linha correspondesse a uma empresa e cada coluna fosse um dos indicadores fundamentalistas. Como os dados dos balanços são apresentados de forma trimestral, tal característica foi utilizada para a produção da planilha em questão. Foram feitas duas versões: uma com os indicadores em seu formato original (sem intervenções) e uma fazendo uso da normalização.

Normalização

A razão pela qual a normalização foi utilizada é o fato do conjunto de empresas usado apresentar muitas diferenças nos valores de seus dados do balanço patrimônial, demonstrativo financeiro, cotações e indicadores fundamentalistas.

Desta forma, foi realizada a normalização dos dados da seguinte maneira: A normalização foi feita por colunas. Normalizou-se as colunas até a coluna de índice "CA"(incluindo-a no processo). Pegou-se o maior valor em módulo (presente na coluna) e utilizou-se dele como divisor para os demais valores (de forma que o maior numero ficaria sendo 1, se só existissem números positvos). Se o maior número em módulo fosse um número negativo, ele se tornaria -1 após a normalização.

3.3.6 Divisão dos arquivos por grupos de ações

Parte da análise seria realizada sobre o conjunto completo de dados, mas para algumas fases específicas foram utilizados os dados dos balanços da empresas agrupados por setores de atuação. Tais subgrupos foram montados de acordo com a divisão de empresas presente no site Fundamentus [3] e escolha dos grupos foi feita tomando como base suas respectivas relevâncias para o cenário econômico brasileiro. Os setores avaliados foram os seguintes: Carnes e derivados; Petróleo, Gás e Biocombustíveis; Tecidos e Vestuário.

3.4 *Modeling* (Modelagem) e *Evaluation* (Avaliação)

O processo de *Modeling*, também conhecido como *Modelagem* é onde, de fato, a mineração de dados ocorreu. Os dados que foram preparados nas etapas anteriores puderam ser combinados e expostos a algoritmos específicos para revelar padrões. Os métodos utilizados foram os seguintes:

- Aplicação do algoritmo *lazy.IBk* para a classificação dos dados (sem deslocamento dos dados);
- Aplicação do algoritmo *lazy.IBk* para a classificação dos dados (com deslocamento dos dados);
- Comparação dos indicadores presentes nas planilhas e geração de gráficos *Scatter Plot*;
- Geração de gráficos em relação ao tempo (utilizando-se dos dados do item anterior);
- Comparações dois a dois de todos os dados das planilhas de dados das empresas;
- Comparações dos dados dos grupos de empresas e geração de gráficos;
- Comparações entre os dados das cotações deflacionadas e os dados Ibovespa deflacionados e geração de gráficos;
- Comparação dos dados das cotações das empresas e da taxa de juros reais e geração de gráficos;
- Comparação dos dados das cotações das empresas com as próprias cotações das empresas e geração de gráficos;
- Comparação dos indicadores presentes nas planilhas e os dados Ibovespa e geração de gráficos;
- Comparação dos indicadores presentes nas planilhas e da taxa de juros reais e geração de gráficos;
- Comparação dos dados das cotações diárias das empresas com as próprias cotações diárias e geração de gráficos;
- Clusterização dos dados e geração de dendogramas para visualização de resultados;

Já o processo de *Evaluation*, também conhecido como *Avaliação* é onde os resultados obtidos são avaliados de forma a se verificar se são tecnicamente corretos e eficazes de acordo com os critérios definidos anteriormente.

Os processos de *Modeling* e *Evaluation* estão descritos a seguir.

3.4.1 Aplicação do algoritmo *lazy.IBk* para a classificação dos dados

A abordagem utilizada no processo de classificação dos dados foi o uso do software *Weka*. Nele, foi feito o uso do algoritmo *k-Nearest-Neighbor* que, no *Weka*, recebe o nome *lazy.IBk* (o IB significa *Instance Based* (Baseado em Instâncias) e o k é o número de vizinhos que o algoritmo examinará). Tal algoritmo faz uso de um conjunto de dados de treinamento para, assim, aprender a classificar um novo registro. Um dos motivos pelos quais esse algoritmo foi escolhido é o fato de que ele trabalha com informações numéricas. Tendo em vista que os dados de nosso conjunto de estudo são, em sua grande maioria, numéricos, o algoritmo pareceu ser a melhor escolha. A configuração utilizada para o algoritmo no software *Weka* se encontra na Figura 3.6.

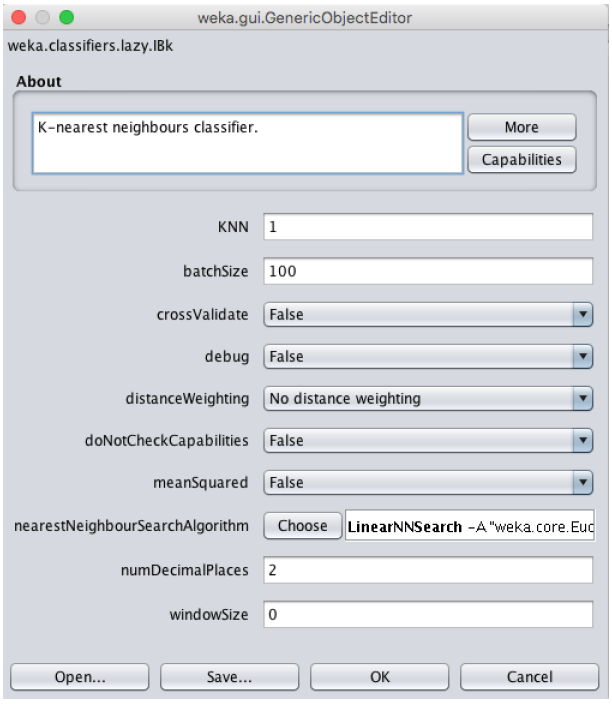


Figura 3.6: Configuração do algoritmo *lazy.IBk* no software *Weka*

A abordagem foi a seguinte: tomava-se um conjunto de dados de determinada empresa como base para treinamento (no nosso caso, a empresa JBS para o grupo de carnes e derivados, a Petrobrás para o grupo de petróleo, gás e biocombustíveis e a Arezzo para o grupo de tecidos e vestuário) e aplicava-se o modelo sobre os demais conjuntos de dados, obtendo, assim, o resultado. Usamos como classe para a classificação o valor das cotações das empresas. Tal método apresentou resultados positivos no que diz respeito aos valores de correlação (tomamos como resultados positivos aqueles cujos valores estavam acima de 0.5 ou abaixo de -0.5).

Havia ainda a divisão em dois grupos de dados: *sem deslocamento* e *com deslocamento*. O grupo *sem deslocamento* apresentava todos os dados conforme haviam sido formatados anteriormente, sem a exclusão de nenhum. Já o grupo *com deslocamento* apresentava menos dados, conforme foi explicado anteriormente, no processo de formatação do conjunto de dados deslocados. Desta forma, o algoritmo treinaria com os dados disponíveis e tentaria prever os dados futuros (que haviam sido retirados da planilha). Isso foi feito de forma que tivéssemos cinco grupos de dados: com o deslocamento de 1 mês de dados (último mês disponível), com o deslocamento de 3 meses de dados, com o deslocamento de 6 meses de dados, com o deslocamento de 9 meses de dados e com o deslocamento de 12 meses de dados). A ideia foi avaliar o poder preditivo dos algoritmos com os diferentes deslocamentos.

Os resultados mais satisfatórios entre os que estavam na zona dos resultados positivos serão apresentados a seguir.

É importante salientar que ainda que tenhamos encontrado resultados interessantes no que diz respeito aos valores de correlação entre determinadas empresas, tal fato não implica em causa ou efeito, ou seja, não é possível chegar a conclusões mais robustas baseando-nos apenas nas correlações encontradas. Entretanto, tais dados podem ser usados para auxiliar na tentativa de previsão do comportamento de forma a auxiliar na seleção de empresas para estudos futuros.

Sem deslocamento de dados

Dentre os resultados obtidos, mostramos abaixo alguns dos que se destacaram. Os exemplos são os seguintes: resultados entre as empresa JBS (empresa utilizada como base) e BRF (Figuras 3.7 e 3.8); resultados entre as empresas Petrobrás (empresa utilizada como base) e Enauta (Figuras 3.9 e 3.10); resultados entre as empresas Arezzo (empresa utilizada como base) e Guararapes Confecções (Figuras 3.11 e 3.12).

As figuras relativas aos resultados aparecem em pares. As que aparecem em primeiro lugar mostram os resultados obtidos no *Weka*. Nestas é possível observar um valor de correlação obtido no processo; tal valor é relativo à relação entre o conjunto de dados e o modelo utilizados. As figuras que aparecem em segunda posição mostram o resultado obtido no *Weka* em forma de gráfico. Tais gráficos têm em seu *eixo X* o valor real das cotações e em seu *eixo Y* a previsão realizada pelo algoritmo; quanto mais próximas as previsões dos valores reais, mais a regressão do gráfico se aproxima de uma reta ascendente.

```
=== Run information ===

Scheme:       weka.classifiers.lazy.IBk -K 3 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""
Relation:     jbss3_norm
Instances:    57
Attributes:   121
              [list of attributes omitted]
Test mode:    user supplied test set:  size unknown (reading incrementally)

=== Classifier model (full training set) ===

IB1 instance-based classifier
using 3 nearest neighbour(s) for classification


Time taken to build model: 0 seconds

=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.04 seconds

=== Summary ===

Correlation coefficient                  0.5266
Mean absolute error                      0.2313
Root mean squared error                  0.28
Relative absolute error                 86.0857 %
Root relative squared error             88.8695 %
Total Number of Instances               42
Ignored Class Unknown Instances                 15
```

Figura 3.7: Resultado do algoritmo *lazy.IBk* para as empresas JBS e BRF (Sem deslocamento de dados)

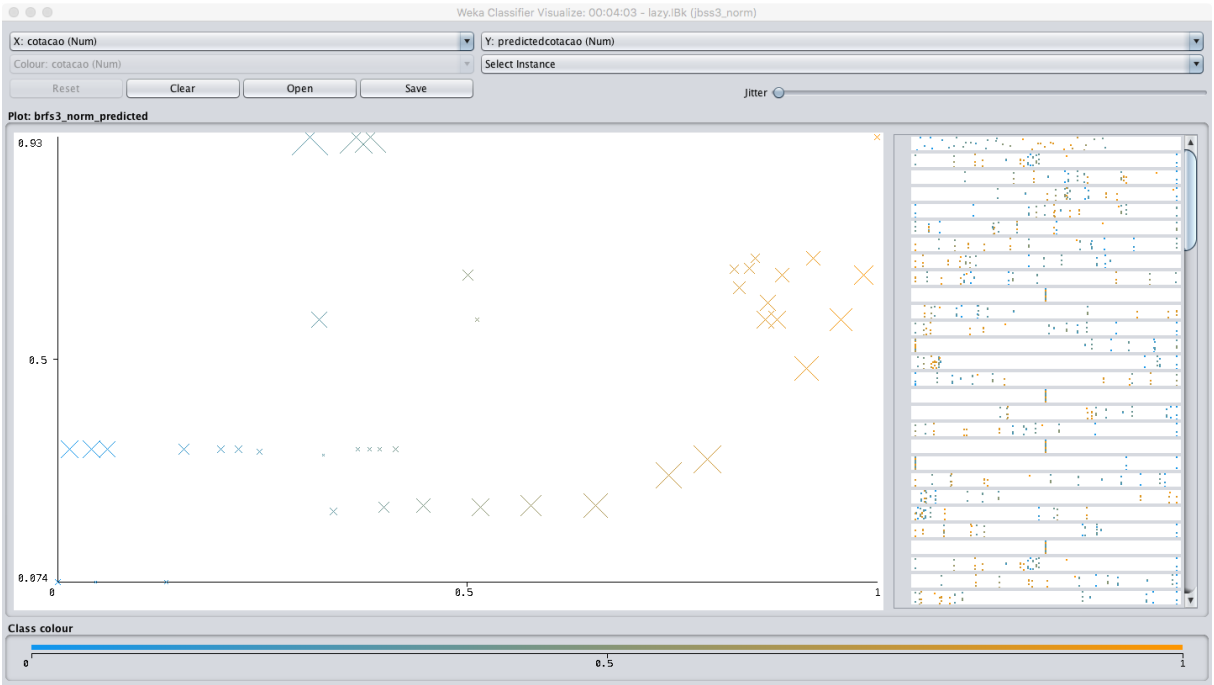


Figura 3.8: Resultado do algoritmo *lazy.IBk* para as empresas JBS e BRF (Sem deslocamento de dados) - Gráfico

```
=== Run information ===

Scheme:       weka.classifiers.lazy.IBk -K 3 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""
Relation:     petr3_norm
Instances:    57
Attributes:   121
              [list of attributes omitted]
Test mode:    user supplied test set:  size unknown (reading incrementally)

=== Classifier model (full training set) ===

IB1 instance-based classifier
using 3 nearest neighbour(s) for classification


Time taken to build model: 0 seconds

=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.02 seconds

=== Summary ===

Correlation coefficient                  0.7004
Mean absolute error                      0.1893
Root mean squared error                  0.2334
Relative absolute error                 79.8669 %
Root relative squared error             78.5259 %
Total Number of Instances               42
Ignored Class Unknown Instances                 15
```

Figura 3.9: Resultado do algoritmo *lazy.IBk* para as empresas Petrobrás e Enauta (Sem deslocamento de dados)

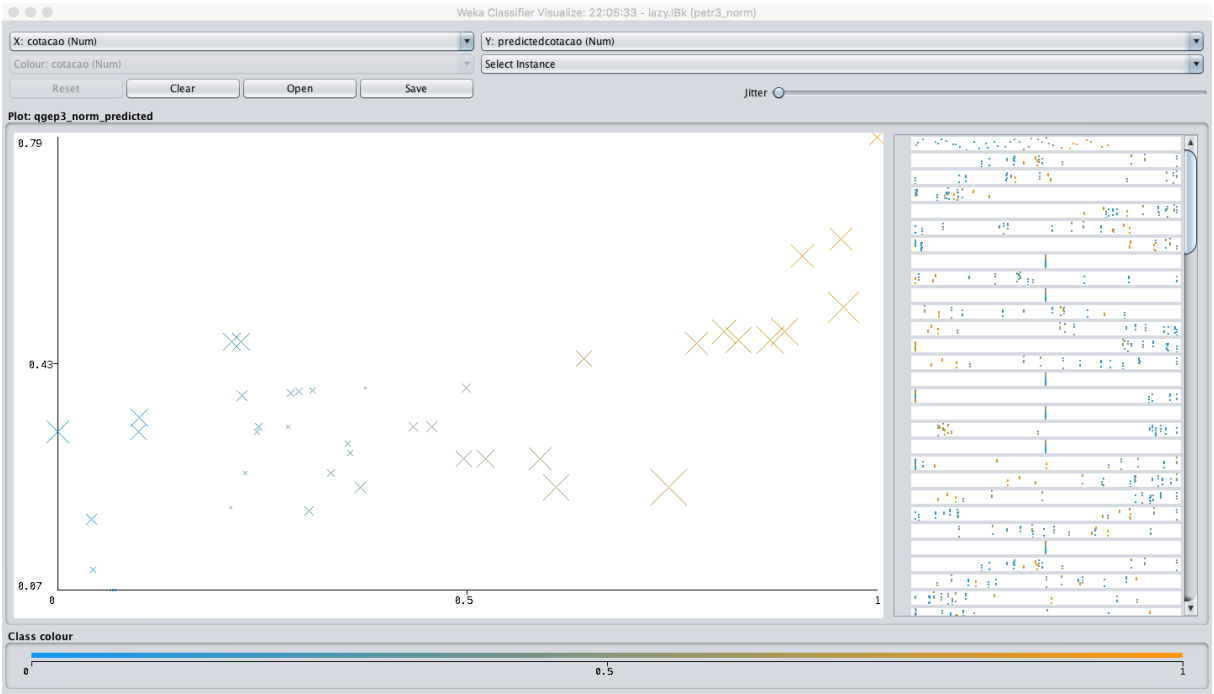


Figura 3.10: Resultado do algoritmo *lazy.IBk* para as empresas Petrobrás e Enauta (Sem deslocamento de dados) - Gráfico

```
=== Run information ===

Scheme:       weka.classifiers.lazy.IBk -K 3 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""
Relation:     arzz3_norm
Instances:    57
Attributes:   121
              [list of attributes omitted]
Test mode:    user supplied test set:  size unknown (reading incrementally)

=== Classifier model (full training set) ===

IB1 instance-based classifier
using 3 nearest neighbour(s) for classification


Time taken to build model: 0 seconds

=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.01 seconds

=== Summary ===

Correlation coefficient                  0.9129
Mean absolute error                      0.1357
Root mean squared error                  0.1766
Relative absolute error                 51.1527 %
Root relative squared error             58.8213 %
Total Number of Instances               42
Ignored Class Unknown Instances                 15
```

Figura 3.11: Resultado do algoritmo *lazy.IBk* para as empresas Arezzo e Guararapes Confecções (Sem deslocamento de dados)

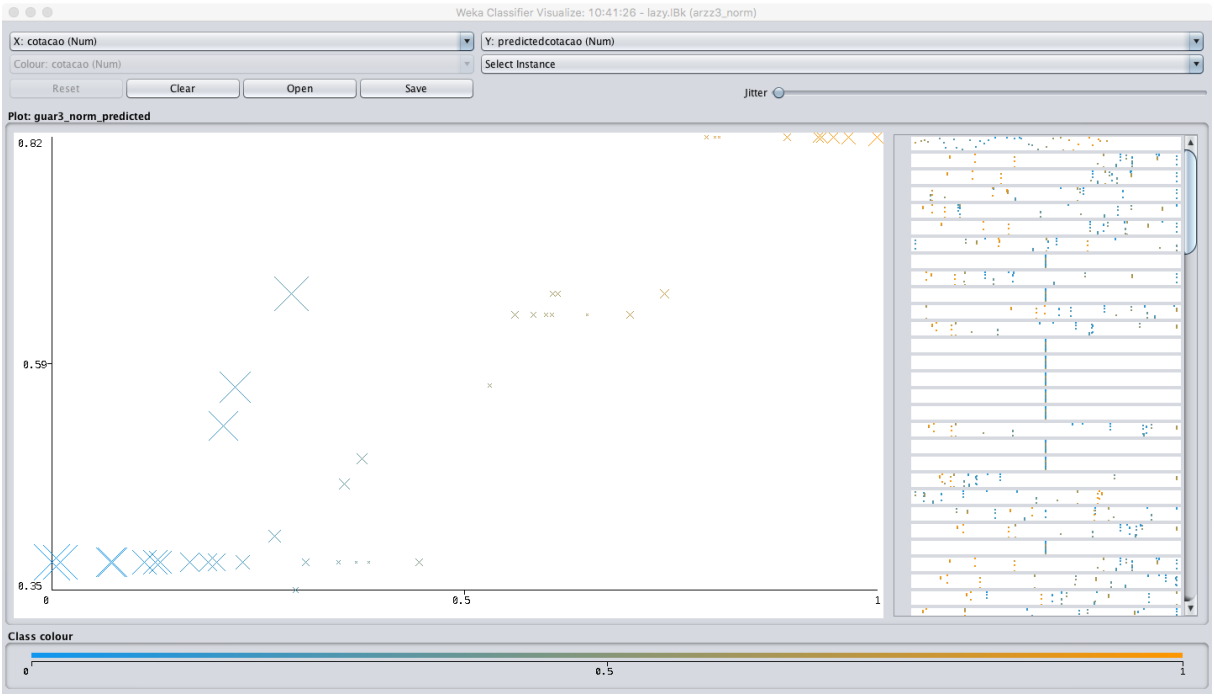


Figura 3.12: Resultado do algoritmo *lazy.IBk* para as empresas Arezzo e Guararapes Confecções (Sem deslocamento de dados) - Gráfico

Com deslocamento de dados

Dentre os resultados obtidos, mostramos abaixo alguns dos que se destacaram. Os exemplos são os seguintes: resultados entre a empresa JBS (empresa utilizada como base) e a empresa BRF (deslocamento de 1 mês) (Figuras 3.13 3.14); resultados entre as empre-

sas Petrobrás (empresa utilizada como base) e Enauta (deslocamento de 3 meses) (Figuras 3.15 e 3.16); resultados entre as empresas Arezzo (empresa utilizada como base) e Grazziotin (deslocamento de 6 meses) (Figuras 3.17 e 3.18); resultados entre a empresa JBS (empresa utilizada como base) e a empresa BRF (deslocamento de 9 meses) (Figuras 3.19 e 3.20); resultados entre as empresas Arezzo (empresa utilizada como base) e Grazziotin (deslocamento de 12 meses) (Figuras 3.21 e 3.22).

Assim como nos exemplos sem deslocamento, aqui as figuras relativas aos resultados também aparecem em pares. As que aparecem em primeiro lugar mostram os resultados obtidos no *Weka*. Nestas é possível observar um valor de correlação obtido no processo; tal valor é relativo à relação entre o conjunto de dados e o modelo utilizados. As figuras que aparecem em segunda posição mostram o resultado obtido no *Weka* em forma de gráfico. Tais gráficos têm em seu *eixo X* o valor real das cotações e em seu *eixo Y* a previsão realizada pelo algoritmo; quanto mais próximas as previsões dos valores reais, mais a regressão do gráfico se aproxima de uma reta ascendente.

```
1  === Run information ===
2
3  Scheme:       weka.classifiers.lazy.IBk -K 3 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""
4  Relation:     jbss3_norm_deslocada1
5  Instances:    41
6  Attributes:   121
7                [list of attributes omitted]
8  Test mode:    user supplied test set:  size unknown (reading incrementally)
9
10 === Classifier model (full training set) ===
11
12 IB1 instance-based classifier
13 using 3 nearest neighbour(s) for classification
14
15
16 Time taken to build model: 0 seconds
17
18 === Evaluation on test set ===
19
20 Time taken to test model on supplied test set: 0.01 seconds
21
22 === Summary ===
23
24 Correlation coefficient                  0.5982
25 Mean absolute error                      0.2141
26 Root mean squared error                  0.2664
27 Relative absolute error                 78.5536 %
28 Root relative squared error             83.991  %
29 Total Number of Instances               41
```

Figura 3.13: Resultado do algoritmo *lazy.IBk* para as empresas JBS e BRF (Com deslocamento de dados de 1 mês) - Gráfico

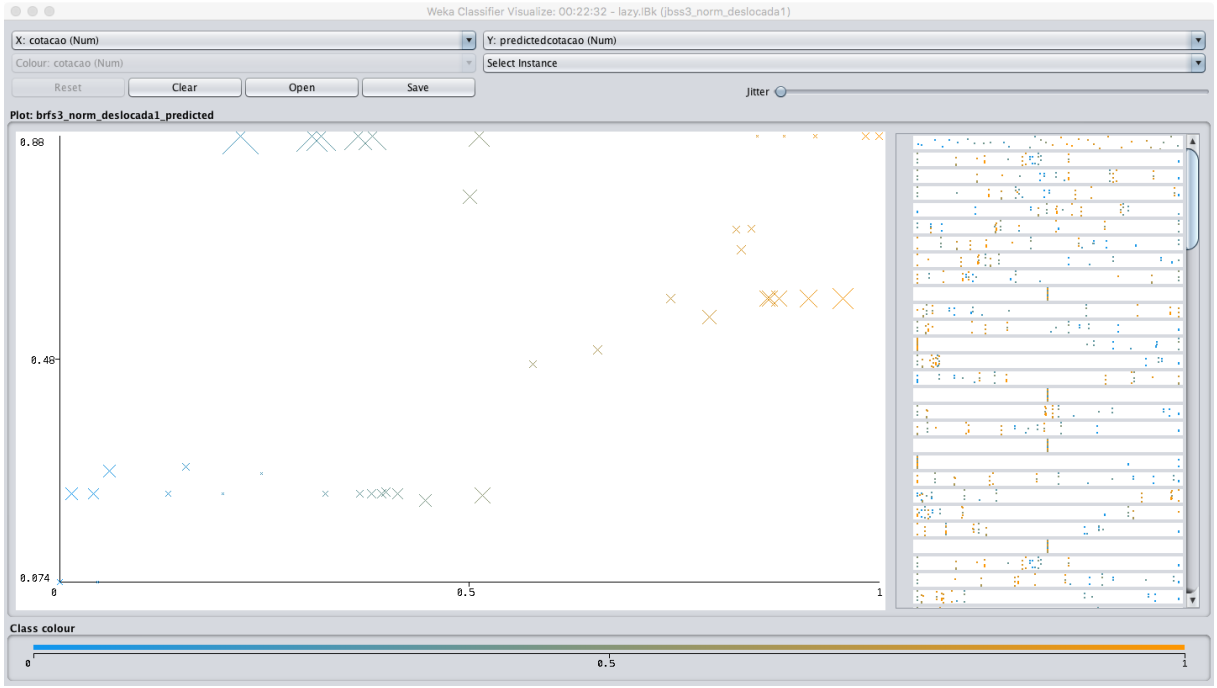


Figura 3.14: Resultado do algoritmo *lazy.IBk* para as empresas JBS e BRF (Com deslocamento de dados de 1 mês - Gráfico

```
1   === Run information ===
2
3   Scheme:       weka.classifiers.lazy.IBk -K 3 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""
4   Relation:     petr3_norm_deslocada3
5   Instances:    39
6   Attributes:   121
7                 [list of attributes omitted]
8   Test mode:    user supplied test set:  size unknown (reading incrementally)
9
10  === Classifier model (full training set) ===
11
12  IB1 instance-based classifier
13  using 3 nearest neighbour(s) for classification
14
15
16  Time taken to build model: 0 seconds
17
18  === Evaluation on test set ===
19
20  Time taken to test model on supplied test set: 0.01 seconds
21
22  === Summary ===
23
24  Correlation coefficient                  0.5047
25  Mean absolute error                      0.2105
26  Root mean squared error                  0.2477
27  Relative absolute error                 95.6685 %
28  Root relative squared error             86.8034 %
29  Total Number of Instances               39
```

Figura 3.15: Resultado do algoritmo *lazy.IBk* para as empresas Petrobrás e Enauta (Com deslocamento de dados de 3 meses)

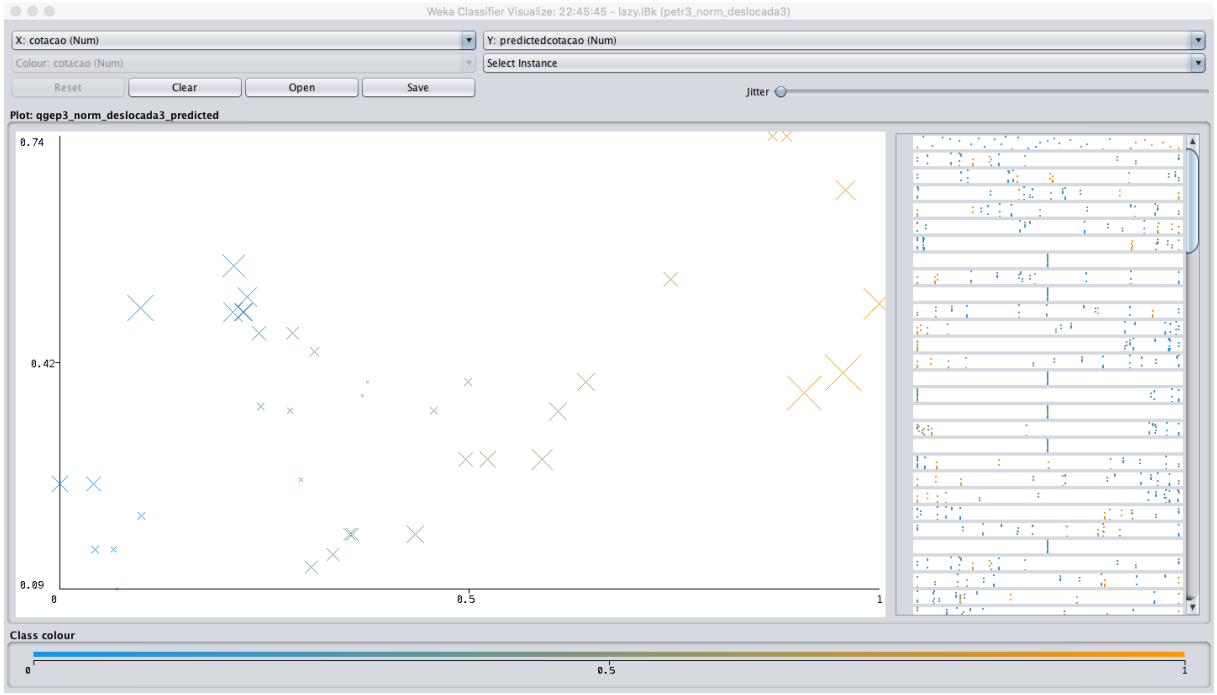


Figura 3.16: Resultado do algoritmo *lazy.IBk* para as empresas Petrobrás e Enauta (Com deslocamento de dados de 3 meses) - Gráfico

```
1   === Run information ===
2
3   Scheme:       weka.classifiers.lazy.IBk -K 3 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""
4   Relation:     arzz3_norm_deslocada6
5   Instances:    36
6   Attributes:   121
7                 [list of attributes omitted]
8   Test mode:    user supplied test set:  size unknown (reading incrementally)
9
10  === Classifier model (full training set) ===
11
12  IB1 instance-based classifier
13  using 3 nearest neighbour(s) for classification
14
15
16  Time taken to build model: 0 seconds
17
18  === Evaluation on test set ===
19
20  Time taken to test model on supplied test set: 0 seconds
21
22  === Summary ===
23
24  Correlation coefficient                  0.8862
25  Mean absolute error                      0.1079
26  Root mean squared error                  0.1266
27  Relative absolute error                 49.4442 %
28  Root relative squared error             48.7424 %
29  Total Number of Instances               36
```

Figura 3.17: Resultado do algoritmo *lazy.IBk* para as empresas Arezzo e Grazziotin (Com deslocamento de dados de 6 meses)

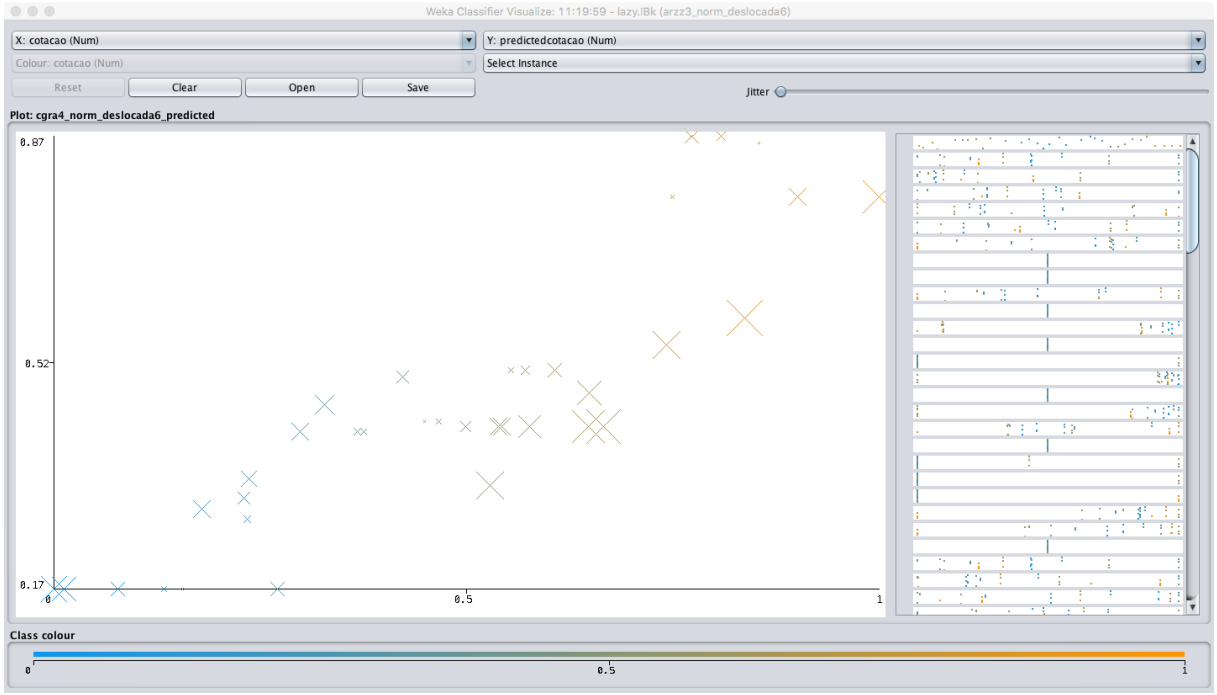


Figura 3.18: Resultado do algoritmo *lazy.IBk* para as empresas Arezzo e Grazziotin (Com deslocamento de dados de 6 meses) - Gráfico

```
1   === Run information ===
2
3   Scheme:       weka.classifiers.lazy.IBk -K 3 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""
4   Relation:     jbss3_norm_deslocada9
5   Instances:    33
6   Attributes:   121
7                 [list of attributes omitted]
8   Test mode:    user supplied test set:  size unknown (reading incrementally)
9
10  === Classifier model (full training set) ===
11
12  IB1 instance-based classifier
13  using 3 nearest neighbour(s) for classification
14
15
16  Time taken to build model: 0 seconds
17
18  === Evaluation on test set ===
19
20  Time taken to test model on supplied test set: 0 seconds
21
22  === Summary ===
23
24  Correlation coefficient                  0.8435
25  Mean absolute error                      0.1434
26  Root mean squared error                  0.1836
27  Relative absolute error                 48.0694 %
28  Root relative squared error             55.1951 %
29  Total Number of Instances               33
```

Figura 3.19: Resultado do algoritmo *lazy.IBk* para as empresas JBS e BRF (Com deslocamento de dados de 9 meses)

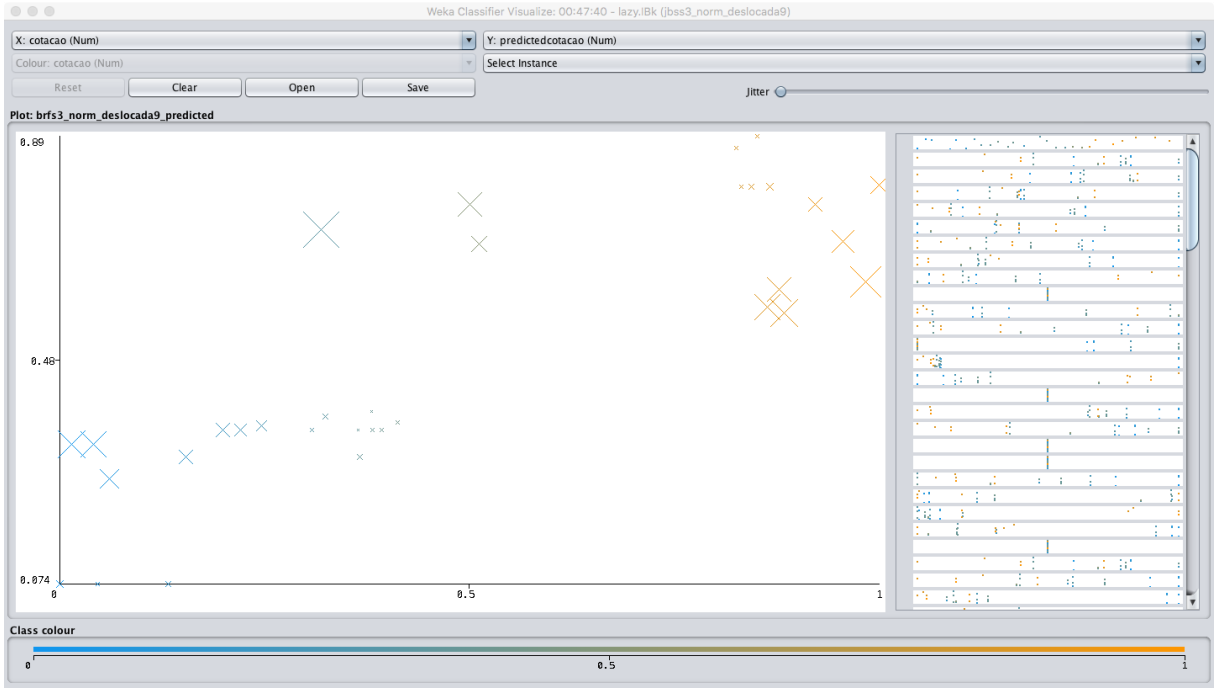


Figura 3.20: Resultado do algoritmo *lazy.IBk* para as empresas JBS e BRF (Com deslocamento de dados de 9 meses) - Gráfico

```
1   === Run information ===
2
3   Scheme:       weka.classifiers.lazy.IBk -K 3 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""
4   Relation:     arzz3_norm_deslocada12
5   Instances:    30
6   Attributes:   121
7                 [list of attributes omitted]
8   Test mode:    user supplied test set:  size unknown (reading incrementally)
9
10  === Classifier model (full training set) ===
11
12  IB1 instance-based classifier
13  using 3 nearest neighbour(s) for classification
14
15
16  Time taken to build model: 0 seconds
17
18  === Evaluation on test set ===
19
20  Time taken to test model on supplied test set: 0 seconds
21
22  === Summary ===
23
24  Correlation coefficient                  0.9195
25  Mean absolute error                      0.0796
26  Root mean squared error                  0.0968
27  Relative absolute error                 39.9568 %
28  Root relative squared error             39.7427 %
29  Total Number of Instances               30
```

Figura 3.21: Resultado do algoritmo *lazy.IBk* para as empresas Arezzo e Grazziotin (Com deslocamento de dados de 12 meses)

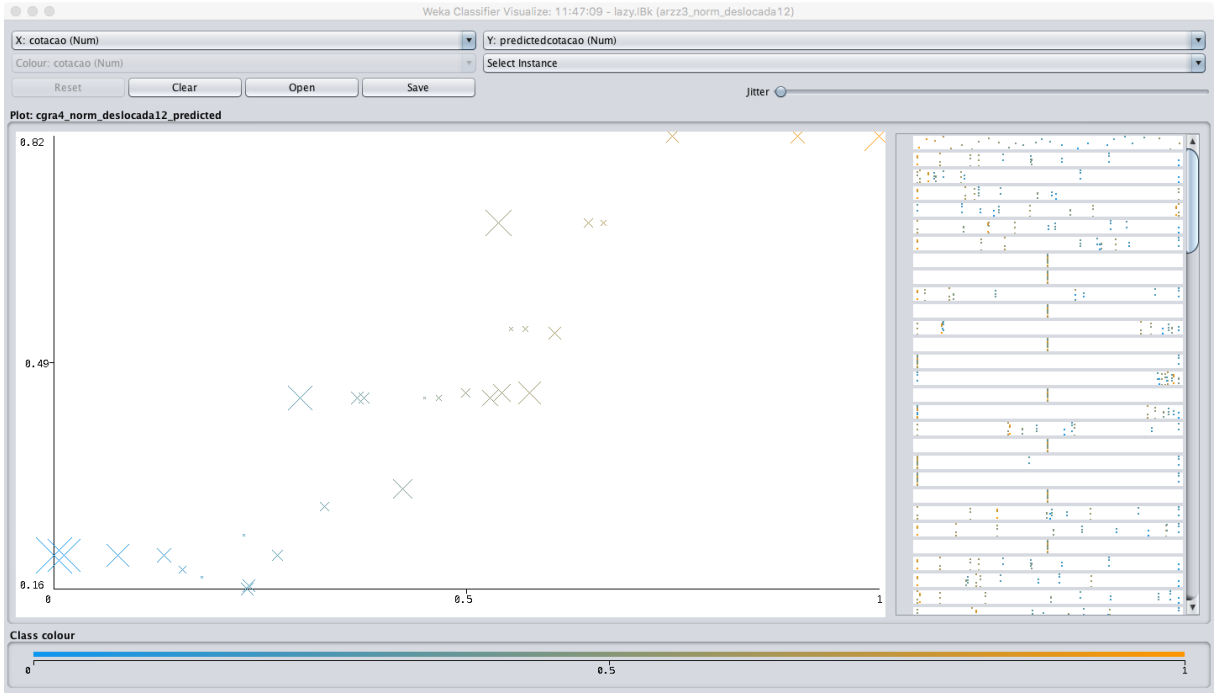


Figura 3.22: Resultado do algoritmo *lazy.IBk* para as empresas Arezzo e Grazziotin (Com deslocamento de dados de 12 meses) - Gráfico

3.4.2 Comparação dos indicadores presentes nas planilhas e geração de gráficos *Scatter Plot*

Utilizando o *script* I.7 em anexo escrito em *Python* foram geradas as correlações dois a dois de todos os dados das planilhas das empresas. Foram geradas 7260(sete mil duzentos e sessenta) correlações para cada empresa, das quais foram selecionadas as com os melhores resultados para análise.

Foi percorrida a matriz de correlação e foi feito um gráfico *Scatter Plot* com uma linha de regressão linear para todas as correlações maiores que 0.5 e menores de -0.5.

A Figura 3.23 é um exemplo de um dos gráficos gerados. Esta mostra uma forte correlação negativa de valor próximo de -1. Os indicadores retratados são o retorno do capital próprio (ROE) dos acionistas da Ambev e a expectativa do consumidor no Brasil.

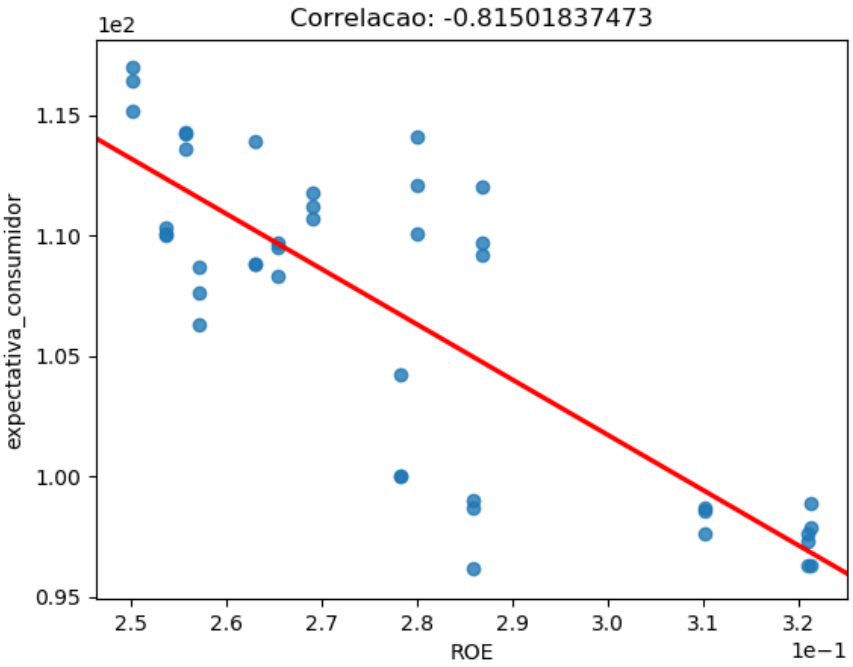


Figura 3.23: ABEV3 - Scatter Plot ROE x Expectava do consumidor

A Figura 3.24 é mais um exemplo de um dos gráficos gerados. Esta mostra uma forte correlação positiva próxima de 1. Os indicadores retratados são a receita financeira da Ambev e o resultado dela antes da tributação e divisão das participações aos acionistas.

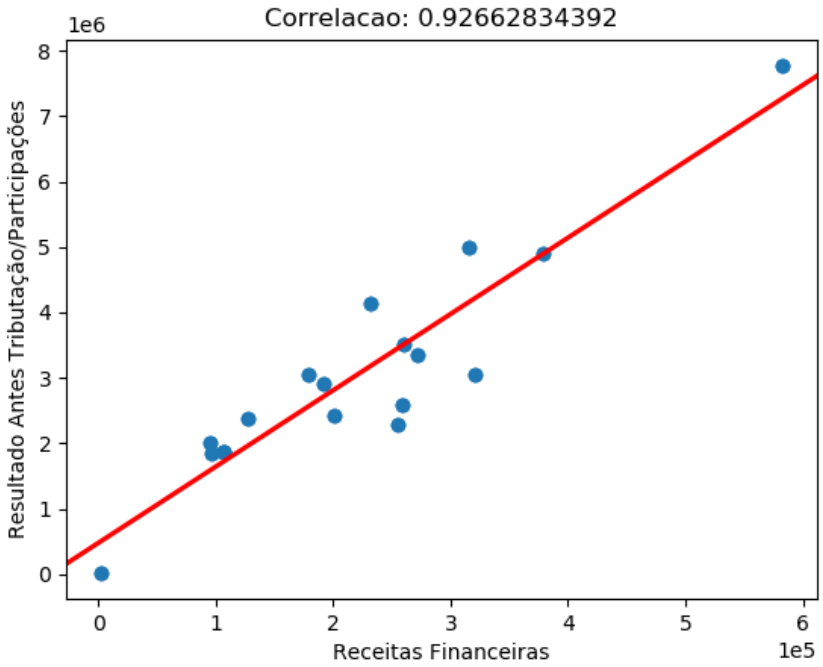


Figura 3.24: ABEV3 - Scatter Plot Receita Financeira x Resultado antes tributação participações

3.4.3 Geração de gráficos em relação ao tempo (utilizando-se dos dados do item anterior)

Selecionou-se os pares de indicadores que obtiveram uma correlação maior que 0.5 e menor -0.5. Com esses dados criou-se gráficos lineares dos valores em relação ao tempo.

A Figura 3.25 mostra que existe uma forte correlação entre os Intangíveis da Ambev e o Passivo Não Circulante.

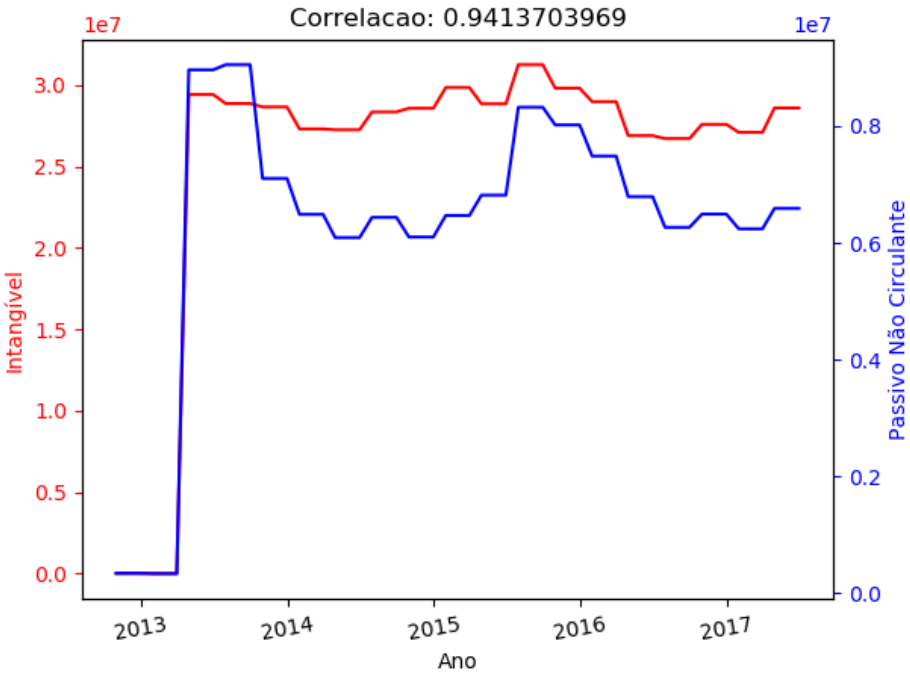


Figura 3.25: Intangível e Passivo Não Circulante da Ambev em relação ao tempo

3.4.4 Comparações dois a dois de todos os dados das planilhas de dados das empresas

Utilizando o *script* I.7 em anexo escrito em *Python* foram gerados as correlações duas a duas de todos os dados das planilhas das empresas. A Figura mostra uma parte do resultado obtido.

	Ativo Total	Ativo Circulante	e Equivalentes de	ções Finan	tas a Rece	Estoques	vos Biológ	itos a Recu	sas Anteci	Ativos Circ	ível a Lo	eiras Avali	Avaliadas	as a Rece	Estoques.1	os Biológic	utos Diferi	as Antecip	m Partes R	ivos Não C
Ativo Total	1	0,957693316	0,82373359	0,450075	0,941069	0,956474		0,543142	0,958409	0,54904	0,860401	-0,65915	0,696585	0,055061			0,765577	0,508081		0,942631
Ativo Circulante	0,957693316	1	0,925479179	0,338859	0,911859	0,898407		0,589164	0,955328	0,593593	0,800541	-0,63789	0,727448	0,054217			0,700062	0,575609		0,89211
Equivalentes d	0,82373359	0,925479179	1	0,310292	0,82272	0,686414		0,366726	0,908809	0,528923	0,65394	-0,53284	0,61767	-0,03714			0,518233	0,365452		0,813433
ícações Finance	0,45007508	0,338859222	0,310292047	1	0,355493	0,390088		-0,34641	0,416393	0,570062	0,536517	-0,24861	0,325818	-0,29236			0,397405	-0,00478		0,663717
ontas a Recebe	0,94106897	0,911859145	0,822719969	0,355493	1	0,841151		0,529016	0,921367	0,343785	0,707786	-0,63937	0,588139	-0,02066			0,608009	0,32319		0,849021
Estoques	0,956474022	0,898407425	0,686413711	0,390088	0,841151	1		0,627977	0,863621	0,577106	0,866518	-0,65048	0,693488	0,109637			0,810459	0,649438		0,88059
tivos Biológicos																				
butos a Recupe	0,543142168	0,58916442	0,366726472	-0,34641	0,529016	0,627977		1	0,457534	0,0585	0,306845	-0,42831	0,362936	0,399893			0,340304	0,721933		0,27454
pesas Antecipa	0,958409123	0,955328264	0,908808504	0,416393	0,921367	0,863621		0,457534	1	0,507593	0,778174	-0,60659	0,608618	0,002986			0,647511	0,423038		0,922493
os Ativos Circul	0,549040036	0,593593031	0,528923017	0,570062	0,343785	0,577106		0,0585	0,507593	1	0,791586	-0,30769	0,798802	-0,10683			0,753901	0,543675		0,668579
ealizável a Long	0,860400838	0,800540788	0,653940412	0,536517	0,707786	0,866518		0,306845	0,778174	0,791586	1	-0,42986	0,835165	-0,02226			0,957313	0,50768		0,87563
nceiras Avaliad	-0,659146204	-0,637890924	-0,532843817	-0,24861	-0,63937	-0,65048		-0,42831	-0,60659	-0,30769	-0,42986	1	-0,46285	-0,06669			-0,46732	-0,35775		-0,60088
iras Avaliadas a	0,696585136	0,727448037	0,617670374	0,325818	0,588139	0,693488		0,362936	0,608618	0,798802	0,835165	-0,46285	1	0,087697			0,846226	0,510033		0,683974
ontas a Receber	0,055061086	0,054217332	-0,037144493	-0,29236	-0,02066	0,109637		0,399893	0,002986	-0,10683	-0,02226	-0,06669	0,087697	1			0,002167	0,188017		-0,0433
Estoques.1																				
tivos Biológicos.1																				
ributos Diferido	0,765576653	0,700062438	0,518233309	0,397405	0,608009	0,810459		0,340304	0,647511	0,753901	0,957313	-0,46732	0,846226	0,002167			1	0,536572		0,738239
esas Antecipad	0,50808069	0,575609436	0,365452198	-0,00478	0,32319	0,649438		0,721933	0,423038	0,543675	0,50768	-0,35775	0,510033	0,188017			0,536572	1		0,373336
com Partes Relacionadas																				
Ativos Não Circ	0,942631314	0,892109542	0,813432535	0,663717	0,849021	0,88059		0,27454	0,922493	0,668579	0,87563	-0,60088	0,683974	-0,0433			0,738239	0,373336		1
Investimentos	-0,427414693	-0,323497411	-0,287378388	-0,45861	-0,44963	-0,346		-0,0004	-0,39831	-0,10313	-0,42328	-0,22482	-0,1939	0,00066			-0,24168	0,096367		-0,4688
Imobilizado	0,992353219	0,934969206	0,764030626	0,421271	0,926479	0,97502		0,597422	0,925549	0,517503	0,846638	-0,6672	0,680116	0,085792			0,764522	0,553299		0,915935
Intangível	0,982536777	0,895869256	0,754462321	0,498094	0,940726	0,928556		0,479284	0,937797	0,449125	0,821	-0,6439	0,606182	0,048688			0,716687	0,393597		0,933917
Diferido																				
Passivo Total	1	0,957693316	0,82373359	0,450075	0,941069	0,956474		0,543142	0,958409	0,54904	0,860401	-0,65915	0,696585	0,055061			0,765577	0,508081		0,942631
assivo Circulant	0,911031571	0,962261929	0,824825591	0,283172	0,813973	0,917479		0,676032	0,870403	0,68333	0,817991	-0,61465	0,786683	0,083505			0,75688	0,749632		0,826151
es Sociais e Tra	0,951477281	0,94873385	0,863531048	0,374257	0,877256	0,891785		0,489414	0,922004	0,634364	0,897369	-0,63011	0,81335	0,059601			0,842977	0,512341		0,895512
Fornecedores	0,95985032	0,976469441	0,885604489	0,402094	0,915738	0,906975		0,562865	0,950404	0,559066	0,775255	-0,64831	0,675995	0,058617			0,64954	0,523826		0,919016

Figura 3.26: Correlação Indicadores

3.4.5 Comparações dos dados dos grupos de empresas e geração de gráficos

As empresas foram dividas em grupos de atuação (Carnes e Derivados, Petróleo, Gás e Biocombustíveis e Tecidos e Vestuário).

Foi selecionado um mesmo indicador de um mesmo grupo e foram tomados dois a dois, e a partir dessa seleção foi feita a correlação e gerado gráficos lineares com relação ao tempo, utilizando o *script* I.7 anexo escrito em *Python*.

Na Figura 3.27 resultante do processo descrito no parágrafo anterior foi feito o gráfico linear no decorrer do tempo do indicador PSR, que representa o quanto o preço das ações esta correlacionado positiva ou negativamente à capacidade de gerar receita da empresa. Para as empresas BRFS3 e JBSS3 que são do setor de carnes e derivados, a correlação resultante foi de mais de 93%, o que, de acordo com o que foi convencionado para o trabalho, é considerado alta.

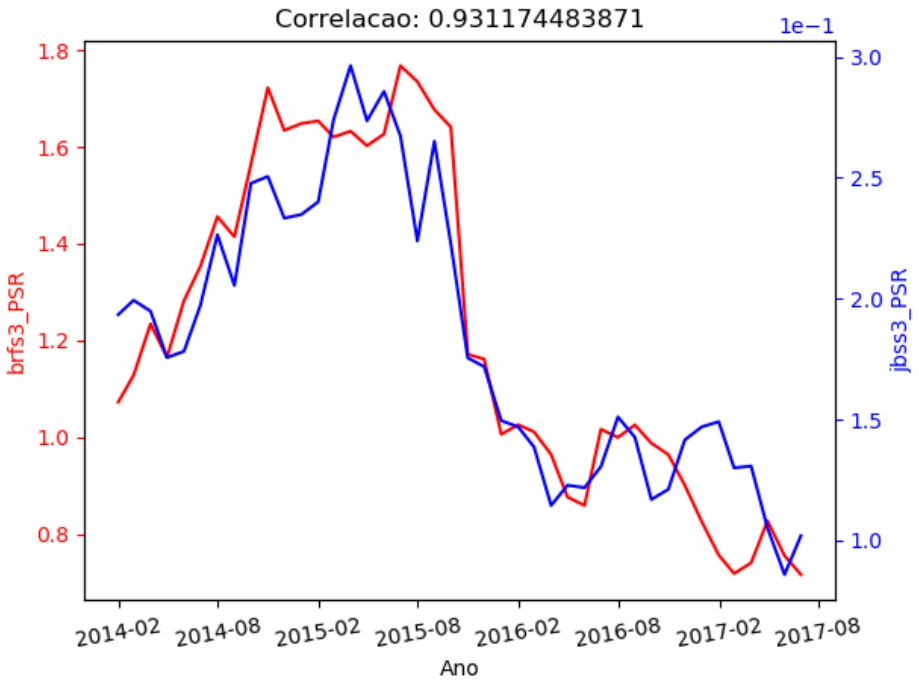


Figura 3.27: Carnes e derivados PSR BRFS3 x JBSS3

3.4.6 Comparações entre os dados das cotações deflacionadas e os dados Ibovespa deflacionados e geração de gráficos

Utilizando um *script* em *Python* foram geradas as correlações das cotações deflacionadas pelo Ibovespa deflacionado e, então, foi gerado um gráfico linear dos dois indicadores, para

cada empresa, em relação ao tempo.

A Figura 3.28 mostra o comportamento no decorrer do tempo da cotação e da cotação da empresa Ambev e do Ibovespa. A correlação é de -72% o que indica que tais indicadores estão inversamente relacionados nesta proporção.

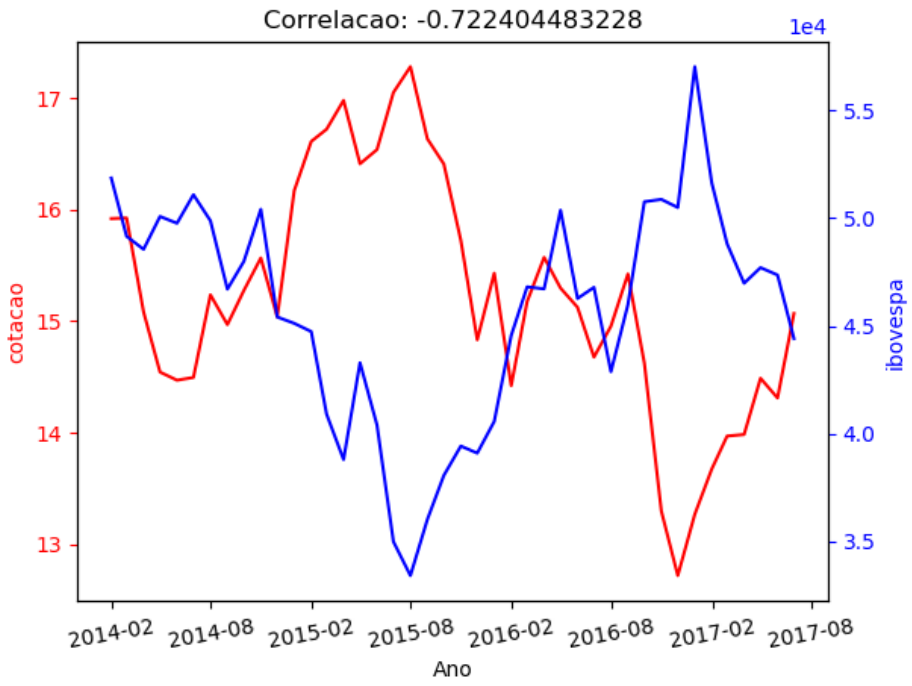


Figura 3.28: abev3 cotacao X ibovespa

3.4.7 Comparação dos dados das cotações das empresas e da taxa de juros reais e geração de gráficos

Utilizando um *script* em *Python* foram geradas as correlações das cotações deflacionadas em relação aos juros reais deflacionados e, então, foi gerado um gráfico linear contemplando os dois indicadores, para cada empresa, em relação ao tempo.

A Figura 3.29 mostra o comportamento no decorrer do tempo da cotação da empresa bvmf3 e do juro real, a correlação é de mais de 44%, o que indica que esses indicadores estão relacionados nesta proporção.

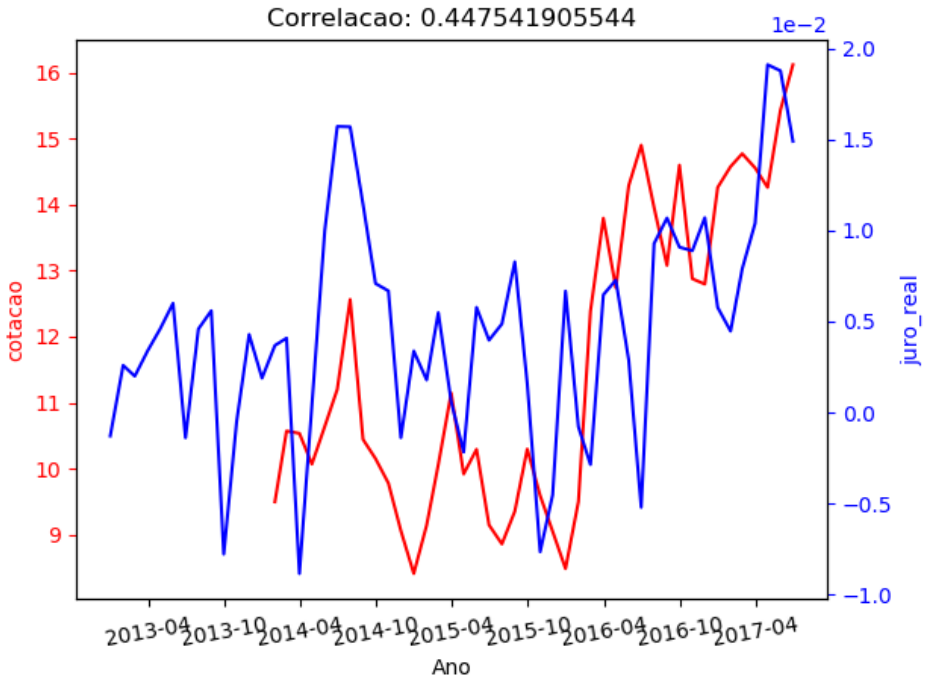


Figura 3.29: bvmf3 Cotação X Juros Reais

3.4.8 Comparação dos dados das cotações

Utilizando um *script* em *Python* foram descobertas as correlações entre as cotações de todas as empresas contra todas empresas, com as cotações deflacionadas. Foram produzidos, então, gráficos lineares de todas as cotações das empresas tomadas duas a duas em relação ao tempo.

A Figura 3.30 mostra um exemplo do descrito no parágrafo anterior, mostrando que a correlação das cotações da amar3 e da llis3 no decorrer do período de janeiro de 2014 à agosto de 2017 é de quase 80%.

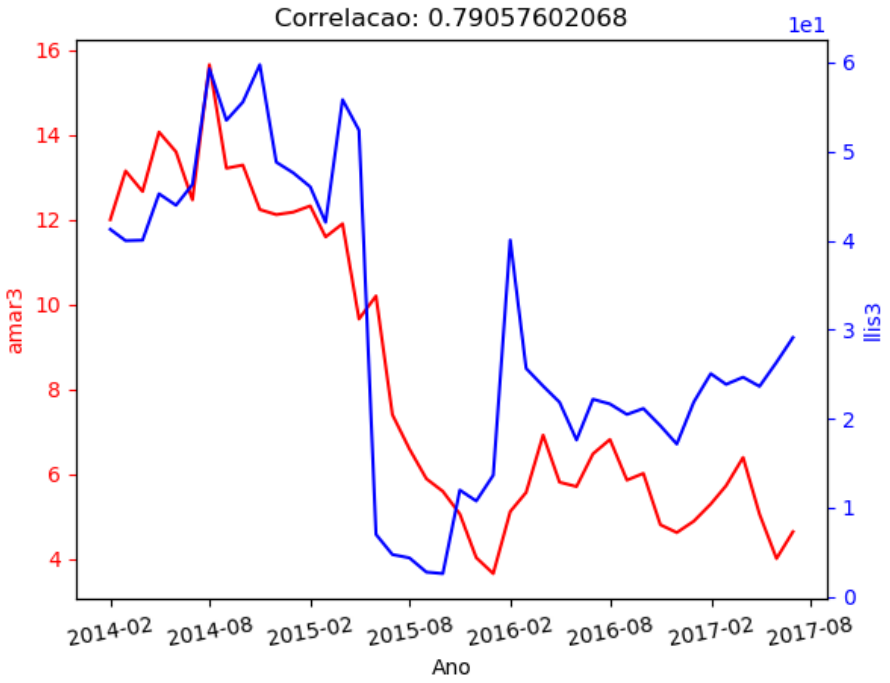


Figura 3.30: amar3 cotacao X llis3 cotacao

3.4.9 Comparação dos indicadores e Ibovespa

Utilizando o *script* I.18 em anexo escrito em *Python*, foram feitas as correlações dos indicadores deflacionados de todas as empresas contra o Ibovespa deflacionado, e, então, foram produzidos gráficos lineares dos indicadores pela Ibovespa

A Figura 3.31 mostra um exemplo do descrito no parágrafo anterior, mostrando que a correlação do Ibovespa e do ativo realizável a longo prazo da Ambev no período determinado é menor que -81%, o que indica que esses indicadores estão inversamente relacionados nesta proporção.

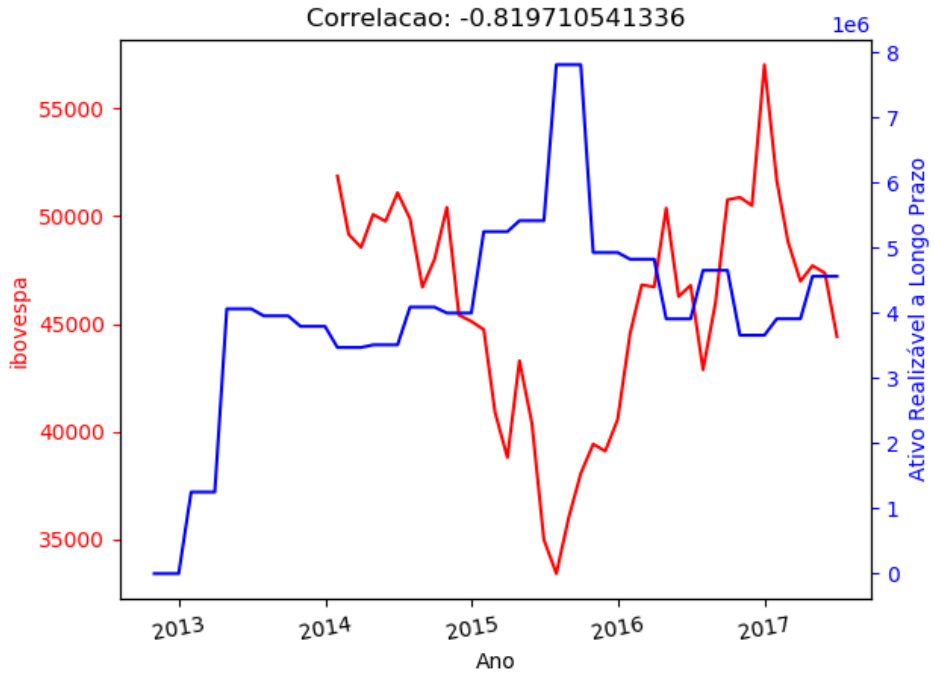


Figura 3.31: abev3 ibovespa X Ativo Realizavel a Longo Prazo

3.4.10 Comparação dos indicadores e taxa de juros reais

Utilizando o *script* I.19 em anexo escrito em *Python* foram descobertas as correlações dos indicadores deflacionados de todas as empresas contra o juro real deflacionado. Foram, então, feitos gráficos lineares dos indicadores pela Ibovespa.

A Figura 3.31 mostra um exemplo do descrito no parágrafo anterior, mostrando que a correlação do juro real e do "empréstimos e financimentos" da Ambev no período determinado é maior que 47%, o que indica que esses indicadores estão relacionados nesta proporção.

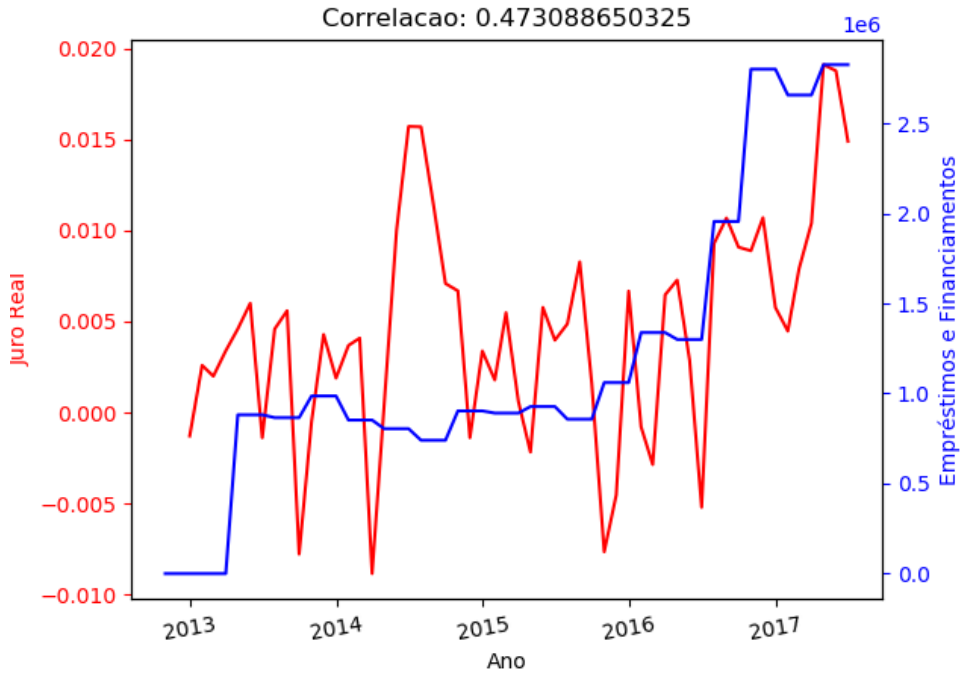


Figura 3.32: abev3 juro real X empréstimos e financiamentos

3.4.11 Comparação dos dados das cotações diárias

Utilizando o *script* I.4 em anexo escrito em *Python*, foram obtidos do site Yahoo! Finanças [4] as cotações diárias de todas empresas da Bovespa no período de Janeiro de 2014 à dezembro de 2018. O valor das cotações foi deflacionado usando o IGP-M.

Foi feita utilizando o *script* I.9 em anexo escrito em *Python* a correlação dos dados obtidos e manipulados conforme descrito no parágrafo anterior. O resultado foi 20706 (vinte mil setessentos e seis) correlações, das quais foram selecionadas as melhores (mais próximas de 100% ou -100%.

A Figura 3.33 mostra os maiores e menores resultado das correlações das cotações diárias deflacionadas das empresas da Ibovespa agrupadas duas a duas. Podem-se destacar a correlação alta da TCSA3 e a TECN3 com uma correlação de mais de 98% e a correlação inversa alta das empresas BRKM3 e PIBB11 de menos de -93%.

Empresas	Correlação	Empresas	Correlação
OIBR3_X_OIBR4	0,99888459	BRKM3_X_PIBB11	-0,939307794
UNIP5_X_UNIP6	0,998005655	BRKM5_X_PIBB11	-0,934736649
BRAP3_X_BRAP4	0,997531515	EQTL3_X_TRPN3	-0,9333243
GOAU3_X_GOAU4	0,997066069	RADL3_X_TRPN3	-0,931609745
CMIG3_X_CMIG4	0,993806629	MNDL3_X_PIBB11	-0,927505053
INEP3_X_INEP4	0,988855349	EEEL3_X_PIBB11	-0,90640667
BRKM3_X_BRKM5	0,987416862	ETER3_X_RADL3	-0,906267271
ELET3_X_ELET6	0,984175427	EQTL3_X_ETER3	-0,891748516
SANB11_X_SANB3	0,983882515	BRFS3_X_BRKM3	-0,89042598
TCSA3_X_TECN3	0,981602169	BRFS3_X_BRKM5	-0,885195973
ITSA4_X_ITUB4	0,978295302	MGLU3_X_PIBB11	-0,884746086
MILS3_X_SLED4	0,977916789	BRKM3_X_VLID3	-0,878172149

Figura 3.33: Correlação Cotações Diárias Cruzamento Empresas

3.4.12 Clusterização dos dados e geração de dendogramas para visualização de resultados

A última etapa da análise dos dados baseou-se na clusterização dos dados e no estudo dos resultados.

O conjunto de dados utilizados é composto por todas as empresas em um único arquivo *.csv*. Tal arquivo continha os dados dos balanços, bem como as informações sobre os indicadores fundamentalistas para cada uma das empresas do grupo de estudo. Os dados eram trimestrais. Foram gerados 14 arquivos para esta etapa (de forma a compreender todo o período estudado).

Foi, ainda, gerado um segundo conjunto de dados contendo os dados das empresas divididas por grupos (Carne e Derivados, Petróleo, Gás e Biocombustíveis, Tecidos e Vestuário).

Para esta fase utilizou-se a clusterização hierárquica, pois, assim seria possível descobrir quais eram as empresas mais semelhantes entre si com base em seus indicadores fundamentalistas. A ideia foi fazer uso da semelhança dos indicadores para que pudéssemos obter candidatos para análises adicionais futuras.

Poderíamos, assim, testar a hipótese de que empresas com indicadores semelhantes podem ter desempenhos semelhantes.

Um ponto importante a ser considerado é que o fato de algumas das empresas terem sido consideradas semelhantes na clusterização não implica que estas terão comportamentos semelhantes; tal informação é apenas um indicativo que pode ser verificado.

Dendrogramas

Para auxiliar na análise dos resultados, foram gerados dendrogramas (Seção 2.3.2) onde as observações eram os nomes das ações das empresas em estudo. Com a geração, foi possível observar alguns pontos interessantes à pesquisa.

No que diz respeito aos dendrogramas gerais, isto é, que contêm todas as empresas (Figuras 3.34, 3.35 e 3.36), foi possível realizar as seguintes observações:

- Notou-se semelhança entre as ações da Petrobrás (ptr3 e ptr4). Tal semelhança pode ser explicada pelo fato de ambas serem ações pertencentes à mesma empresa (embora sejam de tipos diferentes).
- Observou-se que, em 2014, a Ambev (abev3) foi a empresa que apresentou maior diferença com relação às outras empresas selecionadas (no que diz respeito aos indicadores fundamentalistas). Tal posição foi tomada pela Cambuci S.A. (camb4) em 2015 e pela PetroRio (prio3) em 2016.
- Em 2014, era possível observar a proximidade entre as empresas Minerva Foods (beef3) e Marfrig (mrfg3); tal proximidade não se manteve nos anos seguintes.
- Em 2014, era possível observar a proximidade entre as empresas Excelsior (bauh4) e JBS (jbss3); tal proximidade se manteve em 2015, mas não em 2016.

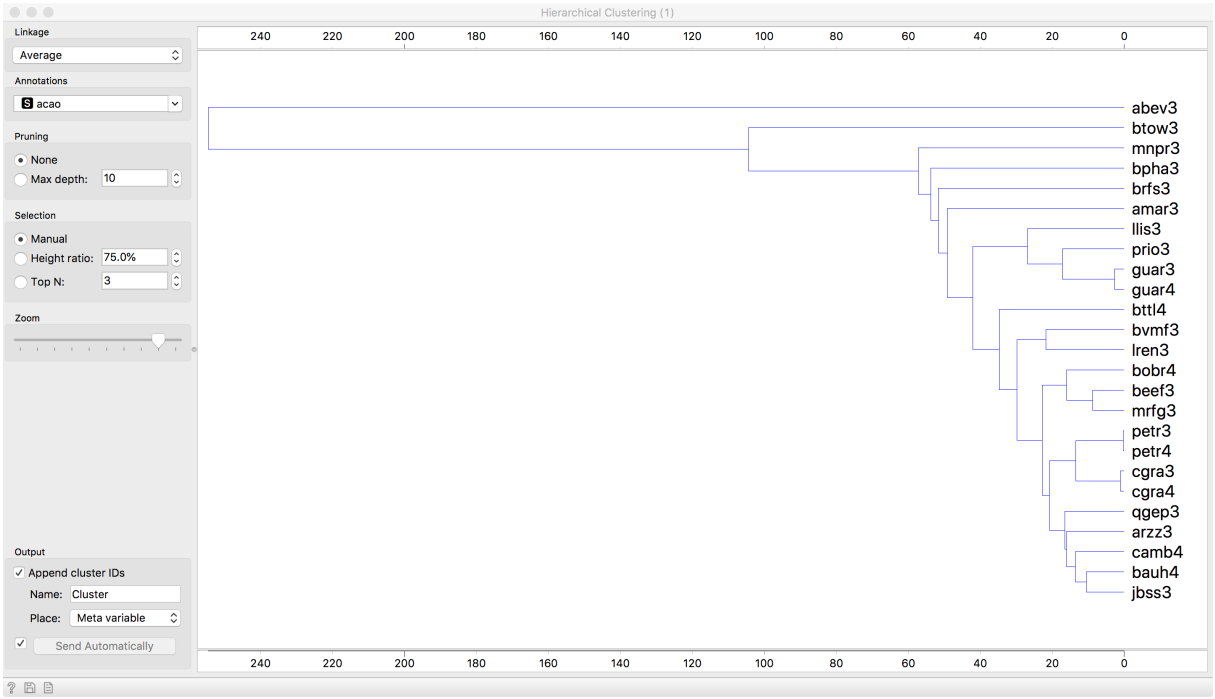


Figura 3.34: Dendrograma gerado a partir dos dados do trimestre iniciado em 31/12/2014

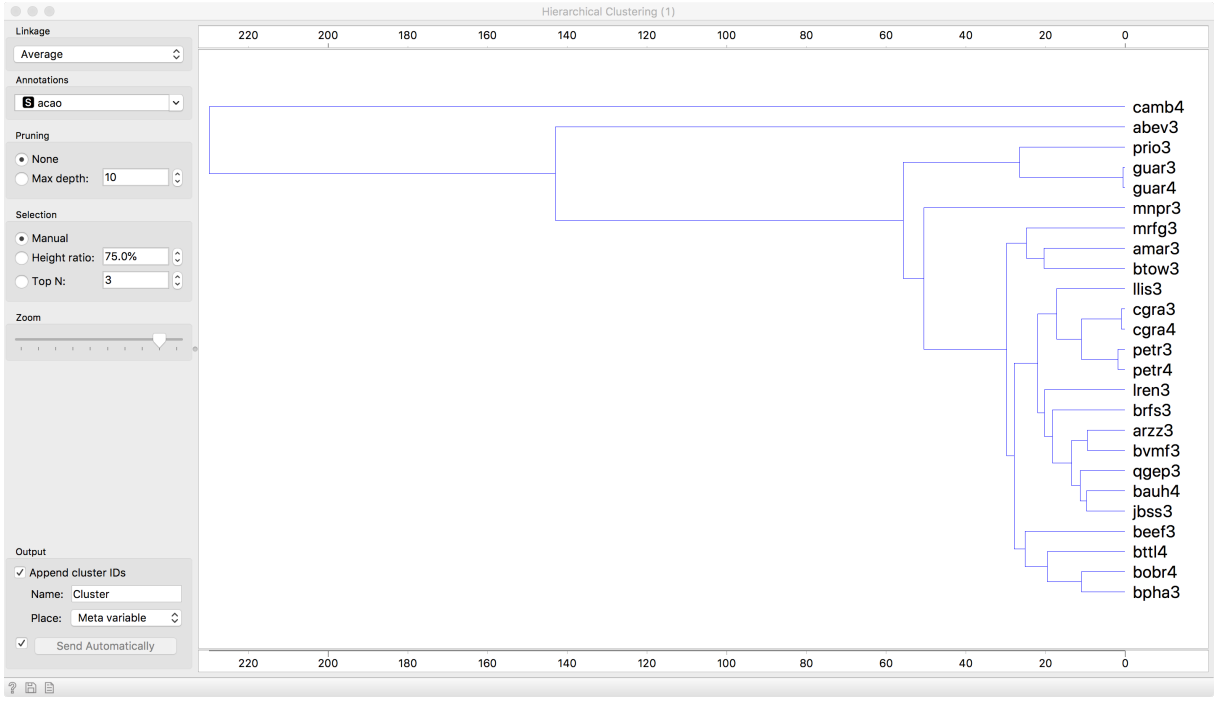


Figura 3.35: Dendrograma gerado a partir dos dados do trimestre iniciado em 31/12/2015

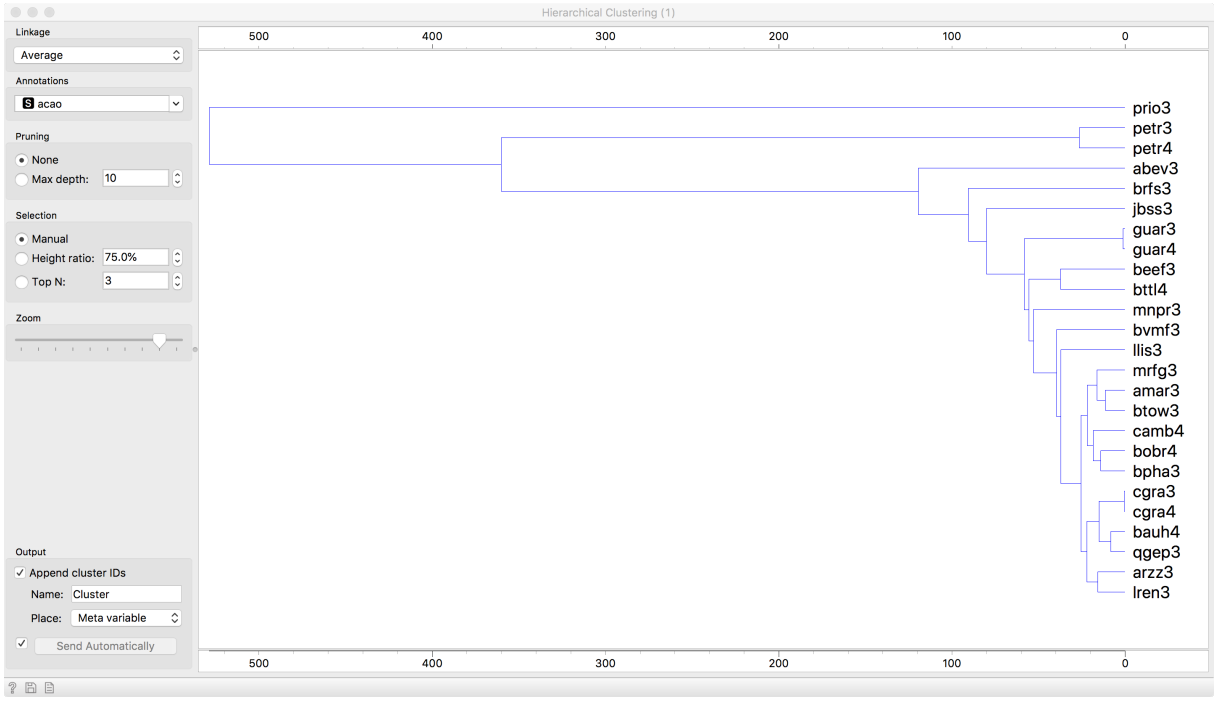


Figura 3.36: Dendrograma gerado a partir dos dados do trimestre iniciado em 31/12/2016

No que diz respeito aos dendrogramas específicos, para o grupo Carnes e Derivados (Figuras 3.37, 3.38 e 3.39), foi possível realizar as seguintes observações:

- Percebe-se que a empresa que apresenta maior distância com relação às demais

empresas do grupo é a BRF (brfs3). Tal estado se mantém de 2014 a 2016 (período retratado nos dendrogramas).

- Em 2014, era possível observar a proximidade entre as empresas Minerva Foods (beef3) e Marfrig (mrfg3); tal proximidade não se manteve nos anos seguintes (conforme era possível observar, também, no dendrograma geral).
- Em 2014, era possível observar a proximidade entre as empresas Excelsior (bauh4) e JBS (jbss3); tal proximidade se manteve em 2015, mas não em 2016 (conforme era possível observar, também, no dendrograma geral).
- Em 2016, era possível observar a proximidade entre as empresas Excelsior (bauh4) e Marfrig (mrfg3); em 2015 e em 2016 as duas empresas apresentavam uma distância maior entre si.

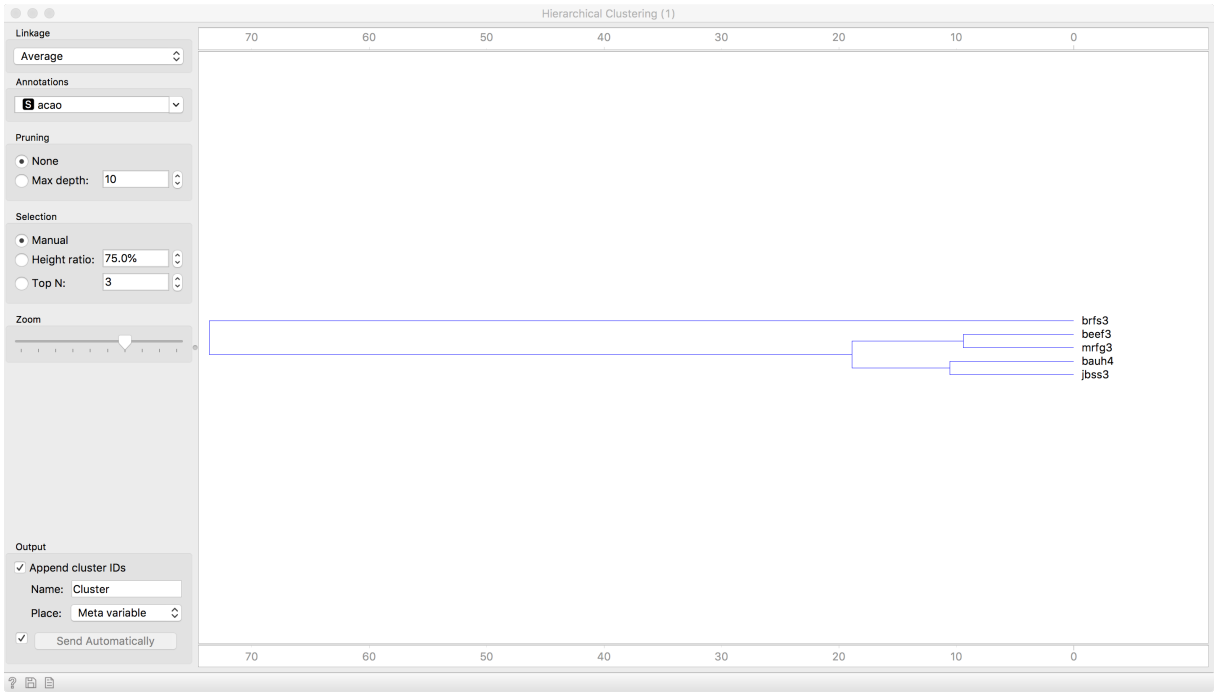


Figura 3.37: Dendrograma gerado a partir dos dados do trimestre iniciado em 31/12/2014 (Grupo Carnes e Derivados)

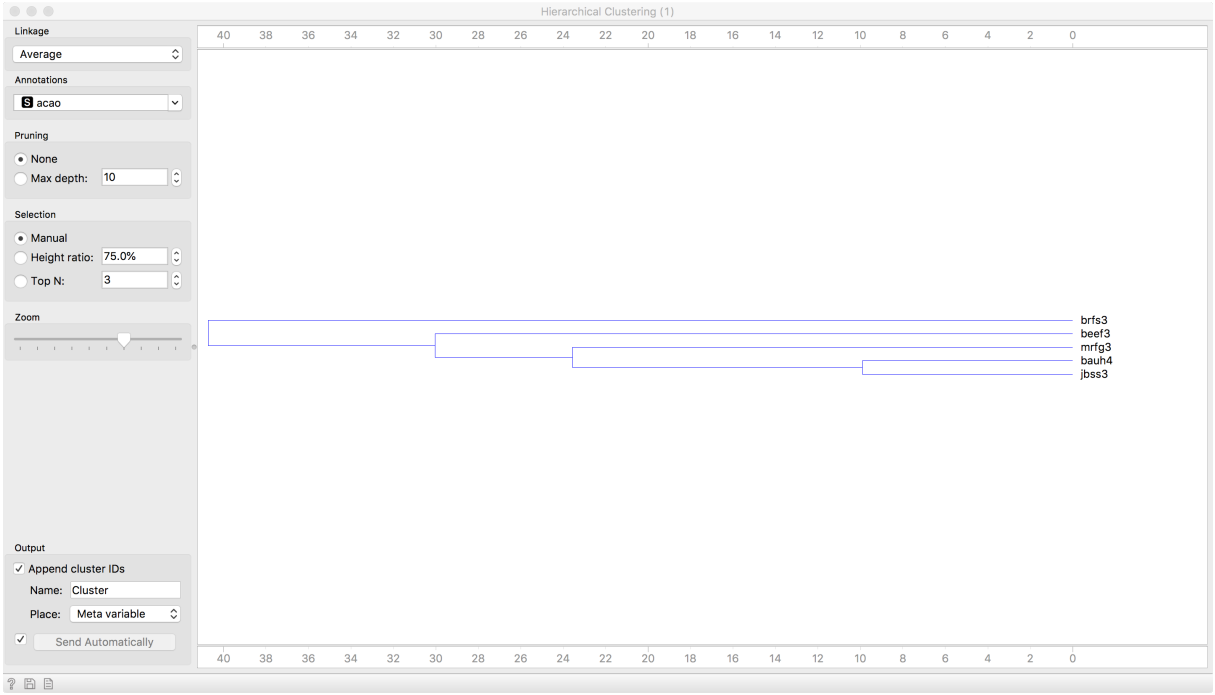


Figura 3.38: Dendrograma gerado a partir dos dados do trimestre iniciado em 31/12/2015 (Grupo Carnes e Derivados)

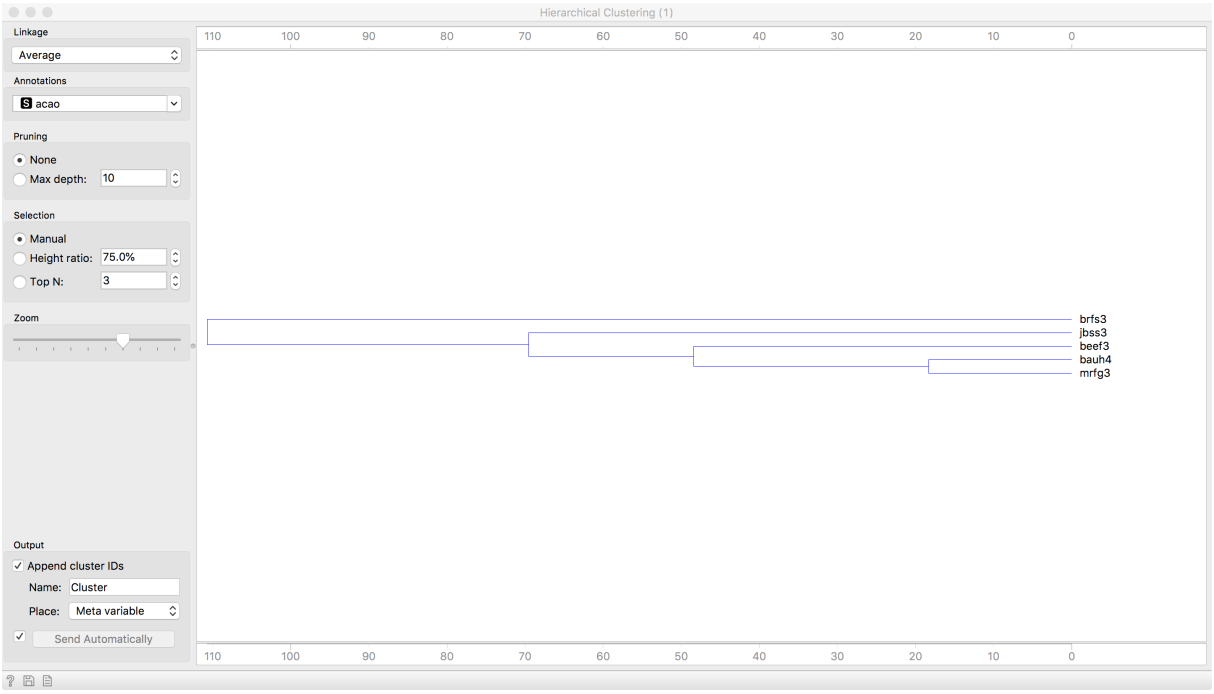


Figura 3.39: Dendrograma gerado a partir dos dados do trimestre iniciado em 31/12/2016 (Grupo Carnes e Derivados)

Ainda no que diz respeito aos dendrogramas específicos, para o grupo Petróleo, Gás e Biocombustíveis (Figuras 3.40, 3.41 e 3.42), foi possível realizar as seguintes observações:

- Percebe-se que a empresa que apresenta maior distância com relação às demais empresas do grupo é a PetroRio (prio3). Tal estado se mantém de 2014 a 2016 (período retratado nos dendrogramas).
- Em 2014, 2015 e 2016 era possível observar a proximidade entre as ações petr3 e petr4; tal comportamento pode ser explicado pelo fato de que ambas são ações da mesma empresa (Petrobrás).

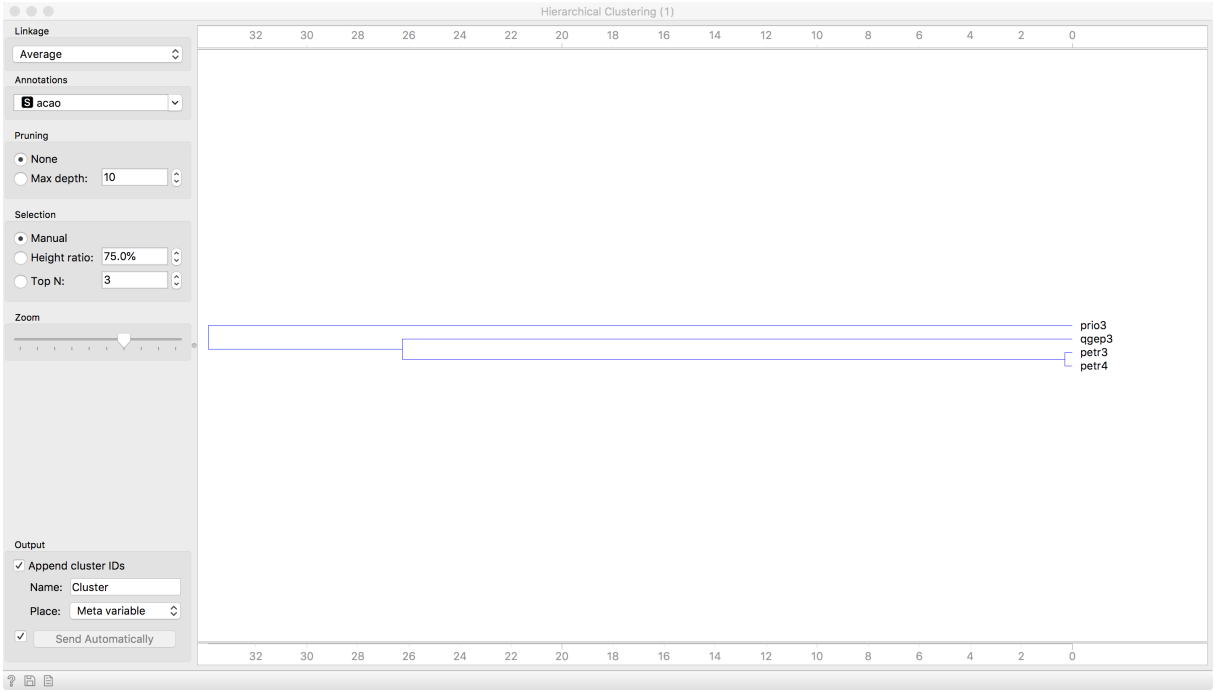


Figura 3.40: Dendrograma gerado a partir dos dados do trimestre iniciado em 31/12/2014 (Grupo Petróleo, Gás e Biocombustíveis)

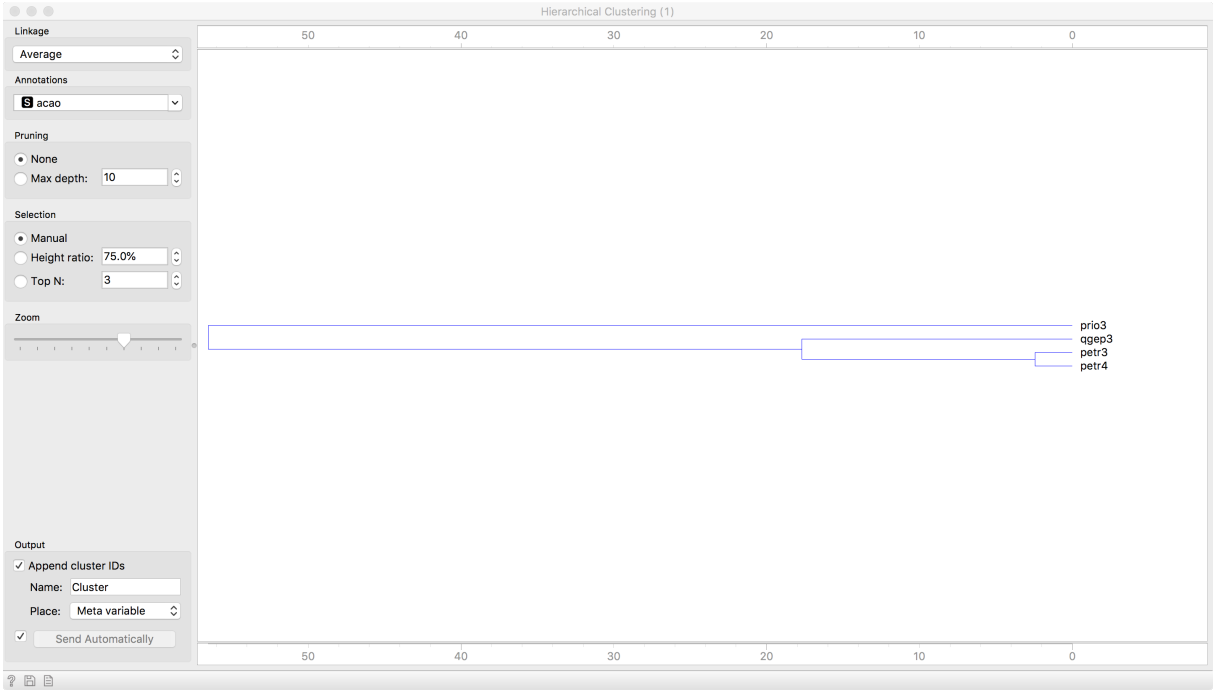


Figura 3.41: Dendrograma gerado a partir dos dados do trimestre iniciado em 31/12/2015 (Grupo Petróleo, Gás e Biocombustíveis)

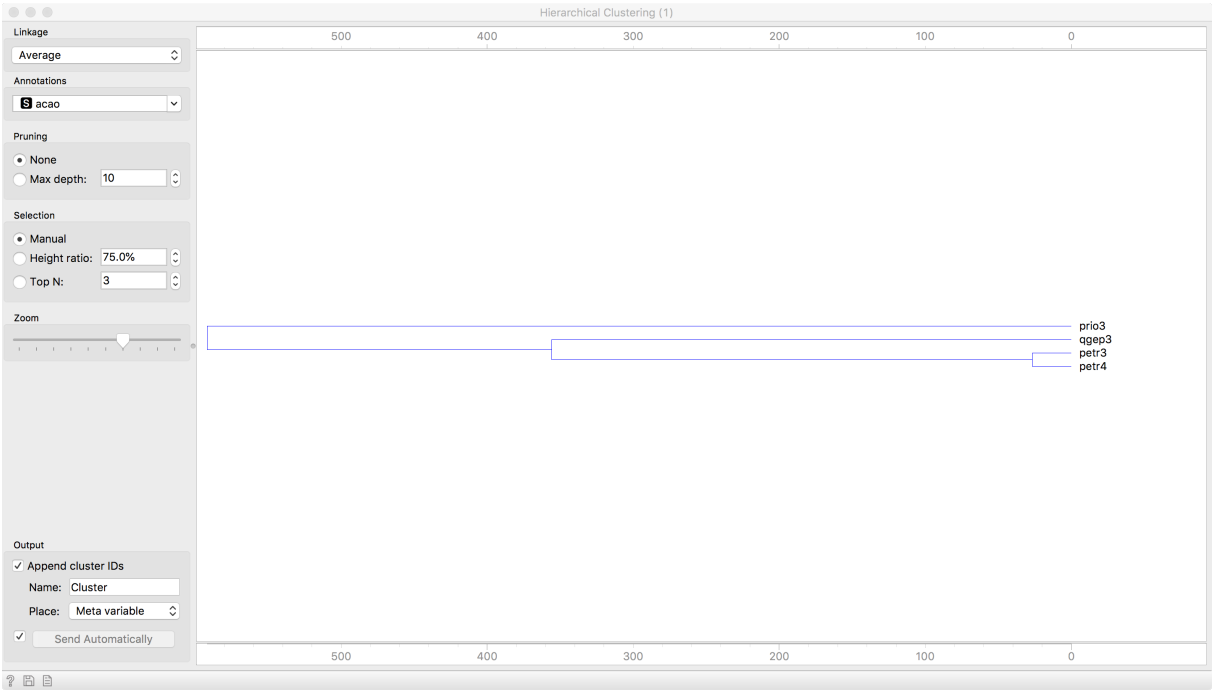


Figura 3.42: Dendrograma gerado a partir dos dados do trimestre iniciado em 31/12/2016 (Grupo Petróleo, Gás e Biocombustíveis)

Ainda no que diz respeito aos dendrogramas específicos, para o grupo Tecidos e Vestuário (Figuras 3.43, 3.44 e 3.45), foi possível realizar as seguintes observações:

- Percebe-se que no ano de 2014 há a presença de dois grupos que possuem uma distância maior entre si; o primeiro grupo é formado pelas empresas Restoque Comércio e Confecções de Roupas (llis3) e Guararapes Confecções (guar3 e guar4), enquanto o segundo grupo é formado pelas empresas Lojas Marisa (amar3), Lojas Renner (lren3), Arezzo (arzz3) e Grazziotin S.A. (cgra3 e cgra4).
- Em 2015 ainda há a presença de dois grupos que possuem uma distância maior entre si, mas estes são diferentes dos de 2014. O primeiro grupo é formado pela empresa Grazziotin S.A. (cgra3 e cgra4), enquanto o segundo grupo é formado pelas empresas Lojas Marisa (amar3), Arezzo (arzz3), Lojas Renner (lren3), Restoque Comércio e Confecções de Roupas (llis3) e Guararapes Confecções (guar3 e guar4).
- Em 2016 ainda há a presença de dois grupos que possuem uma distância maior entre si, mas estes são diferentes dos de 2014 e dos de 2015. O primeiro grupo é formado pela empresa Guararapes Confecções (guar3 e guar4), enquanto o segundo grupo é formado pelas empresas Restoque Comércio e Confecções de Roupas (llis3), Lojas Marisa (amar3), Grazziotin S.A. (cgra3 e cgra4), Arezzo (arzz3) e Lojas Renner (lren3).
- Em 2014, 2015 e 2016 era possível observar a proximidade entre as ações cgra3 e cgra4; tal comportamento pode ser explicado pelo fato de que ambas são ações da mesma empresa (Grazziotin S.A.).
- Em 2014, 2015 e 2016 era possível observar a proximidade entre as ações guar3 e guar4; tal comportamento pode ser explicado pelo fato de que ambas são ações da mesma empresa (Guararapes Confecções).

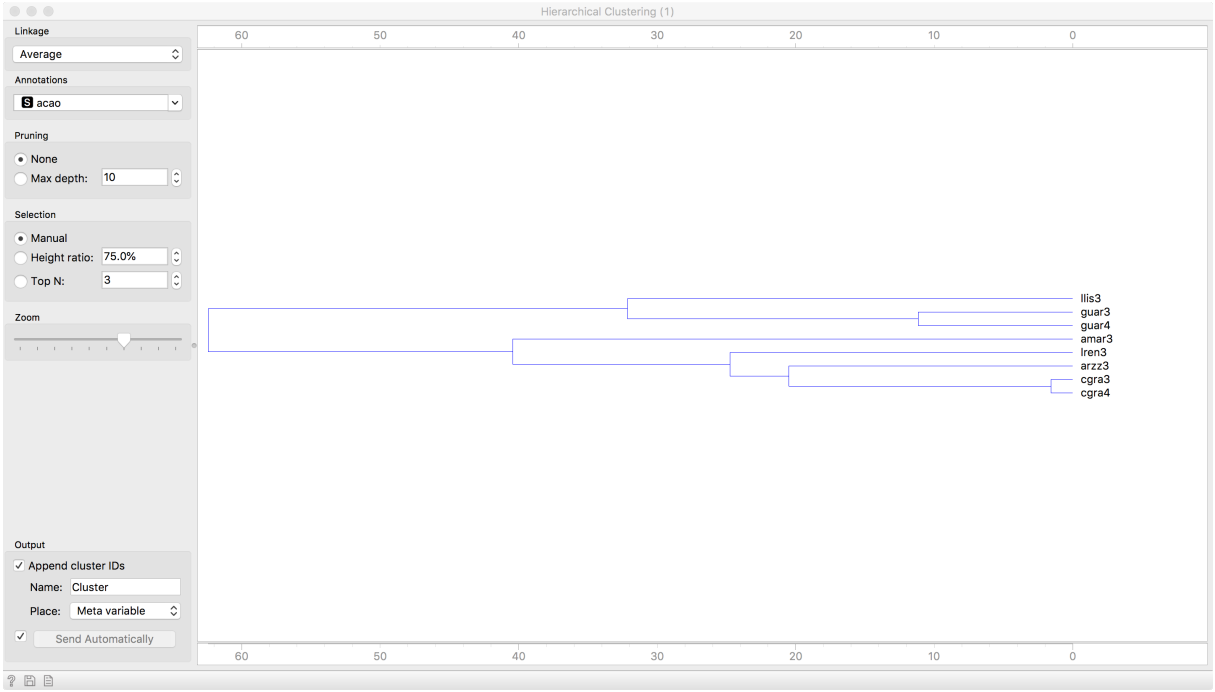


Figura 3.43: Dendrograma gerado a partir dos dados do trimestre iniciado em 31/12/2014 (Grupo Tecidos e Vestuário)

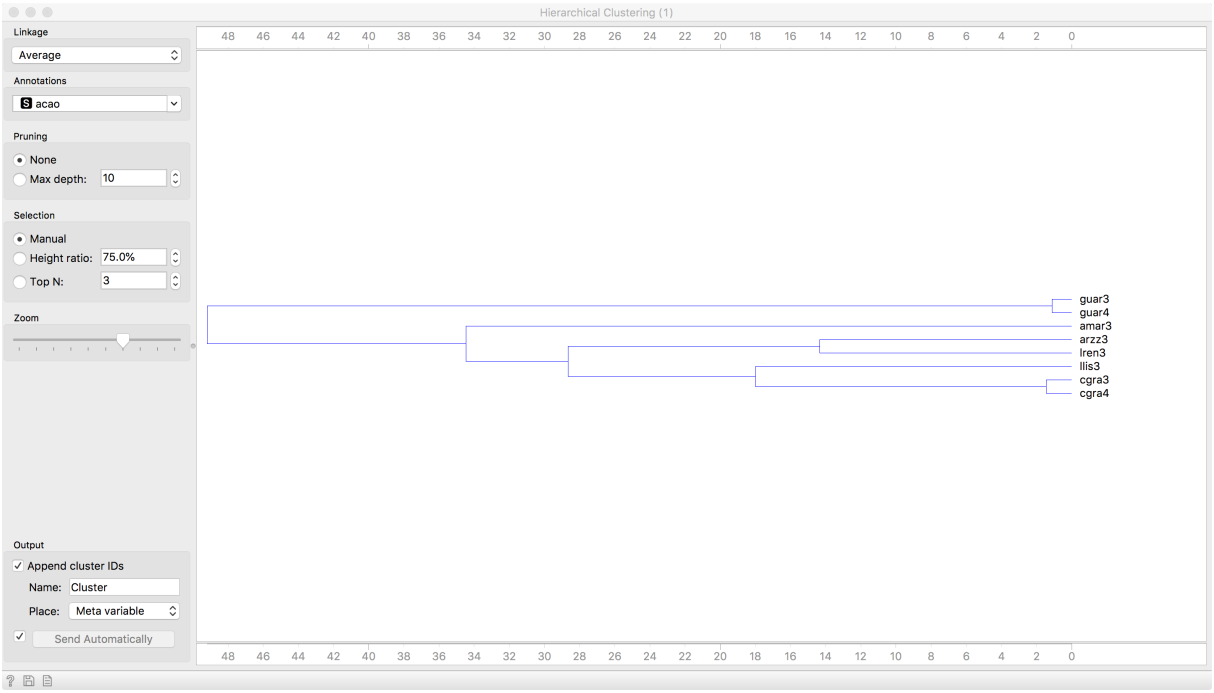


Figura 3.44: Dendrograma gerado a partir dos dados do trimestre iniciado em 31/12/2015 (Grupo Tecidos e Vestuário)

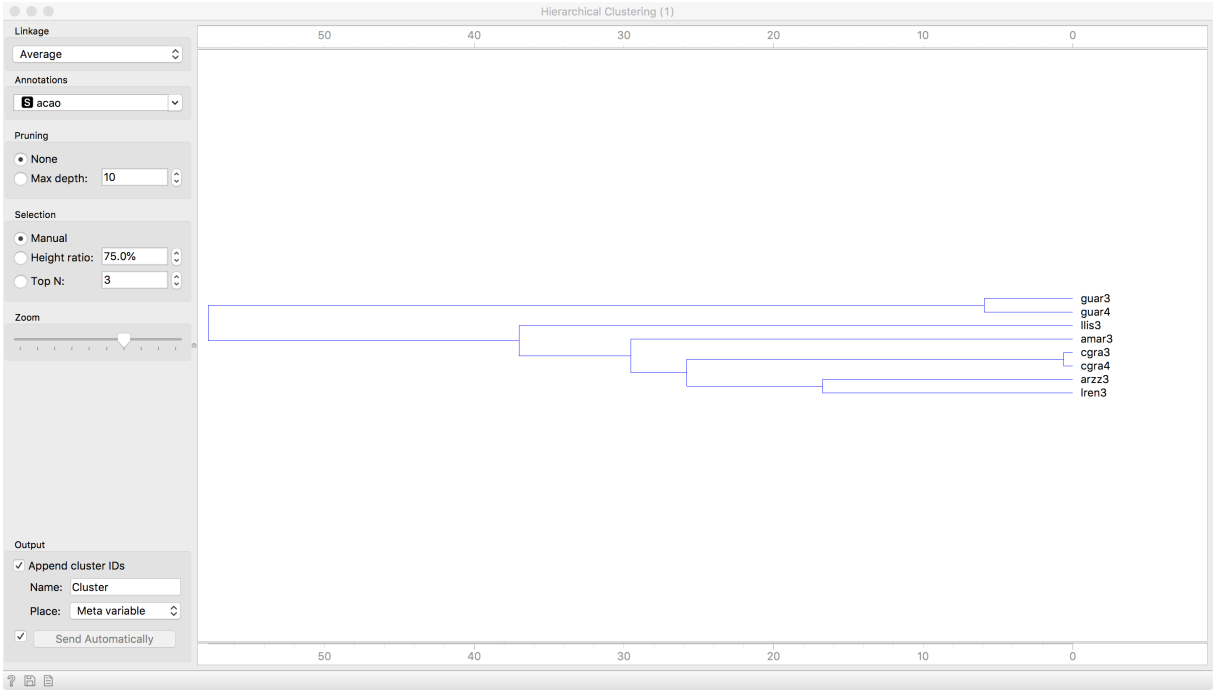


Figura 3.45: Dendrograma gerado a partir dos dados do trimestre iniciado em 31/12/2016 (Grupo Tecidos e Vestuário)

Distance maps

Como parte da análise final, também foram gerados *distance maps* (ou mapas de distância). Como explicado anteriormente (Seção 2.3.2), tais gráficos se caracterizam por promover a visualização da distância entre os objetos que estão sob análise através do uso de espaços coloridos (cadências de cores), onde, por exemplo, o branco significa distância zero e o preto significa distância máxima. As cores que aparecem entre o preto e o branco são as distâncias intermediárias. Distâncias pequenas significam que as empresas são semelhantes e distâncias grandes significam que empresas são menos semelhantes. Tudo isto em relação aos indicadores fundamentalistas.

Ao realizar a análise dos mapas gerais (Figuras 3.46 e 3.47), foram observados os seguintes pontos:

- A empresa Petrobrás encontrava-se mais próxima às demais empresas em 2014; tal condição mudou com o passar do tempo, encontrando-se esta bem distante das outras empresas em 2016; no entanto, seus dois tipos de ações são sempre muito próximos (o que reafirma o que foi observado nos dendrogramas discutidos anteriormente).
- A Grazziotin S.A. (cgra3 e cgra4) e a Arezzo (arezz3) eram próximas em 2014 e continuaram próximas em 2016.

- A empresa Ambev (abev3) encontrava-se distante de todas as demais empresas em 2014; em 2016 é possível notar uma aproximação.
- A empresa PetroRio encontrava-se mais próxima às demais empresas em 2014; tal condição mudou com o passar do tempo, encontrando-se esta bem distante das outras empresas em 2016.

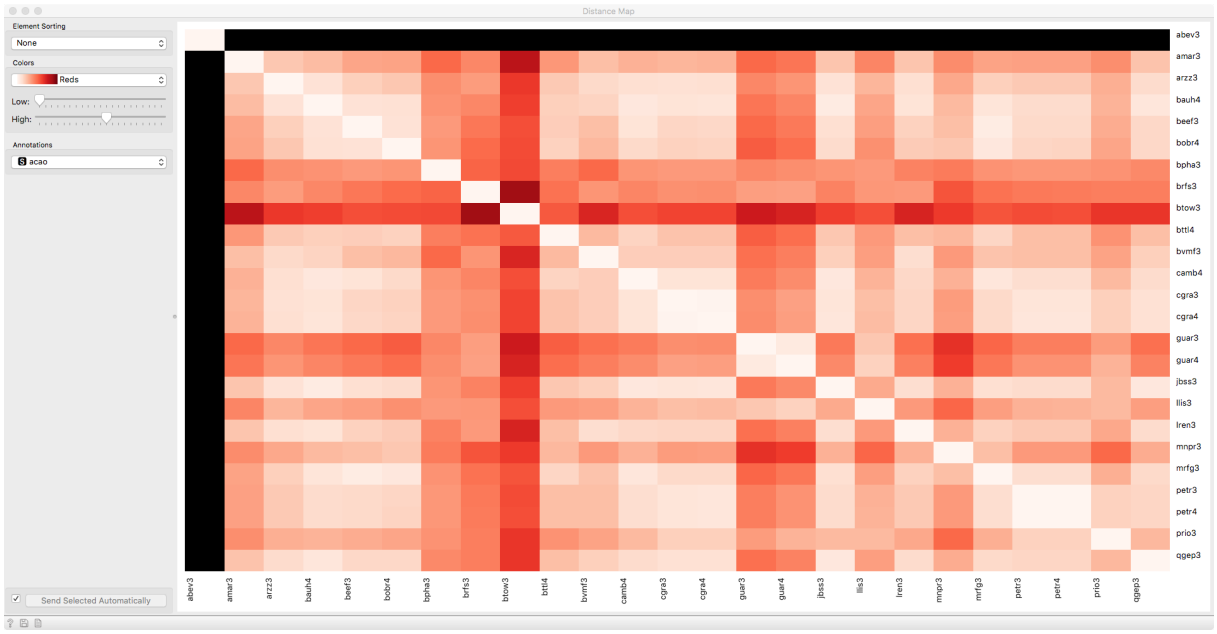


Figura 3.46: Distance Map gerado a partir dos dados do trimestre iniciado em 31/12/2014

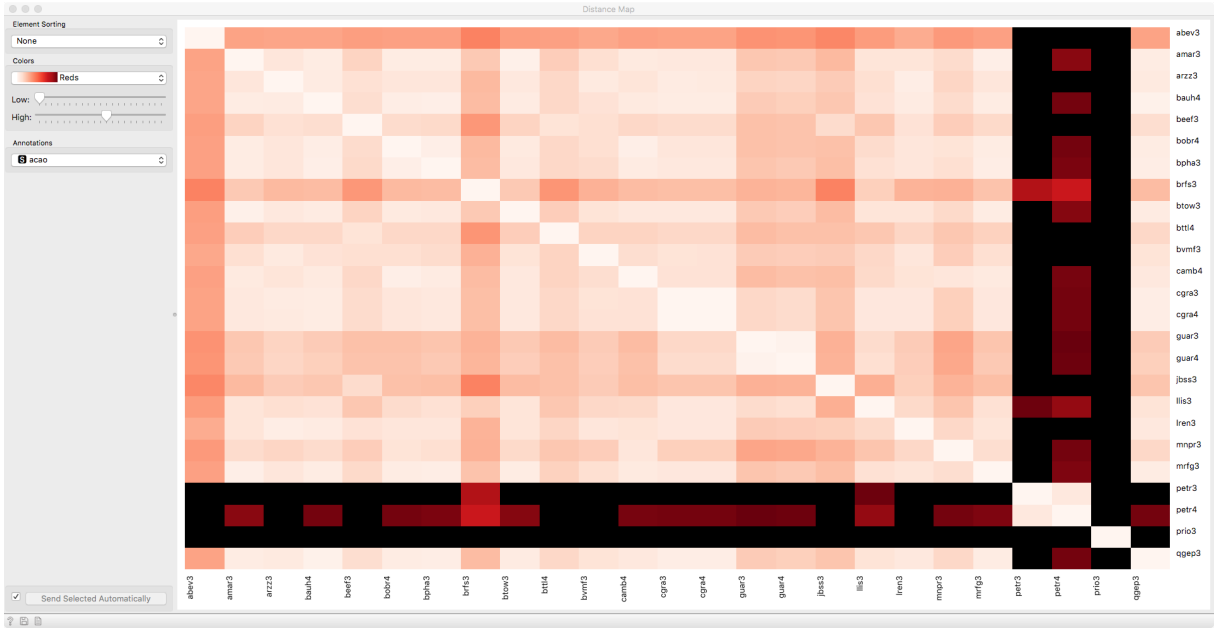


Figura 3.47: Distance Map gerado a partir dos dados do trimestre iniciado em 31/12/2016

No que diz respeito aos *distance maps* específicos, para o grupo Carnes e Derivados (Figuras 3.48 e 3.49), foi possível realizar as seguintes observações:

- Percebe-se que a empresa que apresenta maior distância com relação às demais empresas do grupo é a BRF (brfs3). Tal estado se mantém em 2014 e em 2016 (período retratado nos *distance maps*). Tal fato reafirma o que foi observado nos *distance maps* discutidos anteriormente.
- Em 2014, era possível observar a proximidade entre as empresas Minerva Foods (beef3) e Marfrig (mrfg3); tal proximidade não se manteve nos anos seguintes (conforme era possível observar, também, no *distance map* geral).
- Em 2014, era possível observar a proximidade entre as empresas Excelsior (bauh4) e JBS (jbss3); tal proximidade não se manteve em 2016 (conforme era possível observar, também, no *distance map* geral).

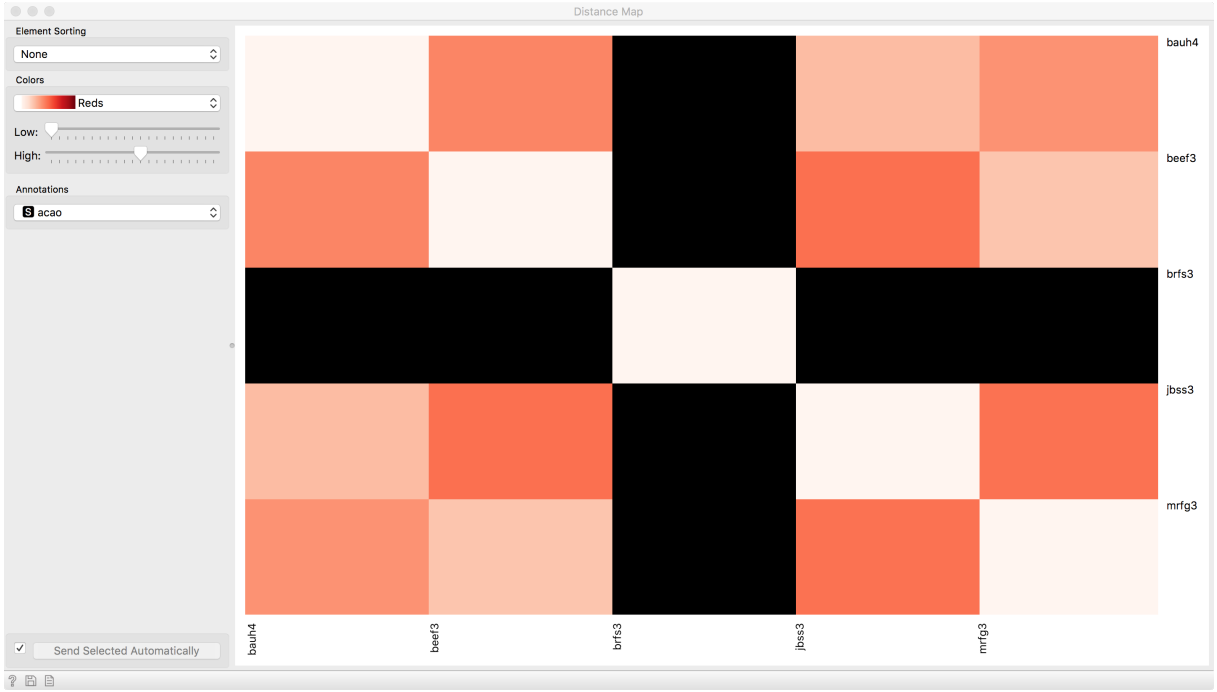


Figura 3.48: Distance Map gerado a partir dos dados do trimestre iniciado em 31/12/2014 (Grupo Carnes e Derivados)

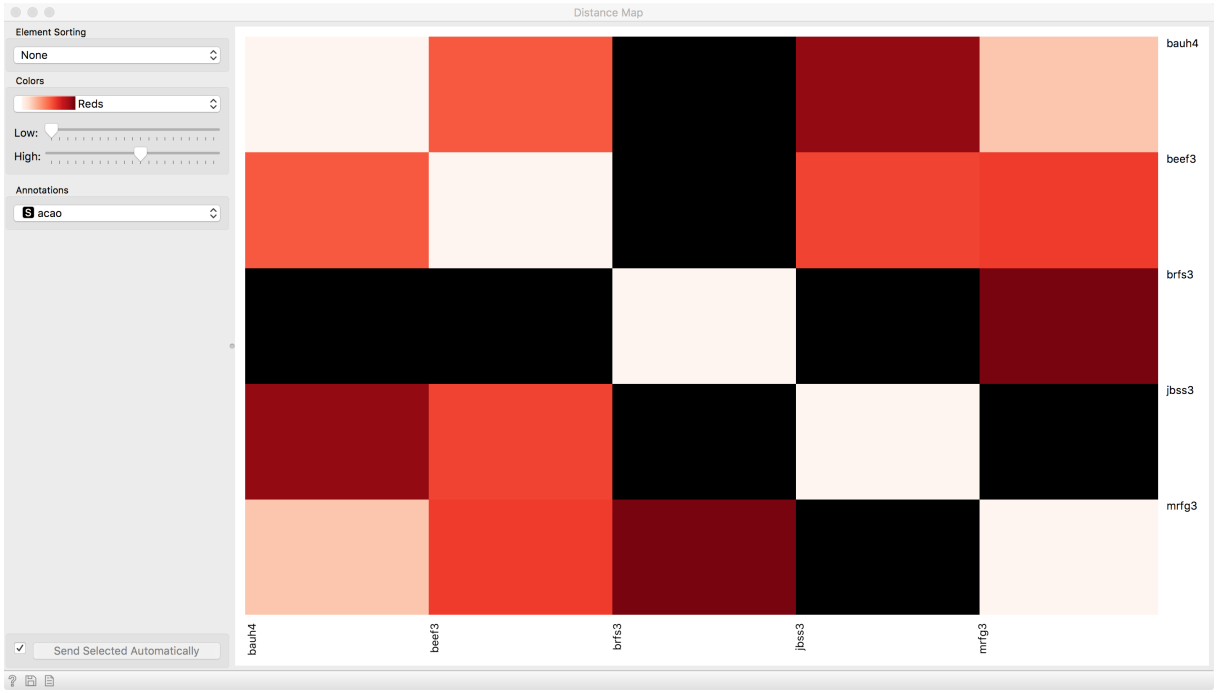


Figura 3.49: Distance Map gerado a partir dos dados do trimestre iniciado em 31/12/2016 (Grupo Carnes e Derivados)

Ainda no que diz respeito aos *distance maps* específicos, para o grupo Petróleo, Gás e Biocombustíveis (Figuras 3.50 e 3.51), foi possível realizar as seguintes observações:

- Percebe-se que a empresa que apresenta maior distância com relação às demais empresas do grupo é a PetroRio (prio3). Tal estado se mantém em 2014 e em 2016 (o que confirma o que foi observado nos *distance maps* gerais).
- Em 2014 e 2016 era possível observar a proximidade entre as ações petr3 e petr4; tal comportamento é compreensível, tendo em vista que ambas são ações da mesma empresa (Petrobrás) (o que, mais uma vez, confirma o que foi observado nos *distance maps* gerais).

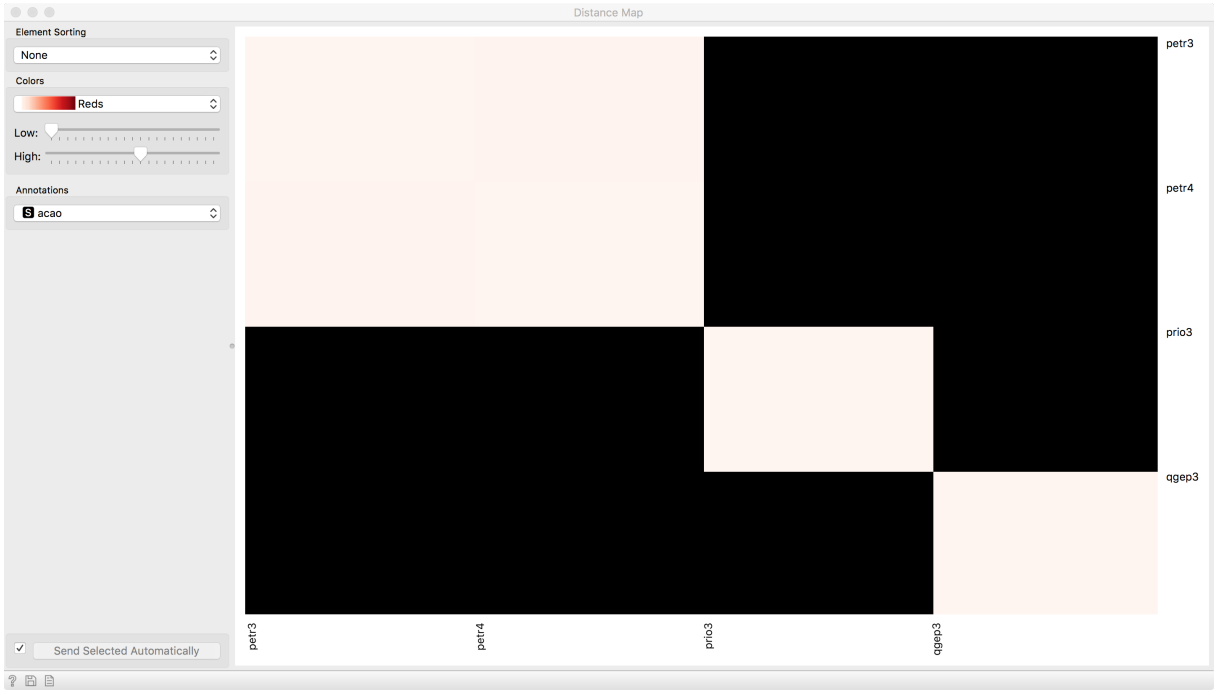


Figura 3.50: Distance Map gerado a partir dos dados do trimestre iniciado em 31/12/2014 (Grupo Petróleo, Gás e Biocombustíveis)

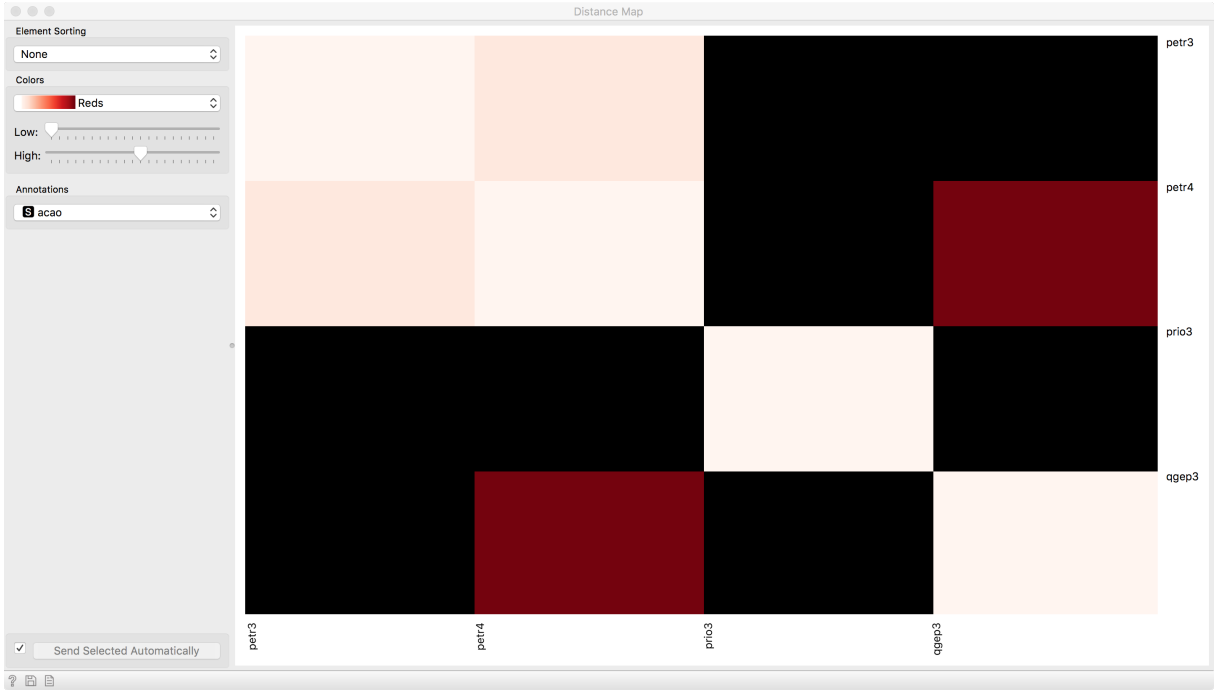


Figura 3.51: Distance Map gerado a partir dos dados do trimestre iniciado em 31/12/2016 (Grupo Petróleo, Gás e Biocombustíveis)

Ainda no que diz respeito aos *distance maps* específicos, para o grupo Tecidos e Vestuário (Figuras 3.52 e 3.53), foi possível realizar as seguintes observações:

- Percebe-se que a empresa que apresenta maior distância com relação às demais empresas do grupo é a Guararapes Confecções (guar3 e guar4). Tal estado se mantém em 2014 e em 2016 (o que confirma o que foi observado nos *distance maps* gerais).
- Em 2014 as empresas diferentes que apresentavam menor distância entre si (com relação às demais empresas do grupo) eram Arezzo (arzz3) e Grazziotin S.A. (cgra3 e cgra4). Tal estado não se manteve em 2016.
- As empresas Grazziotin S.A. (cgra3 e cgra4) e Lojas Marisa (amar3) apresentavam grande distância em 2014 e se aproximaram em 2016.
- As empresas Arezzo (arzz3) e Lojas Renner (lren3) mantiveram a mesma distância em 2014 e em 2016.
- As empresas Guararapes Confecções (guar3 e guar4) e Restoque Comércio e Confecções de Roupas (llis3) se distanciaram entre 2014 e 2016.

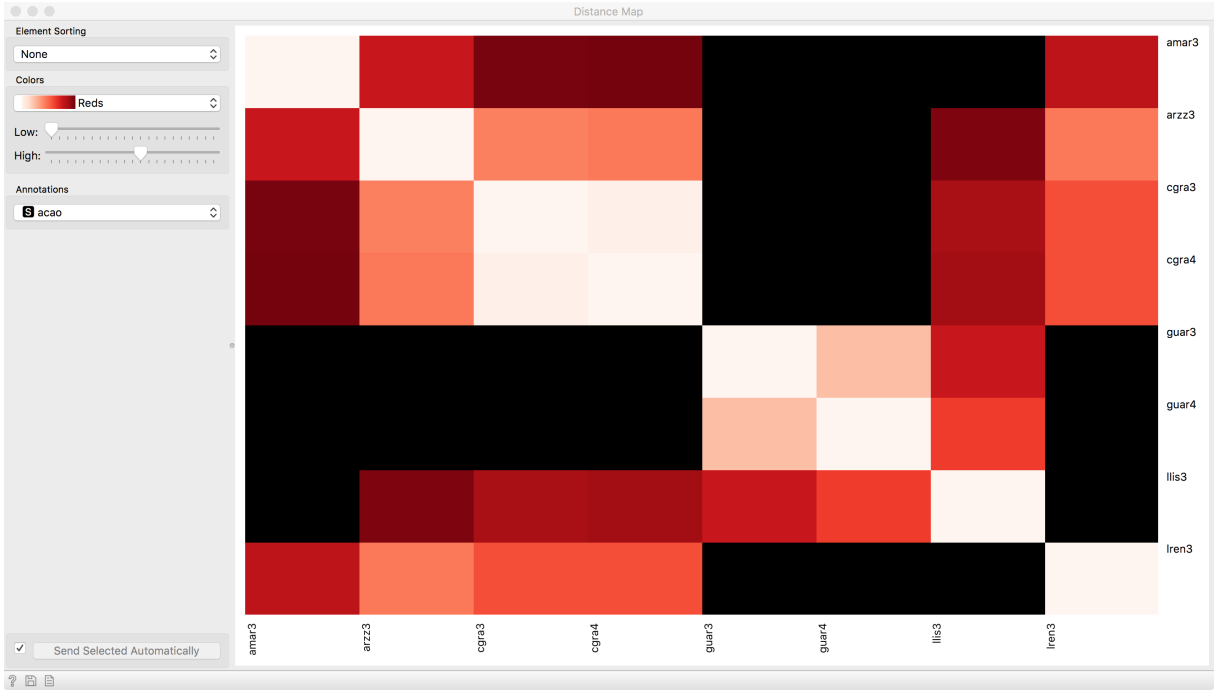


Figura 3.52: Distance Map gerado a partir dos dados do trimestre iniciado em 31/12/2014 (Grupo Tecidos e Vestuário)

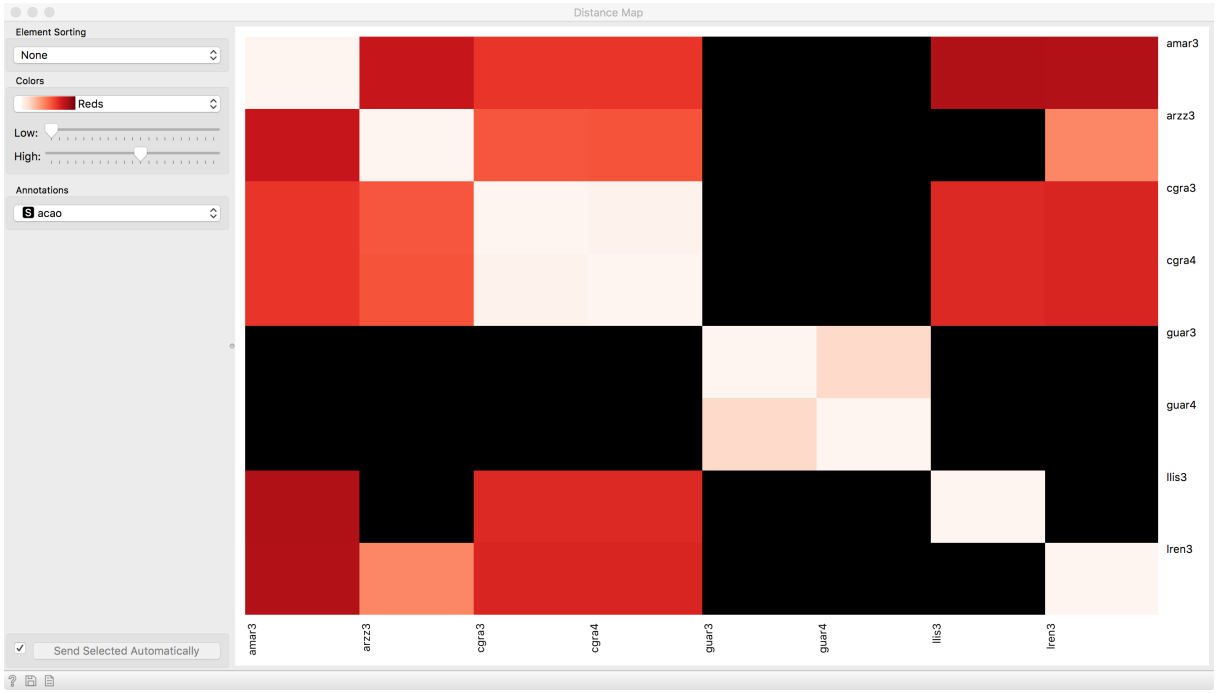


Figura 3.53: Distance Map gerado a partir dos dados do trimestre iniciado em 31/12/2016 (Grupo Tecidos e Vestuário)

Capítulo 4

Conclusão

Neste trabalho foram estudados dados de balanço e cotações de empresas presentes na bolsa de valores. Os resultados obtidos podem ser utilizados como análise inicial para futuras seleções de candidatos para estudos. O processo envolveu a obtenção e tratamento dos dados, classificação, clusterização e geração de gráficos.

Inicialmente os dados foram obtidos do site Fundamentus de acordo com os critérios definidos para a seleção das empresas a serem utilizadas no estudo. Os critérios principais consistiam na presença dos indicadores fundamentalistas no balanço e no período de dados disponível (que deveria ser de dezembro de 2013 até junho de 2017) e a importância do setor. Os principais grupos utilizados no projeto foram: Carnes e Derivados, Petróleo, Gás e Biocombustíveis e Tecidos e Vestuário.

Após o processo de obtenção dos dados, foi preciso tratá-los, de forma que estivesse de acordo para o uso das ferramentas de mineração de dados que seriam utilizadas. Os dados dos balanços foram integrados em uma só planilha (para cada uma das empresas selecionadas), foram adicionados dados sobre as cotações do período estudado e alguns outros indicadores que foram descritos ao longo do capítulo de análise de dados.

Depois de tratados, os dados foram usados para a mineração de dados propriamente dita. Foram utilizadas abordagens diferentes para que pudessem ser obtidos resultados mais robustos para a análise. Desta forma, para cada uma das abordagens o conjunto de dados sofria alguma alteração em sua estrutura básica (conforme descrito no capítulo de análise de dados).

Para o processo de classificação dos dados foi utilizado o *software Weka* e o algoritmo *lazy.IBk*. Neste processo foram obtidos resultados ao comparar empresas pertencentes ao mesmo grupo. Os arquivos de dados utilizados estavam estruturados de forma que possuíssem ou não deslocamento temporal para se tentar prever valores futuros baseado em valores passados.

Para os processos de obtenção de valores de correlação e geração de gráficos para análise, foi utilizada a biblioteca *Pandas* do *Python*. Tais processos incluíam diversas variações no que diz respeito à combinação de dados para geração de resultados (cotações *versus* dados da ibovespa, cotações *versus* taxa de juros do período, entre outras combinações).

Para o processo de clusterização e geração de dendrogramas e *distance maps* foi utilizado o *software Orange*. Para tanto, a estrutura dos dados consistia em duas organizações diferentes: a primeira baseava-se na junção de todas as planilhas de empresas em uma só de forma que cada planilha geral correspondesse a um trimestre de dados; a segunda, seguia o mesmo modelo da primeira, mas as planilhas eram divididas por grupos (Carnes e Derivados, Petróleo, Gás e Biocombustíveis e Tecidos e Vestuário), ou seja, cada planilha trimestral continha os dados das empresas pertencentes a um dos grupos e haviam planilhas para todos os grupos.

Ao final da análise, foi possível observar que os algoritmos de classificação conseguiram encontrar padrões relacionando indicadores fundamentalistas e as cotações futuras. Na clusterização hierárquica descobriu-se agrupamentos de empresas com situação financeira semelhante.

4.1 Trabalhos futuros

Como sugestão de trabalhos futuros, sugerimos o uso das informações obtidas para tentar alcançar resultados melhores na previsão do comportamento das empresas, realizar as análises usando mais empresas com indicadores ou mesmo usar outras ferramentas de mineração de dados (como redes neurais, por exemplo).

Referências

[1] DAVENPORT, T. H.; PRUSAK, L: *Ecologia da informação: por que só a tecnologia não basta para o sucesso na era da informação.* Futura, São Paulo, 1998. xvi, 4, 5, 7

[2] Cios, Krzysztof, Witold Pedrycz, Roman W. Swiniarski e Lukasz Kurgan: *Data Mining: A Knowledge Discovery Approach.* janeiro 2007, ISBN 978-0-387-36795-8. 1, 7, 8, 10

[3] *Fundamentus invista consciente.* <`http://www.fundamentus.com.br`>. Acesso em 6 dez. 2017. 3, 22, 26, 29, 34

[4] *Yahoo finanças.* <`https://br.financas.yahoo.com/`>. Acesso em 5 dez. 2017. 3, 23, 26, 27, 55

[5] CAICARA, Cícero; PARIS, Wanderson: *Informática, Internet e Aplicativo.* IBPEX, Curitiba, 2006. 4

[6] FAYYAD, U. M., PIATETSKY SHAPIRO G. SMITH P.: *From Data Mining to Knowledge Discovery: An Overview.In Advances in Knowledge Discovery and Data Mining.* AAAI Press, 1996. 5

[7] REZENDE, Denis Alcides: *Tecnologia da informação aplicada a sistemas de informação empresariais.* Editora Saraiva, São Paulo, 4ª edição, 2006. 5

[8] O'BRIEN, James A: *Sistemas de Informação e as Decisões Gerenciais na Era da Internet.* Editora Saraiva, São Paulo, 3ª edição, 2010. 5

[9] SETZER, VALDEMAR W.: *Dado, informação, conhecimento e competência.* <`https://www.ime.usp.br/~vwsetzer/dado-info.html`>. Acesso em 9 jun. 2019. 5

[10] DRUCKER, Peter: *Desafios Gerenciais para o Século XXI.* Thompson Learning,São Paulo, 1999. 6

[11] SETZER, V.W: *Dado, informação, conhecimento e competência. Os Meios Eletrônicos e a Educação: Uma Visão alternativa.* Datagrama, 10ª edição, 2001. 6, 7

[12] SEARLE, J. Minds: *Brains Science: the 1984 Reith Lectures.* . Penguin Books, New York, 1991. 6

[13] Han, Jiawei e Micheline Kamber: *Data Mining: Concepts and Techniques.* 2006. 7, 10

[14] Corporation, IBM: *IBM SPSS Modeler CRISP-DM Guide*. 2011. 8, 9, 10, 25, 28

[15] CAMILO, C. O., SILVA J. C.: *Mineração de Dados: Conceitos, Tarefas, Métodos e Ferramentas*. Instituto de Informática Universidade Federal de Goiás, 2009. 10, 11, 12

[16] *Árvore de decisão*. <https://www.maxwell.vrac.puc-rio.br/3710/3710_4.PDF>. Acesso em 3 jul. 2019. 10

[17] *Fundamentos teóricos - classificação*. <https://www.maxwell.vrac.puc-rio.br/9637/9637_3.PDF>. Acesso em 3 jul. 2019. 10

[18] OCHI, L. S., DIAS C. R. SOARES S. S. F.: *Clusterização em mineração de dados*. <http://www2.ic.uff.br/~satoru/conteudo/artigos/ERI-Minicurso-SATORU.pdf>. Acesso em 23 jun. 2019. 11

[19] *Dendrograma*. <https://support.minitab.com/pt-br/minitab/18/help-and-how-to/modeling-statistics/multivariate/how-to/cluster-observations/interpret-the-results/all-statistics-and-graphs/dendrogram/>. Acesso em 23 jun. 2019. 11

[20] *Distance map*. <https://docs.biolab.si//3/visual-programming/widgets/unsupervised/distancemap.html>. Acesso em 24 jun. 2019. 11

[21] ABERNETHY, Michael: *Data mining with weka, part 3: Nearest neighbor and server-side library*. <https://www.ibm.com/developerworks/library/os-weka3/os-weka3-pdf.pdf>. Acesso em 9 jun. 2019. 12

[22] GENNARI, J. H., LANGLEY P. FISHER D.: *Models of incremental concept formation*. Irvine Computational Intelligence Project, Department of Information and Computer Science, University of California, 1989. 13

[23] HALL, M. A.: *Correlation-based Feature Selection for Machine Learning*. Department of Computer Science, University of Waikato, 1999. 13

[24] FILHO, D. B. F., JÚNIOR J. A. S.: *Desvendando os mistérios do coeficiente de correlação de pearson (r)*. <https://periodicos.ufpe.br/revistas/politicahoje/article/viewFile/3852/3156>. Acesso em 3 jul. 2019. 13

[25] *Correlation does not imply causation*. <https://pdfs.semanticscholar.org/d886/674ab022d6d34447320ed62b96ebab9ead60.pdf>. Acesso em 3 jul. 2019. 13

[26] BORGES, L. E.: *Python para desenvolvedores*. Rio de Janeiro - Brasil: Editora Novatec, 2014. 13

[27] *The 3-Clause BSD License*. <https://opensource.org/licenses/BSD-3-Clause>. Acesso em 10 jun. 2019. 13

[28] *Python Data Analysis Library*. <https://pandas.pydata.org/>. Acesso em 10 jun. 2019. 13

[29] Eibe Frank, Mark A. Hall e Ian H. Witten: *The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques"*. Morgan Kaufmann, 2016. 14

[30] E. Frank, I. H. Witten e M. A. Hell: *Data Mining: Practical Machine Learning Tools and Technique*. Morgan Kaufmann, 2011. 14

[31] Witten, E. Frank I. H.: *Data Mining: Practical Machine Learning Tools and Technique*. Morgan Kaufmann, 2005. 14

[32] Demšar, J., Zupan B.: *Orange: Data mining fruitful and fun.* <`http://ailab.ijs.si/dunja/TuringSLAIS-2012/Papers/Demsar_Orange.pdf`>. Acesso em 26 jun. 2019. 15

[33] RASSIER, L. H.;HILGERT, S. Pn: *Aprenda a investir na bolsa de valores*. IESDE, Curitiba, 2009. 16, 17

[34] FEITOSA, Anderson: *O que é Capital Social e como definir seu valor?* <`https://conube.com.br/blog/o-que-e-capital-social/`>, 2018. Acesso em 12 out. 2018. 16

[35] BIANCHI, Bruno Parasmo: *A Evolução do Prêmio de Controle no Brasil*. INSPER, São Paulo, 2010. 17

[36] MILTERSTEINER, M. R: *A validade estatistica do uso de indices fundamentalistas no mercado de capitais brasileiro: um estudo aplicado ao setor bancario*. Universidade Federal de Santa Catarina, Florianopolis, 2003. 18

[37] SILVA, Cezar Augusto Tibúrcio; RODRIGUES, Fernanda Fernandes: *Curso Prático de Contabilidade*. Editora Atlas, São Paulo, 2ª edição, 2018. 18, 19, 20

[38] GITMAN, L.J: *Princípios de Administração Financeira*. Pearson Addison Wesley, São Paulo, 10ª edição, 2004. 20, 21, 22

[39] SPERANZINI, Milton Medeiros: *Efeitos da política de dividendos sobre valor das ações no mercado brasileiro de capitais*. Faculdade de Economica, Administração e Contabilidade da Universidade de São Paulo - FEA/USP, São Paulo, 1994. 21

[40] R. Vruwink, David, Jeffrey J. Quirin e David O'Bryan: *A modified price-sales ratio: A useful tool for investors?* Journal of Business Economics Research (JBER), 5, fevereiro 2011. 22

[41] BERESTEIN, M.: *Uso de Mineração de Dados na Bolsa de Valores*. Universidade do Vale do Itajaí; Centro de Ciências Tecnológicas da Terra e do Mar, 2010. 23

[42] BRANCO, G. M. R., BARROSO M. A. R.: *Mining StockTec: Predição de preço de ações através de mineração de dados e análise de sentimentos.* Universidade Federal do Estado do Rio de Janeiro; Escola de Informática Aplicada, 2014. 23

[43] LEITE, L. O. G.: *Mineração de Dados aplicada à previsão do preço de ações de concessionárias de energia elétrica do estado de São Paulo*. Universidade de São Paulo; Escola de Engenharia de São Carlos, 2016. 24

[44] *Taxa de juros selic.* <https://idg.receita.fazenda.gov.br/orientacao/tributaria/pagamentos-e-parcelamentos/taxa-de-juros-selic>. Acesso em 12 jun. 2019. 27

[45] Dornbusch, Rudiger e Stanley Fischer: *Macroeconomia*. São Paulo: Makron Books, 5ª edição, 1991. 27

[46] *Índice Geral de Preços do Mercado - Metodologia.* <http://portalibre.fgv.br/lumis/portal/file/fileDownload.jsp?fileId=8A7C82C53F58BFCA013F77D97BFA40ED>. Acesso em 10 jun. 2019. 27

[47] *Ibge. indicadores sociais.* <https://ww2.ibge.gov.br/home/mapa_site/mapa_site.php#indicadores>. Acesso em 12 jun. 2019. 28

[48] *Banco central do brasil. indicadores econômicos.* <https://www.bcb.gov.br/estatisticas/indicadoresconsolidados>. Acesso em 12 jun. 2019. 28

[49] *Instituto Brasileiro de Geografia e Estatística.* <https://ww2.ibge.gov.br/home/estatistica/indicadores/trabalhoerendimento/pnad_continua/primeiros_resultados/analise01.shtm>. Acesso em 26 jun. 2019. 28

[50] *Conjuntura Econômica.* <http://www.fazenda.gov.br/centrais-de-conteudos/publicacoes/conjuntura-economica>. Acesso em 10 jun. 2019. 28

[51] *INEC - Índice Nacional de Expectativa do Consumidor.* <https://www.portaldaindustria.com.br/estatisticas/inec-indice-nacional-de-expectativa-do-consumidor/>. Acesso em 10 jun. 2019. 28

Anexo I

I.1 deflaciona_balanco.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import codecs # evitar unicode erro
import sys # evitar unicode erro
import xlwt
import xlrd
import trimestre
import helper

UTF8Writer = codecs.getwriter('utf8') # evitar unicode erro
sys.stdout = UTF8Writer(sys.stdout) # evitar unicode erro


def planilhasCreate(nomeEmpresa):
 igpm = xlrd.open_workbook("IGPM/inflacao.xlsx")
 deflator = igpm.sheet_by_index(1) # pega a segunda planilha de inflacao

 planilhaBalanco = xlrd.open_workbook("balanco/"+nomeEmpresa+".xls") # Abre
     o arquivo do excel de balanco
 balanco = planilhaBalanco.sheet_by_index(0) # Pega a primeira planilha do
    excel de balanco
 demonstrativo = planilhaBalanco.sheet_by_index(1) # pega a segunda
    planilha do excel de balanco

 planilhaSaida = xlwt.Workbook(encoding = 'utf-8') # abre o arquivo do
    excel de saida

 balancoSaida = planilhaSaida.add_sheet(balanco.name)
 for curr_col in range(0,20): #coluna 0 (titulos) ate a 19 (dez 2012)
  for curr_row in range(balanco.nrows):
```

```
    if curr_col > 0 and curr_row > 1 and isinstance(balanco.cell_value(
        curr_row,curr_col),float): # curr_col > 0 nao esta nas colunas dos
        titulos, curr_row > 1 nao esta nas linhas das datas ou do titulo da
        planilha. Logo esta em algum valor numerico
     valor = balanco.cell_value(curr_row,curr_col)/deflator.cell_value(
         curr_col,1) # A coluna do balanco eh equivalente a linha do deflator
    else:
     valor = balanco.cell_value(curr_row,curr_col)
    balancoSaida.write(curr_row,curr_col,valor) # coloca valor na planilha
        de saida na mesma posicao da planilha de entrada

  demonstrativoSaida = planilhaSaida.add_sheet(demonstrativo.name)
  for curr_col in range(0,20): #coluna 0 (titulos) ate a 19 (dez 2012)
   for curr_row in range(demonstrativo.nrows):
    if curr_col > 0 and curr_row > 1 and isinstance(demonstrativo.cell_value
        (curr_row,curr_col),float): # curr_col > 0 nao esta nas colunas dos
        titulos, curr_row > 1 nao esta nas linhas das datas ou do titulo da
        planilha. Logo esta em algum valor numerico
     valor = demonstrativo.cell_value(curr_row,curr_col)/deflator.cell_value
         (curr_col,1) # A coluna do balanco eh equivalente a linha do
         deflator
    else:
     valor = demonstrativo.cell_value(curr_row,curr_col)
    demonstrativoSaida.write(curr_row,curr_col,valor) # coloca o valor da
        planilha de saida na mesma posicao da planilha de entrada


  planilhaSaida.save("balanco_deflacionado/"+nomeEmpresa+".xls")


# Cria diretorio caso na exista
helper.create_directory('balanco_deflacionado/')
empresas = trimestre.empresas_csv() #retorna a lista de nomes da empresa
    com a extensao .csv
for empresa in empresas:
 nomeEmpresa = empresa.replace(".csv","") # tira o .csv do nome
 if helper.exists_directory('balanco_deflacionado/'+ nomeEmpresa + '.xls'):
  print nomeEmpresa + ' balanco ja deflacionado!'
  continue
 print("Deflaciona balanco " + nomeEmpresa + " Comecou!")
 planilhasCreate(nomeEmpresa)
 print("balanco_deflacionado/" + nomeEmpresa + ".xls Criado!" )
```

I.2 deflaciona_cotacao.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import codecs # evitar unicode erro
import sys # evitar unicode erro
import xlrd
import trimestre
import pandas as pd
import helper
UTF8Writer = codecs.getwriter('utf8') # evitar unicode erro
sys.stdout = UTF8Writer(sys.stdout) # evitar unicode erro


def planilhasCreate(nomeEmpresa):
 igpm = xlrd.open_workbook("IGPM/inflacao.xlsx")
 deflator = igpm.sheet_by_index(2) # pega a terceira planilha de inflacao
 cotacao = pd.read_csv("cotacao/"+nomeEmpresa+".csv",index_col=0,
     parse_dates=True) # carrega aquivo csv de
 #print cotacao.head()
 row_index = 1 #indice da linha equivalente a primeira cotacao (01/02/2014
     - usado no calculo de 31/01/2014)
 for row_title, row in cotacao.iterrows():
  for col_title, val in row.items():
   cotacao.loc[row_title,col_title] = val/deflator.cell_value(row_index,1)
  row_index+=1 # Proxima cotacao

 cotacao.to_csv('cotacao_deflacionada/'+nomeEmpresa+'.csv')

# Cria diretorio caso na exista
helper.create_directory('cotacao_deflacionada/')
empresas = helper.empresas_csv() #retorna a lista de nomes da empresa com a
     extensao .csv
for empresa in empresas:
 if helper.exists_directory('cotacao_deflacionada/'+empresa):
  print empresa + ' cotacao ja deflacionada!'
  continue
 nomeEmpresa = empresa.replace(".csv","") # tira o .csv do nome
 print("Deflaciona cotacao " + nomeEmpresa + " Comecou!")
 planilhasCreate(nomeEmpresa)
 print("cotacao_deflacionada/" + nomeEmpresa + ".csv Criado!" )
```

I.3 deflaciona_cotacao_diaria.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import codecs # evitar unicode erro
import sys # evitar unicode erro
import xlrd
import trimestre
import pandas as pd
import helper
UTF8Writer = codecs.getwriter('utf8') # evitar unicode erro
sys.stdout = UTF8Writer(sys.stdout) # evitar unicode erro

def planilhasCreate(nomeEmpresa):
 igpm = xlrd.open_workbook("IGPM/inflacao.xlsx")
 deflator = igpm.sheet_by_index(4) # pega a quinta planilha de inflacao (
     deflator cotacao diaria)

 cotacao = pd.read_csv("cotacao_diaria/dentro_periodo/"+nomeEmpresa+".csv",
     index_col=0, parse_dates=True) # carrega aquivo csv de
 row_index = 1 #indice da linha equivalente a primeira cotacao (01/02/2014
     - usado no calculo de 31/01/2014)
 for row_title, row in cotacao.iterrows():
  for col_title, val in row.items():
   #print cotacao.loc[row_title, col_title]
   #print deflator.cell_value(row_index,1)
   if val is not None:
    cotacao.loc[row_title,col_title] = val/deflator.cell_value(row_index,1)
   else:
    cotacao.loc[row_title,col_title] = val

  row_index+=1 # Proxima cotacao

 cotacao.to_csv('cotacao_diaria_deflacionada/'+nomeEmpresa+'.csv')


# Cria diretorio caso na exista
helper.create_directory('cotacao_diaria_deflacionada/')
#empresas = pd.read_csv("bovespa_empresas_lista.csv", sep=';', decimal=",")
empresas = helper.empresas_cotacao_diaria_csv()
#for empresa in empresas.iterrows():
for empresa in empresas:
 nomeEmpresa = empresa.replace(".csv","") # tira o .csv do nome
 #nomeEmpresa = empresa[1][0] # pega o nome
```

```
 if helper.exists_directory('cotacao_diaria_deflacionada/'+nomeEmpresa+'.
    csv'):
  print nomeEmpresa + ' cotacao diaria ja deflacionada!'
  continue
 print("Deflaciona cotacao " + nomeEmpresa + " Comecou!")
 planilhasCreate(nomeEmpresa)
 print("cotacao_diaria_deflacionada/" + nomeEmpresa + ".csv Criado!" )
```

I.4 download_cotacao.py

```
import re
import requests
import sys
from pdb import set_trace as pb
import helper
import pandas as pd


empresas = pd.read_csv("bovespa_empresas_lista.csv", sep=';', decimal=",")
for empresa in empresas.iterrows():
 print empresa[1][0]
 if helper.exists_directory('cotacao_diaria/'+empresa[1][0]+'.csv') or
    helper.exists_directory('cotacao_diaria/cotacoes_fora_periodo/'+empresa
    [1][0]+'.csv'):
  print empresa[1][0] + ' cotacao diaria ja baixada!'
  continue

 symbol = empresa[1][0] + '.SA'
 #symbol = 'LLIS3.SA'
 start_date = '1388541600' # start date timestamp
 end_date   = '1546221600' # end date timestamp

 crumble_link = 'https://finance.yahoo.com/quote/{0}/history?p={0}'
 crumble_regex = r'CrumbStore":{"crumb":"(.*?)"}'
 cookie_regex = r'set-cookie: (.*?);'
 quote_link = 'https://query1.finance.yahoo.com/v7/finance/download/{}?
    period1={}&period2={}&interval=1d&events=history&crumb={}'

 link = crumble_link.format(symbol)
 session = requests.Session()
 response = session.get(link)

 # get crumbs
```

```
31
    text = str(response.content)
33  match = re.search(crumble_regex, text)
    crumbs = match.group(1)
35
    # get cookie
37
    cookie = session.cookies.get_dict()
39
    url = "https://query1.finance.yahoo.com/v7/finance/download/%s?period1=%s&
        period2=%s&interval=1d&events=history&crumb=%s" % (symbol, start_date,
        end_date, crumbs)
41
    r = requests.get(url,cookies=session.cookies.get_dict(),timeout=5, stream=
        True)
43
    out = r.text
45
    filename = 'cotacao_diaria/{}.csv'.format(empresa[1][0])
47
    with open(filename,'w') as f:
49      f.write(out)
```

I.5 empresa_dados_mes_ano_manipula_cotacao_mes.py

```
1  import pandas as pd
   import openpyxl
3  import matplotlib.pyplot as plt
   import seaborn as sns
5  import xlsxwriter
   import helper
7  import numpy as np
   # encoding=utf8
9  import sys
   reload(sys)  #
11 sys.setdefaultencoding('utf8')

13 def percentualAumento(atual,futuro):
    if np.isnan(futuro):
15   return futuro
    aumento = futuro - atual
```

```
17  return aumento/futuro

19 def planilhasCreate(nomeEmpresa):
    empresa = pd.read_csv("empresa_dados_mes_ano_csv/"+nomeEmpresa+".csv",
        index_col=0, sep=',', decimal=".") # carrega aquivo csv de
21
    desloca_cotacao = empresa.copy()
23  desloca_cotacao3 = empresa.copy()
    desloca_cotacao6 = empresa.copy()
25  desloca_cotacao9 = empresa.copy()
    desloca_cotacao12 = empresa.copy()
27  percentual_aumento_cotacao = empresa.copy()
    percentual_aumento_cotacao3 = empresa.copy()
29  percentual_aumento_cotacao6 = empresa.copy()
    percentual_aumento_cotacao9 = empresa.copy()
31  percentual_aumento_cotacao12 = empresa.copy()
    i = 41;
33  for row_title, row in empresa.iterrows():
     if i - 1 >= 0:
35    desloca_cotacao.ix[i,'cotacao'] = empresa.iloc[i-1]['cotacao']
      percentual_aumento_cotacao.ix[i,'cotacao'] = percentualAumento(empresa.
          iloc[i]['cotacao'],empresa.iloc[i-1]['cotacao'])
37   else:
      desloca_cotacao.ix[i,'cotacao'] = None
39    percentual_aumento_cotacao.ix[i,'cotacao'] = None

41   if i - 3 >= 0:
      desloca_cotacao3.ix[i,'cotacao'] = empresa.iloc[i-3]['cotacao']
43    percentual_aumento_cotacao3.ix[i,'cotacao'] = percentualAumento(empresa.
          iloc[i]['cotacao'],empresa.iloc[i-3]['cotacao']) # O anterior eh i+1
          pois a planilha esta indo da maior data para menor, logo a data
          anterior esta na proxima linha
     else:
45    desloca_cotacao3.ix[i,'cotacao'] = None
      percentual_aumento_cotacao3.ix[i,'cotacao'] = None
47
     if i - 6 >= 0:
49    desloca_cotacao6.ix[i,'cotacao'] = empresa.iloc[i-6]['cotacao']
      percentual_aumento_cotacao6.ix[i,'cotacao'] = percentualAumento(empresa.
          iloc[i]['cotacao'],empresa.iloc[i-6]['cotacao']) # O anterior eh i+1
          pois a planilha esta indo da maior data para menor, logo a data
          anterior esta na proxima linha
51   else:
      desloca_cotacao6.ix[i,'cotacao'] = None
53    percentual_aumento_cotacao6.ix[i,'cotacao'] = None
```

```
    if i - 9 >= 0:
     desloca_cotacao9.ix[i,'cotacao'] = empresa.iloc[i-9]['cotacao']
     percentual_aumento_cotacao9.ix[i,'cotacao'] = percentualAumento(empresa.
         iloc[i]['cotacao'],empresa.iloc[i-9]['cotacao']) # O anterior eh i+1
         pois a planilha esta indo da maior data para menor, logo a data
         anterior esta na proxima linha
    else:
     desloca_cotacao9.ix[i,'cotacao'] = None
     percentual_aumento_cotacao9.ix[i,'cotacao'] = None

    if i - 12 >= 0:
     desloca_cotacao12.ix[i,'cotacao'] = empresa.iloc[i-12]['cotacao']
     percentual_aumento_cotacao12.ix[i,'cotacao'] = percentualAumento(empresa
         .iloc[i]['cotacao'],empresa.iloc[i-12]['cotacao']) # O anterior eh i
         +1 pois a planilha esta indo da maior data para menor, logo a data
         anterior esta na proxima linha
    else:
     desloca_cotacao12.ix[i,'cotacao'] = None
     percentual_aumento_cotacao12.ix[i,'cotacao'] = None

    i-=1

desloca_cotacao = desloca_cotacao.dropna(axis=0, how='any') # exclui
    qualquer linha que tenha valor nulo
desloca_cotacao3 = desloca_cotacao3.dropna(axis=0, how='any') # exclui
    qualquer linha que tenha valor nulo
desloca_cotacao6 = desloca_cotacao6.dropna(axis=0, how='any') # exclui
    qualquer linha que tenha valor nulo
desloca_cotacao9 = desloca_cotacao9.dropna(axis=0, how='any') # exclui
    qualquer linha que tenha valor nulo
desloca_cotacao12 = desloca_cotacao12.dropna(axis=0, how='any') # exclui
    qualquer linha que tenha valor nulo
desloca_cotacao.to_csv('empresa_dados_mes_ano_csv/cotacao_deslocada_1_mes/
    '+nomeEmpresa+'_deslocada1.csv', encoding='utf-8', sep=',', decimal="."
    )
desloca_cotacao3.to_csv('empresa_dados_mes_ano_csv/
    cotacao_deslocada_3_meses/'+nomeEmpresa+'_deslocada3.csv', encoding='
    utf-8', sep=',', decimal=".")
# desloca_cotacao6 esta com VIRGULA ao inves de ponto
desloca_cotacao6.to_csv('empresa_dados_mes_ano_csv/
    cotacao_deslocada_6_meses/'+nomeEmpresa+'_deslocada6.csv', encoding='
    utf-8', sep=',', decimal=".")
```

```
desloca_cotacao9.to_csv('empresa_dados_mes_ano_csv/
    cotacao_deslocada_9_meses/'+nomeEmpresa+'_deslocada9.csv', encoding='
    utf-8', sep=',', decimal=".")
desloca_cotacao12.to_csv('empresa_dados_mes_ano_csv/
    cotacao_deslocada_12_meses/'+nomeEmpresa+'_deslocada12.csv', encoding='
    utf-8', sep=',', decimal=".")
percentual_aumento_cotacao = percentual_aumento_cotacao.dropna(axis=0, how
    ='any') # exclui qualquer linha que tenha valor nulo
percentual_aumento_cotacao3 = percentual_aumento_cotacao3.dropna(axis=0,
    how='any') # exclui qualquer linha que tenha valor nulo
percentual_aumento_cotacao6 = percentual_aumento_cotacao6.dropna(axis=0,
    how='any') # exclui qualquer linha que tenha valor nulo
percentual_aumento_cotacao9 = percentual_aumento_cotacao9.dropna(axis=0,
    how='any') # exclui qualquer linha que tenha valor nulo
percentual_aumento_cotacao12 = percentual_aumento_cotacao12.dropna(axis=0,
     how='any') # exclui qualquer linha que tenha valor nulo
percentual_aumento_cotacao.to_csv('empresa_dados_mes_ano_csv/
    cotacao_percentual_aumento1/'+nomeEmpresa+'_percentual_aumento1.csv',
    encoding='utf-8', sep=',', decimal=".")
percentual_aumento_cotacao3.to_csv('empresa_dados_mes_ano_csv/
    cotacao_percentual_aumento3/'+nomeEmpresa+'_percentual_aumento3.csv',
    encoding='utf-8', sep=',', decimal=".")
percentual_aumento_cotacao6.to_csv('empresa_dados_mes_ano_csv/
    cotacao_percentual_aumento6/'+nomeEmpresa+'_percentual_aumento6.csv',
    encoding='utf-8', sep=',', decimal=".")
percentual_aumento_cotacao9.to_csv('empresa_dados_mes_ano_csv/
    cotacao_percentual_aumento9/'+nomeEmpresa+'_percentual_aumento9.csv',
    encoding='utf-8', sep=',', decimal=".")
percentual_aumento_cotacao12.to_csv('empresa_dados_mes_ano_csv/
    cotacao_percentual_aumento12/'+nomeEmpresa+'_percentual_aumento12.csv',
     encoding='utf-8', sep=',', decimal=".")



helper.create_directory('empresa_dados_mes_ano_csv')
helper.create_directory('empresa_dados_mes_ano_csv/cotacao_deslocada_1_mes'
    )
helper.create_directory('empresa_dados_mes_ano_csv/
    cotacao_deslocada_3_meses')
helper.create_directory('empresa_dados_mes_ano_csv/
    cotacao_deslocada_6_meses')
helper.create_directory('empresa_dados_mes_ano_csv/
    cotacao_deslocada_9_meses')
```

```
helper.create_directory('empresa_dados_mes_ano_csv/
    cotacao_deslocada_12_meses')
helper.create_directory('empresa_dados_mes_ano_csv/
    cotacao_percentual_aumento1')
helper.create_directory('empresa_dados_mes_ano_csv/
    cotacao_percentual_aumento3')
helper.create_directory('empresa_dados_mes_ano_csv/
    cotacao_percentual_aumento6')
helper.create_directory('empresa_dados_mes_ano_csv/
    cotacao_percentual_aumento9')
helper.create_directory('empresa_dados_mes_ano_csv/
    cotacao_percentual_aumento12')
empresas = helper.empresas_csv() #retorna a lista de nomes da empresa com a
     extensao .csv
for empresa in empresas:
 nomeEmpresa = empresa.replace(".csv","") # tira o .csv do nome
 print("Processa empresa_dados_mes_ano " + nomeEmpresa + " Comecou!")
 planilhasCreate(nomeEmpresa)
 print("empresa_dados_mes_ano_csv/cotacao_deslocada_1_mes/" + nomeEmpresa +
     ".csv Criado!" )
print("Sucesso!")
```

I.6 empresa_dados_mes_ano_trimestral_manipula_cotacao_mes.py

```
import pandas as pd
import openpyxl
import matplotlib.pyplot as plt
import seaborn as sns
import xlsxwriter
import helper
import numpy as np
# encoding=utf8
import sys
reload(sys)  #
sys.setdefaultencoding('utf8')

def percentualAumento(atual,futuro):
 if np.isnan(futuro):
  return futuro
 aumento = futuro - atual
 return aumento/futuro
```

```
18
def planilhasCreate(nomeEmpresa):
20  empresa = pd.read_csv("empresa_dados_mes_ano_csv_trimestral/"+nomeEmpresa+
        "_trimestre.csv", index_col=0, sep=',', decimal=".") # carrega aquivo
        csv de

22  desloca_cotacao = empresa.copy()
    desloca_cotacao2 = empresa.copy()
24  percentual_aumento_cotacao = empresa.copy()
    percentual_aumento_cotacao2 = empresa.copy()
26  i = 13;
    for row_title, row in empresa.iterrows():
28   if i - 1 >= 0:
      desloca_cotacao.ix[i,'cotacao'] = empresa.iloc[i-1]['cotacao']
30    percentual_aumento_cotacao.ix[i,'cotacao'] = percentualAumento(empresa.
          iloc[i]['cotacao'],empresa.iloc[i-1]['cotacao']) # O anterior eh i+1
          pois a planilha esta indo da maior data para menor, logo a data
          anterior esta na proxima linha
      i-=1
32   else:
      desloca_cotacao.ix[i,'cotacao'] = None
34    break
    desloca_cotacao = desloca_cotacao.dropna(axis=0, how='any') # exclui
        qualquer linha que tenha valor nulo
36  desloca_cotacao.to_csv('empresa_dados_mes_ano_csv_trimestral/
        cotacao_deslocada_1_trimestre/'+nomeEmpresa+'_trimestre_deslocada1.csv'
        , encoding='utf-8', sep=',', decimal=".")
    percentual_aumento_cotacao = percentual_aumento_cotacao.dropna(axis=0, how
        ='any') # exclui qualquer linha que tenha valor nulo
38  percentual_aumento_cotacao.to_csv('empresa_dados_mes_ano_csv_trimestral/
        cotacao_percentual_aumento1/'+nomeEmpresa+'
        _trimestre_percentual_aumento1.csv', encoding='utf-8', sep=',', decimal
        =".")



40



42

helper.create_directory('empresa_dados_mes_ano_csv_trimestral')
44 helper.create_directory('empresa_dados_mes_ano_csv_trimestral/
       cotacao_deslocada_1_trimestre')
   helper.create_directory('empresa_dados_mes_ano_csv_trimestral/
       cotacao_percentual_aumento1')
46 empresas = helper.empresas_csv() #retorna a lista de nomes da empresa com a
       extensao .csv
   for empresa in empresas:
```

```
#if helper.exists_directory('empresa_dados_mes_ano_csv_trimestral/
    cotacao_deslocada_um_mes/'+empresa):
 # print empresa + ' ja processada!'
 # continue
 nomeEmpresa = empresa.replace(".csv","") # tira o .csv do nome
 print("Processa empresa_dados_mes_ano " + nomeEmpresa + "_trimestre
    Comecou!")
 planilhasCreate(nomeEmpresa)
 print("empresa_dados_mes_ano_csv_trimestral/cotacao_deslocada_1_mes/" +
    nomeEmpresa + "_trimestre.csv Criado!" )
print("Sucesso!")
```

I.7 gerar_correlacao.py

```
import pandas as pd
import openpyxl
import matplotlib.pyplot as plt
import seaborn as sns
import xlsxwriter
import helper

# encoding=utf8
import sys
reload(sys)  #
sys.setdefaultencoding('utf8')

def gerarCorrelacaoCSV(nomeEmpresa):
 empresa = pd.read_csv("empresa_dados_mes_ano_csv/"+nomeEmpresa+".csv",
    index_col=0, parse_dates=True, sep=';', decimal=",") # carrega aquivo
    csv de

 corr = empresa.corr()

 name_sheet= 'Sheet1'

 writer = pd.ExcelWriter('correlacao/'+nomeEmpresa+'.xlsx',engine='
    xlsxwriter')
 corr.to_excel(writer,sheet_name=name_sheet)
 workbook = writer.book
 worksheet = writer.sheets[name_sheet]

 worksheet.conditional_format('A1:DP120', {'type': '3_color_scale'})
 writer.save()
```

```
27  corr.to_csv('correlacao_csv/'+nomeEmpresa+'.csv', encoding='utf-8', sep=';
        ', decimal=",")

29# Cria diretorio caso na exista
helper.create_directory('correlacao/')
31helper.create_directory('correlacao_csv/')
gerarCorrelacaoCSV('abev3')
33#empresas = trimestre.empresas_csv() #retorna a lista de nomes da empresa
    com a extensao .csv
```

I.8 gerar_correlacao_colunas_selecionadas.py

```
import pandas as pd
2import openpyxl
import matplotlib.pyplot as plt
4import seaborn as sns
import xlsxwriter
6import helper

8# encoding=utf8
import sys
10reload(sys)  #
sys.setdefaultencoding('utf8')
12
def gerarCorrelacaoCSV(nomeEmpresa):
14 # carrega aquivo csv de
 empresa = pd.read_csv("empresa_dados_mes_ano_csv/"+nomeEmpresa+".csv",
     index_col=0, sep=',', decimal=".")
16
 empresa = pd.DataFrame(empresa, columns=['cotacao','expectativa_renda','
     Juro Real','ibovespa','taxa_de_desocupacao_media','
     indicadores_da_conjuntura_economica_salario_real_na_industria_de_transformacao
     ','
     indicadores_da_conjuntura_economica_horas_trabalhadas_na_producao_na_industria_de_t
     ','indicadores_da_conjuntura_economica_vendas_industriais_reais','
     indicadores_de_nivel_de_emprego_formal_construcao_civil','
     indicadores_de_nivel_de_emprego_formal_construcao_civil','
     indicadores_de_nivel_de_emprego_formal_servico','
     indicadores_de_nivel_de_emprego_formal_comercio'])
18 corr = empresa.corr()

20 name_sheet= 'correlacao'
```

```
22  writer = pd.ExcelWriter('correlacao_colunas_selecionadas/'+nomeEmpresa+'
        _correlacao.xlsx',engine='xlsxwriter')
    corr.to_excel(writer,sheet_name=name_sheet)
24  workbook = writer.book
    worksheet = writer.sheets[name_sheet]
26
    worksheet.conditional_format('A1:D4', {'type': '3_color_scale'})
28  writer.save()
    corr.to_csv('correlacao_colunas_selecionadas_csv/'+nomeEmpresa+'
        _correlacao.csv', encoding='utf-8', sep=';', decimal=",")
30
# Cria diretorio caso na exista
32 helper.create_directory('correlacao_colunas_selecionadas/')
   helper.create_directory('correlacao_colunas_selecionadas_csv/')
34
   gerarCorrelacaoCSV('jbss3')
```

I.9 gerar_correlacao_diaria.py

```
import helper
2 import pandas as pd
  import numpy as np
4 import xlsxwriter
# encoding=utf8
6 import sys
  reload(sys)  #
8 sys.setdefaultencoding('utf8')

10 def correlationWithDate(nomeEmpresa1,nomeEmpresa2):
    empresa_dados1 = pd.read_csv("cotacao_diaria_deflacionada/"+nomeEmpresa1+"
        .csv",index_col=0, parse_dates=True) # carrega aquivo csv de
12  empresa_dados2 = pd.read_csv("cotacao_diaria_deflacionada/"+nomeEmpresa2+"
        .csv",index_col=0, parse_dates=True) # carrega aquivo csv de
    cotacao_empresa1 = empresa_dados1['Close'] # retorna um objeto do tipo
        series
14  cotacao_empresa2 = empresa_dados2['Close']

16  correlacao = cotacao_empresa1.to_frame().corrwith(cotacao_empresa2.
        to_frame(),axis='index') # to_frame() transforma um objeto series em um
         dataframe

18  return correlacao['Close']
```

```
# Cria diretorio caso na exista
helper.create_directory('correlacao_cotacao_diaria/')
helper.create_directory('correlacao_cotacao_diaria_csv/')

correlacao = pd.DataFrame()

empresas = helper.empresas_cotacao_diaria_csv() #retorna a lista de nomes
    da empresa com a extensao .csv
empresas2 = list(empresas) # cria copia da lista
index = 0;
for empresa1 in empresas:
 empresas2.remove(empresa1)
 nomeEmpresa1 = empresa1.replace(".csv","") # tira o .csv do nome
 for empresa2 in empresas2:
  nomeEmpresa2 = empresa2.replace(".csv","") # tira o .csv do nome
  corr = correlationWithDate(nomeEmpresa1,nomeEmpresa2)
  correlacao.loc[nomeEmpresa1 + "_X_" + nomeEmpresa2, 1] = corr
  print "END: " + nomeEmpresa1 + "_X_" + nomeEmpresa2
  index += 1

correlacao.loc['', 1] = -1
#correlacao = correlacao.sort_index(axis=1, inplace=True)

name_sheet= 'Sheet1'
writer = pd.ExcelWriter('correlacao_cotacao_diaria/
    correlacao_cotacao_diaria.xlsx',engine='xlsxwriter')
correlacao.to_excel(writer,sheet_name=name_sheet)
workbook = writer.book
worksheet = writer.sheets[name_sheet]

worksheet.conditional_format('A1:B20303', {'type': '3_color_scale'})
writer.save()
correlacao.to_csv('correlacao_cotacao_diaria_csv/correlacao_cotacao_diaria.
    csv', encoding='utf-8', sep=';', decimal=",")
```

I.10 gerar_correlacao_diaria_grupos.py

```
import helper
import pandas as pd
import numpy as np
import xlsxwriter
# encoding=utf8
import sys
```

```
reload(sys)  #
sys.setdefaultencoding('utf8')

def correlationWithDate(grupo, nomeEmpresa1,nomeEmpresa2):
 empresa_dados1 = pd.read_csv("cotacao_diaria_deflacionada/grupos/"+ grupo
     + "/" + nomeEmpresa1 +".csv",index_col=0, parse_dates=True) # carrega
     aquivo csv de
 empresa_dados2 = pd.read_csv("cotacao_diaria_deflacionada/grupos/"+ grupo
     + "/" + nomeEmpresa2 +".csv",index_col=0, parse_dates=True) # carrega
     aquivo csv de
 cotacao_empresa1 = empresa_dados1['Close'] # retorna um objeto do tipo
     series
 cotacao_empresa2 = empresa_dados2['Close']

 correlacao = cotacao_empresa1.to_frame().corrwith(cotacao_empresa2.
     to_frame(),axis='index') # to_frame() transforma um objeto series em um
      dataframe
 print correlacao.head()
 return correlacao['Close']

# Cria diretorio caso na exista
helper.create_directory('correlacao_cotacao_diaria/')
helper.create_directory('correlacao_cotacao_diaria/grupos')
helper.create_directory('correlacao_cotacao_diaria_csv/')
helper.create_directory('correlacao_cotacao_diaria_csv/grupos')

for grupo in helper.grupos_empresa_cotacao_diaria():
 correlacao = pd.DataFrame()
 #correlacao.loc["Empresas", 1] = "Correlacao"
 empresas = helper.empresas_grupo_cotacao_diaria(grupo) #retorna a lista de
      nomes da empresa com a extensao .csv
 empresas2 = list(empresas) # cria copia da lista
 index = 0;
 helper.create_directory('correlacao_cotacao_diaria/grupos/'+grupo)
 helper.create_directory('correlacao_cotacao_diaria_csv/grupos/'+grupo)
 for empresa1 in empresas:
  empresas2.remove(empresa1)
  nomeEmpresa1 = empresa1.replace(".csv","") # tira o .csv do nome
  for empresa2 in empresas2:
   nomeEmpresa2 = empresa2.replace(".csv","") # tira o .csv do nome
   corr = correlationWithDate(grupo,nomeEmpresa1,nomeEmpresa2)
   print corr
   correlacao.loc[nomeEmpresa1 + "_X_" + nomeEmpresa2, 1] = corr
   print "END: " + nomeEmpresa1 + "_X_" + nomeEmpresa2
   index += 1
```

```
44
    name_sheet= 'Sheet1'
46  writer = pd.ExcelWriter('correlacao_cotacao_diaria/grupos/'+ grupo + '/
        correlacao_cotacao_diaria_'+ grupo +'.xlsx',engine='xlsxwriter')
    correlacao.to_excel(writer,sheet_name=name_sheet)
48  workbook = writer.book
    worksheet = writer.sheets[name_sheet]
50
    worksheet.conditional_format('A1:B661', {'type': '3_color_scale'})
52  writer.save()
    correlacao.to_csv('correlacao_cotacao_diaria_csv/grupos/'+ grupo + '/
        correlacao_cotacao_diaria_' + grupo + '.csv', encoding='utf-8', sep=';'
        , decimal=",")
```

I.11 gerar_empresa_dados_mes_ano.py

```
1  #!/usr/bin/env python
   # -*- coding: utf-8 -*-
3  import codecs # evitar unicode erro
   import sys # evitar unicode erro
5  import trimestre,sql
   import xlwt
7  import xlrd
   import numpy as np
9  import pandas as pd
   from datetime import datetime
11 from dateutil.relativedelta import relativedelta
   import helper
13
   UTF8Writer = codecs.getwriter('utf8') # evitar unicode erro
15 sys.stdout = UTF8Writer(sys.stdout) # evitar unicode erro

17 def last_day_of_month(any_day):
       next_month = any_day.replace(day=28) + relativedelta(days=4)  # this
           will never fail
19     return next_month - relativedelta(days=next_month.day)

21 def replaceVirgulaToPonto(palavra):
    if isinstance(palavra,float):
23   return str(palavra)
    return palavra
25
   def toFloat(value):
```

```
27  if isinstance(value, basestring) or np.isnan(value):
     return value
29  return str(float(value))

31 def writeInvertido(sheet,col,row,value):
    if col == 0:
33   sheet.write(col, row-1, value)
    # se for na linha da data
35  elif row == 1:
     col = col * 3
37   ultimo_mes_trimestre = datetime.strptime(value, '%d/%m/%Y').date()
     primeiro_mes_trimestre = ultimo_mes_trimestre - relativedelta(months=2)
39   segundo_mes_trimestre = ultimo_mes_trimestre - relativedelta(months=1)
     primeiro_mes_trimestre = last_day_of_month(primeiro_mes_trimestre)
41   segundo_mes_trimestre = last_day_of_month(segundo_mes_trimestre)
     sheet.write(col, row-1, primeiro_mes_trimestre.strftime('%d/%m/%Y'))
43   sheet.write(col-1, row-1, segundo_mes_trimestre.strftime('%d/%m/%Y'))
     sheet.write(col-2, row-1, value)
45  else:
     col = col * 3
47   sheet.write(col-2, row-1, toFloat(value))
     sheet.write(col-1, row-1, toFloat(value))
49   sheet.write(col, row-1, toFloat(value))

51 def indicadores(i,sheet,row,col,col_data,cotacao,quantidadeAcoes,
       nomeEmpresa):
    variaveis1 = trimestre.Variaveis(float(cotacao.iloc[row,4]),
        quantidadeAcoes,nomeEmpresa)
53  variaveis2 = trimestre.Variaveis(float(cotacao.iloc[row+1,4]),
        quantidadeAcoes,nomeEmpresa)
    variaveis3 = trimestre.Variaveis(float(cotacao.iloc[row+2,4]),
        quantidadeAcoes,nomeEmpresa)
55  # P/L
    if i == 0:
57   sheet.write(row+1, col, variaveis1.getPrecoAcaoPorLucro(col_data))
     sheet.write(row+2, col, variaveis2.getPrecoAcaoPorLucro(col_data))
59   sheet.write(row+3, col, variaveis3.getPrecoAcaoPorLucro(col_data))
    # P/VP
61  elif i == 1:
     sheet.write(row+1, col, variaveis1.
         getPrecoAcaoPorPatrimonioLiquidoPorAcao(col_data))
63   sheet.write(row+2, col, variaveis2.
         getPrecoAcaoPorPatrimonioLiquidoPorAcao(col_data))
     sheet.write(row+3, col, variaveis3.
         getPrecoAcaoPorPatrimonioLiquidoPorAcao(col_data))
```

```
65  # P/EBIT
    elif i == 2:
67   sheet.write(row+1, col, variaveis1.
         getPrecoAcaoPorEBITUltimo12mesesPorAcao(col_data))
     sheet.write(row+2, col, variaveis2.
         getPrecoAcaoPorEBITUltimo12mesesPorAcao(col_data))
69   sheet.write(row+3, col, variaveis3.
         getPrecoAcaoPorEBITUltimo12mesesPorAcao(col_data))
    # PSR
71  elif i == 3:
     sheet.write(row+1, col, variaveis1.getPrecoAcaoPorReceitaLiquidaPorAcao(
         col_data))
73   sheet.write(row+2, col, variaveis2.getPrecoAcaoPorReceitaLiquidaPorAcao(
         col_data))
     sheet.write(row+3, col, variaveis3.getPrecoAcaoPorReceitaLiquidaPorAcao(
         col_data))
75  # P/Ativos
    elif i == 4:
77   sheet.write(row+1, col, variaveis1.getPrecoAcaoPorAtivosTotais(col_data))
     sheet.write(row+2, col, variaveis2.getPrecoAcaoPorAtivosTotais(col_data))
79   sheet.write(row+3, col, variaveis3.getPrecoAcaoPorAtivosTotais(col_data))
    # P/Cap. Giro
81  elif i == 5:
     sheet.write(row+1, col, variaveis1.getPrecoAcaoPorCapitalGiroPorAcao(
         col_data))
83   sheet.write(row+2, col, variaveis2.getPrecoAcaoPorCapitalGiroPorAcao(
         col_data))
     sheet.write(row+3, col, variaveis3.getPrecoAcaoPorCapitalGiroPorAcao(
         col_data))
85  # P/Ativ Circ Liq
    elif i == 6:
87   sheet.write(row+1, col, variaveis1.
         getPrecoAcaoPorAtivoCirculanteLiquidoPorAcao(col_data))
     sheet.write(row+2, col, variaveis2.
         getPrecoAcaoPorAtivoCirculanteLiquidoPorAcao(col_data))
89   sheet.write(row+3, col, variaveis3.
         getPrecoAcaoPorAtivoCirculanteLiquidoPorAcao(col_data))
    # Div. Yield
91  elif i == 7:
     sheet.write(row+1, col, variaveis1.getDividendoYield(col_data))
93   sheet.write(row+2, col, variaveis2.getDividendoYield(col_data))
     sheet.write(row+3, col, variaveis3.getDividendoYield(col_data))
95  # EV/EBIT
    elif i == 8:
```

```
97    sheet.write(row+1, col, variaveis1.getValorDaFirmaPorEBITUltimos12meses(
          col_data))
      sheet.write(row+2, col, variaveis2.getValorDaFirmaPorEBITUltimos12meses(
          col_data))
99    sheet.write(row+3, col, variaveis3.getValorDaFirmaPorEBITUltimos12meses(
          col_data))
    # Giros Ativos
101 elif i == 9:
      sheet.write(row+1, col, variaveis1.getGirosAtivos(col_data))
103   sheet.write(row+2, col, variaveis2.getGirosAtivos(col_data))
      sheet.write(row+3, col, variaveis3.getGirosAtivos(col_data))
105 # LP (Lucro por acao Ultimos 12 meses)
    elif i == 10:
107   sheet.write(row+1, col, variaveis1.getLucroPorAcao(col_data))
      sheet.write(row+2, col, variaveis2.getLucroPorAcao(col_data))
109   sheet.write(row+3, col, variaveis3.getLucroPorAcao(col_data))
    # VPA (Patrimonio Liquido por acao)
111 elif i == 11:
      sheet.write(row+1, col, variaveis1.getPatrimonioLiquidoPorAcao(col_data))
113   sheet.write(row+2, col, variaveis2.getPatrimonioLiquidoPorAcao(col_data))
      sheet.write(row+3, col, variaveis3.getPatrimonioLiquidoPorAcao(col_data))
115 # Marg. Bruta (Lucro Bruto/ Receita Liquida) - ultimos 12 meses
    elif i == 12:
117   sheet.write(row+1, col, variaveis1.getMargBruta(col_data))
      sheet.write(row+2, col, variaveis2.getMargBruta(col_data))
119   sheet.write(row+3, col, variaveis3.getMargBruta(col_data))
    # Marg. Ebit (EBIT/Receita Liquida) - ultimos 12 meses
121 elif i == 13:
      sheet.write(row+1, col, variaveis1.getMargEBIT(col_data))
123   sheet.write(row+2, col, variaveis2.getMargEBIT(col_data))
      sheet.write(row+3, col, variaveis3.getMargEBIT(col_data))
125 # Marg Liquida (Lucro Liquido/ Receita Liquida) ultimos 12 meses
    elif i == 14:
127   sheet.write(row+1, col, variaveis1.getMargLiquida(col_data))
      sheet.write(row+2, col, variaveis2.getMargLiquida(col_data))
129   sheet.write(row+3, col, variaveis3.getMargLiquida(col_data))
    # EBIT/ATIVO (EBIT ultimos 12 meses/ Ativos totais ultimo trimestre)
131 elif i == 15:
      sheet.write(row+1, col, variaveis1.getEBITporAtivos(col_data))
133   sheet.write(row+2, col, variaveis2.getEBITporAtivos(col_data))
      sheet.write(row+3, col, variaveis3.getEBITporAtivos(col_data))
135 # ROIC = EBIT ultimos 12 meses/ (Ativos Totais - caixa - fornecedores)
        ultimo trimestre
    elif i == 16:
137   sheet.write(row+1, col, variaveis1.getROIC(col_data))
```

```
            sheet.write(row+2, col, variaveis2.getROIC(col_data))
            sheet.write(row+3, col, variaveis3.getROIC(col_data))
        # ROE = Lucro Liquido ultimos 12 meses/ Patrimonio Liquido ultimo trimestre
        elif i == 17:
            sheet.write(row+1, col, variaveis1.getROE(col_data))
            sheet.write(row+2, col, variaveis2.getROE(col_data))
            sheet.write(row+3, col, variaveis3.getROE(col_data))
        # Liquidez Corr (Ativo Circulante/Passivo Circulante)
        elif i == 18:
            sheet.write(row+1, col, variaveis1.getLiquidezCorr(col_data))
            sheet.write(row+2, col, variaveis2.getLiquidezCorr(col_data))
            sheet.write(row+3, col, variaveis3.getLiquidezCorr(col_data))
        # Div Br / Patrim (Divida Bruta / Patrimonio Liquido)
        elif i == 19:
            sheet.write(row+1, col, variaveis1.getDividaBrutaPorPatrimonioLiquido(col_data))
            sheet.write(row+2, col, variaveis2.getDividaBrutaPorPatrimonioLiquido(col_data))
            sheet.write(row+3, col, variaveis3.getDividaBrutaPorPatrimonioLiquido(col_data))


def planilhasCreate(nomeEmpresa):
    planilhaBalanco = xlrd.open_workbook("balanco_deflacionado/"+nomeEmpresa+".xls") # Abre o arquivo do excel de balanco
    balanco = planilhaBalanco.sheet_by_index(0) # Pega a primeira planilha do excel de balanco
    demonstrativo = planilhaBalanco.sheet_by_index(1) # pega a segunda planilha do excel de balanco
    # Cotacao modelo novo
    cotacao = pd.read_csv("cotacao_deflacionada/"+nomeEmpresa+".csv") # carrega aquivo csv de cotacao
    cotacao = cotacao.sort_values(by=['Date'],ascending=False) # ordena valores de forma descrescente por data do csv de cotacao
    cotacao = cotacao.reset_index(drop=True) # resta os indices para que o primeiro indice depois da ordenacao seja 0 (na ordenacao ele preserva o indice)
    # End Cotacao modelo novo

    planilhaSaida = xlwt.Workbook(encoding = 'utf-8') # abre o arquivo do excel de saida

    sheet = planilhaSaida.add_sheet('dados_e_indicadores', cell_overwrite_ok=True) # adiciona uma planilha ao arquivo do excel de saida(Contem todos
```

```
    os indicadores, os dados do balanco e demonstrativo, indicadores do
    IBGE e contacao)

quantidadeAcoes = balanco.cell_value(1,0) # Pega a quantidade de acoes da
    empresa na segunda linha primeira coluna

# Inscreva na planilha de saida os balancos de forma invertida(linha vira
    coluna e coluna vira linha)
for curr_col in range(balanco.ncols):
 for curr_row in range(1,balanco.nrows):
  writeInvertido(sheet,curr_col,curr_row,balanco.cell_value(curr_row,
      curr_col))

# Inscreva na planilha de saida os demonstrativos de forma invertida(linha
    vira coluna e coluna vira linha)
for curr_col in range(demonstrativo.ncols):
 for curr_row in range(2,demonstrativo.nrows):
  row = curr_row + balanco.nrows - 2
  writeInvertido(sheet,curr_col,row,demonstrativo.cell_value(curr_row,
      curr_col))

# Cotacao Iterator
col_cotacao = balanco.nrows + demonstrativo.nrows - 3 #O numero 3 se
    refere a quantidade de linhas que sao descartadas durante a execu  o
    do script (linhas nao utilizadas da planilha)
sheet.write(0,col_cotacao,"cotacao") # Cabeca
for index, row in cotacao.iterrows():
 sheet.write(index+1,col_cotacao,row['Close'])

indicadores_cabecalho = ["P/L","P/VP","P/EBIT","PSR","P/Ativos","P/Cap.
    Giro","P/Ativ Circ Liq","Div. Yield","EV/EBIT","Giros Ativos","LP","VPA
    ","Marg. Bruta","Marg. EBIT","Marg. Liquida","EBIT/ATIVO","ROIC","ROE",
    "Liquidez Corr","Div br/Patrim"]

for i in range(len(indicadores_cabecalho)):
   indicador_nome = indicadores_cabecalho[i]
   col = i + balanco.nrows + demonstrativo.nrows - 2 #O n mero 2 se
       refere a quantidade de linhas que sao descartadas durante a
       execu  o do script (linhas nao utilizadas da planilha) + 1 (devido
        a coluna de cotacoes)
   sheet.write(0, col, indicador_nome)
   col_data = 0
   for curr_row in range(0,40,3):
    col_data+=1
```

```
199         indicadores(i,sheet,curr_row,col,col_data,cotacao,quantidadeAcoes,
                nomeEmpresa)

201     planilhas = ['expectativa_inflacao','expectativa_desemprego','
            expectativa_consumidor','expecativa_compras_bens_maior_valor','
            endividamento','expectativa_renda','indice_de_commodites_agropecuaria',
            'indice_de_commodites_metal','indice_de_commodites_energia', '
            indicadores_de_nivel_de_emprego_formal_industria_de_transformacao','
            indicadores_de_nivel_de_emprego_formal_comercio','
            indicadores_de_nivel_de_emprego_formal_servico','
            indicadores_de_nivel_de_emprego_formal_construcao_civil','
            indicadores_da_conjuntura_economica_vendas_industriais_reais','
            indicadores_da_conjuntura_economica_horas_trabalhadas_na_producao_na_industria_de_t
            ','
            indicadores_da_conjuntura_economica_emprego_na_industria_de_transformacao
            ','
            indicadores_da_conjuntura_economica_salario_real_na_industria_de_transformacao
            ','taxa_de_desocupacao_media']

203     for i in range(len(planilhas)):
         planilhaDados = xlrd.open_workbook("outros_indicadores/"+planilhas[i]+".
             xlsx")
205      dados = planilhaDados.sheet_by_index(0)
         col = i + balanco.nrows + demonstrativo.nrows + len(indicadores_cabecalho
             ) - 2 #O n mero 2 se refere a quantidade de linhas que sao
             descartadas durante a execu  o do script (linhas nao utilizadas da
             planilha) + 1 (devido a coluna de cotacoes)
207      sheet.write(0,col,planilhas[i])
         for curr_row in range(1,58):
209       sheet.write(curr_row,col,float(dados.cell_value(curr_row-1,1)))

211     igpm = xlrd.open_workbook("IGPM/inflacao.xlsx")
        igpmPlanilha = igpm.sheet_by_index(0) # pega a primeira planilha de
            inflacao
213     selic = xlrd.open_workbook("SELIC/selic.xlsx")
        selicPlanilha = selic.sheet_by_index(0)
215     col_juros_real = balanco.nrows + demonstrativo.nrows + len(
            indicadores_cabecalho) - 2 + len(planilhas) #O n mero 2 se refere a
            quantidade de linhas que sao descartadas durante a execu  o do script
             (linhas nao utilizadas da planilha) + 1 (devido a coluna de cotacoes)
        sheet.write(0,col_juros_real,"Juro Real")
217     for curr_row in range(1,56):
         igpm_mes = igpmPlanilha.cell_value(curr_row,2) # inflacao do mes
219      selic_mes = selicPlanilha.cell_value(curr_row,2) # juros selic mes
         juro_real = calcula_juro_real(igpm_mes,selic_mes)
```

```
    sheet.write(curr_row,col_juros_real,juro_real)

 # indice bovespa(ibovespa)
 deflator = igpm.sheet_by_index(2)
 ibovespa = pd.read_csv("ibovespa/bvsp.csv") # carrega aquivo csv de
 col_ibovespa = balanco.nrows + demonstrativo.nrows + len(
     indicadores_cabecalho) - 2 + len(planilhas) + 1 #O n mero 2 se refere
     a quantidade de linhas que sao descartadas durante a execu  o do
     script (linhas nao utilizadas da planilha) + 1 (devido a coluna de
     cotacoes)
 sheet.write(0,col_ibovespa,"ibovespa")
 for index, row in ibovespa.iterrows():
  ibovespa_deflacionado = row['Close']/deflator.cell_value(index+1,1)
  sheet.write(index+1,col_ibovespa,ibovespa_deflacionado)

 sheet.write(0,0,"data")
 planilhaSaida.save("empresa_dados_mes_ano/"+nomeEmpresa+".xls")
def calcula_juro_real(igpm_mes,selic_mes):
 return ((1+selic_mes)/(1+igpm_mes))-1

# Transforma a planilha de xls para CSV usando o plugin pandas
def toCSV(nomeEmpresa):
    data_xls = pd.read_excel('empresa_dados_mes_ano/'+nomeEmpresa+'.xls', '
        dados_e_indicadores', index_col=None, encoding='utf-8')
    data_xls.to_csv('empresa_dados_mes_ano_csv/'+nomeEmpresa+'.csv',
        encoding='utf-8', index=False, sep=',', decimal=".")


# Cria diretorio caso na exista
helper.create_directory('empresa_dados_mes_ano_csv/')
empresas = helper.empresas_csv() #retorna a lista de nomes da empresa com a
    extensao .csv

#planilhasCreate('abev3')
#toCSV('abev3')

helper.create_directory('empresa_dados_mes_ano_csv')
for empresa in empresas:
 if helper.exists_directory('empresa_dados_mes_ano_csv/'+empresa):
  print empresa + ' ja processada!'
  continue
 nomeEmpresa = empresa.replace(".csv","") # tira o .csv do nome
 print("Processa empresa_dados_mes_ano " + nomeEmpresa + " Comecou!")
 planilhasCreate(nomeEmpresa)
 print("empresa_dados_mes_ano/ " + nomeEmpresa + ".xls Criado!" )
```

```
259  print("Converter para CSV empresa_dados_mes_ano " + nomeEmpresa + "
         Comecou!")
     toCSV(nomeEmpresa)
261  print("empresa_dados_mes_ano_csv/ " + nomeEmpresa + ".csv Criado!" )
     print("Sucesso!")
263


265  grupos_empresas = xlrd.open_workbook("grupos_empresas.xlsx")
     dados = grupos_empresas.sheet_by_index(0)
267  empresa_dados_mes_ano_csv = 'empresa_dados_mes_ano_csv'
     helper.create_directory('empresa_dados_mes_ano_csv/grupos')
269  for curr_col in range(dados.ncols):
       grupo = dados.cell_value(0,curr_col)
271    helper.create_directory('empresa_dados_mes_ano_csv/grupos/'+grupo)
       print 'copiando para grupo ' + grupo
273    for curr_row in range(1,dados.nrows-1):
        empresa = dados.cell_value(curr_row,curr_col)
275     if empresa != "":
         print 'copiar: ' + empresa
277      data_xls = pd.read_excel('empresa_dados_mes_ano/'+empresa+'.xls', '
             dados_e_indicadores', index_col=None, encoding='utf-8')
         data_xls.to_csv('empresa_dados_mes_ano_csv/grupos/'+grupo+'/'+empresa+'
             .csv', encoding='utf-8',index=False, sep=',', decimal=".")
279      print 'copiado: ' + empresa
```

I.12 gerar_empresa_dados_mes_ano_trimestral.py

```
1  # encoding=utf8
   import sys
3  reload(sys)  #
   sys.setdefaultencoding('utf8')
5

   import openpyxl
7  import matplotlib.pyplot as plt
   import seaborn as sns
9  import xlsxwriter
   import helper
11 import numpy as np
   import pandas as pd
13

   # import numpy as np
15 # import plotly.plotly as py
   # import plotly.graph_objs as go
```

```
# import seaborn as sns

def planilhasCreate(nomeEmpresa):
 empresa = pd.read_csv("empresa_dados_mes_ano_csv/"+nomeEmpresa+".csv",
     index_col=0, sep=',', decimal=".") # carrega aquivo csv de

 empresa_trimestre = empresa.copy()
 for i in range(0,42,3):
  empresa_trimestre.ix[i+1,'cotacao'] = None
  empresa_trimestre.ix[i+2,'cotacao'] = None

 empresa_trimestre = empresa_trimestre.dropna(axis=0, how='any') # exclui
     qualquer linha que tenha valor nulo
 empresa_trimestre.to_csv('empresa_dados_mes_ano_csv_trimestral/'+
     nomeEmpresa+'_trimestre.csv', encoding='utf-8', sep=',', decimal=".")



helper.create_directory('empresa_dados_mes_ano_csv_trimestral')
empresas = helper.empresas_csv() #retorna a lista de nomes da empresa com a
    extensao .csv
for empresa in empresas:
 #if helper.exists_directory('empresa_dados_mes_ano_csv/
     cotacao_deslocada_um_mes/'+empresa):
 # print empresa + ' ja processada!'
 # continue
 nomeEmpresa = empresa.replace(".csv","") # tira o .csv do nome
 print("Processa empresa_dados_mes_ano_csv_trimestral " + nomeEmpresa + "
     Comecou!")
 planilhasCreate(nomeEmpresa)
print("Sucesso!")
```

I.13 gerar_empresa_x_indicadores.py

```
# -*- coding: utf-8 -*-
import pandas as pd
import openpyxl
import matplotlib.pyplot as plt
import seaborn as sns
import xlsxwriter
import helper
import numpy as np
```

```
import sys
reload(sys)  #
sys.setdefaultencoding('utf8')

def planilhasCreate(index, columns):
 empresas = helper.empresas_csv() #retorna a lista de nomes da empresa com
     a extensao .csv
 df = pd.DataFrame([],columns= columns)
 df_normalizado = pd.DataFrame([],columns= columns)
 df = df.rename_axis('acao')
 nomePlanilha = str(index).replace('/','-')
 for empresa in empresas:
  nomeEmpresa = empresa.replace(".csv","")
  empresa = pd.read_csv("empresa_dados_mes_ano_csv_trimestral/"+nomeEmpresa
      +"_trimestre.csv", index_col=0, sep=',', decimal=".") # carrega aquivo
       csv de

  df.loc[nomeEmpresa,:] = empresa.loc[index,:]

 df.index.rename('acao', inplace=True)

 zero_cols = [ col for col, is_zero in ((df == 0).sum() == df.shape[0]).
     items() if is_zero ]
 zero_cols.extend([col for col in df.columns if float(df.ix[0,col]) == 0])

 df.to_csv('empresa_x_indicadores/'+nomePlanilha+'_empresa_x_indicadores.
     csv', encoding='utf-8', sep=';', decimal=",")

 return zero_cols

def normaliza(df):
     result = pd.DataFrame(df.copy())

     for feature_name in df.columns:
      max_value = df.ix[0,feature_name]
      result[feature_name] = df[feature_name] / max_value
     return result

def normalizaLinha(df):
     result = pd.DataFrame(df.copy())

     for row_title, row in df.iterrows():
   max_value = row[0]
  # print row_title
   for col_title, val in row.items():
```

```
    if col_title == 'cotacao':
50     break
    result.loc[row_title,col_title] = float(val)/float(max_value)
52
     return result
54
def normalizaEmpresa(nomeEmpresa):
 empresa1 = pd.read_csv("empresa_x_indicadores/"+nomeEmpresa+".csv",
     index_col=0, sep=';', decimal=",") # carrega aquivo csv de
 empresa1.to_csv("empresa_x_indicadores/"+nomeEmpresa+".csv", encoding='utf
     -8', sep=';', decimal=",", float_format='%g')
58 empresa = normalizaLinha(empresa1)

60 empresa.to_csv('empresa_x_indicadores/normalizado/'+nomeEmpresa+'_norm.csv
     ', encoding='utf-8', sep=',', decimal=".", float_format='%.20f')


62
def excluiColunasZeradas(nomeEmpresa,all_zero_cols):
64 empresa = pd.read_csv('empresa_x_indicadores/' +nomeEmpresa+'.csv',
     index_col=0, sep=';', decimal=",") # carrega aquivo csv de
 empresa.drop(all_zero_cols, axis=1, inplace=True)
66 empresa.to_csv('empresa_x_indicadores/'+nomeEmpresa+'.csv', encoding='utf
     -8', sep=';', decimal=",")

68 helper.create_directory('empresa_x_indicadores/normalizado')
empresa = pd.read_csv("empresa_dados_mes_ano_csv_trimestral/abev3_trimestre
    .csv", index_col=0, sep=',', decimal=".") # carrega aquivo csv de
70 zero_cols_old = []
all_zero_cols = []
72 for index  in empresa.index:
 print 'Begin create '+ index + '_empresa_x_indicadores'
74 zero_cols_atual = planilhasCreate(index,empresa.columns);

76 all_zero_cols.extend(zero_cols_atual)

78 print 'End create '+ index + '_empresa_x_indicadores'

80 print len(all_zero_cols)
all_zero_cols = list(set(all_zero_cols)) #exclui valores duplicados
82 print len(all_zero_cols)

84 empresas = helper.empresas_x_indicadores() #retorna a lista de nomes da
    empresa com a extensao .csv
for empresa  in empresas:
86 if helper.is_directory("empresa_x_indicadores/"+empresa):
```

```
    continue;
  nomeEmpresa = empresa.replace(".csv","")
  print 'Begin exclui colunas zeradas '+ index + '_empresa_x_indicadores'
  excluiColunasZeradas(nomeEmpresa, all_zero_cols)
  print 'End exclui colunas zeradas '+ index + '_empresa_x_indicadores'


helper.create_directory('empresa_x_indicadores/normalizado')
empresas = helper.empresas_x_indicadores() #retorna a lista de nomes da
    empresa com a extensao .csv
for empresa  in empresas:
  if helper.is_directory("empresa_x_indicadores/"+empresa):
    continue;
  nomeEmpresa = empresa.replace(".csv","")
  print 'Begin normaliza '+ index + '_empresa_x_indicadores'
  zero_cols_atual = normalizaEmpresa(nomeEmpresa);
  print 'End normaliza '+ index + '_empresa_x_indicadores'
```

I.14 helper.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import codecs # evitar unicode erro
import sys

import os
import shutil
from unicodedata import normalize

reload(sys)
sys.setdefaultencoding('utf-8')
UTF8Writer = codecs.getwriter('utf8') # evitar unicode erro
sys.stdout = UTF8Writer(sys.stdout) # evitar unicode erro


# Devolve copia de uma str substituindo os caracteres
# acentuados pelos seus equivalentes nao acentuados.
#
# ATENCAO: carateres graficos nao ASCII e nao alfa-numericos,
# tais como bullets, travessoes, aspas assimetricas, cedilha etc.
# sao simplesmente removidos!
#
def remover_acentos(txt, codif='utf-8'):
```

```
 return normalize('NFKD', txt.decode(codif)).encode('ASCII', 'ignore')

def replace_barra_e_espaco(txt):
 return txt.replace(" ","_").replace("/","_")
# Verifica se o diretorio existe, caso contrario cria o mesmo
def create_directory(directory):
 if not os.path.exists(directory):
  os.makedirs(directory)

def limita_tamanho(txt,tamanho):
 if len(txt) > tamanho:
  return txt[0:tamanho-1]
 return txt

def empresas_csv():
 diretoria_atual = os.getcwd()
 empresas_csv = os.listdir(os.getcwd()+'/cotacao')
 return empresas_csv

def empresas_cotacao_diaria_csv():
 diretoria_atual = os.getcwd()
 empresas_csv = os.listdir(os.getcwd()+'/cotacao_diaria/dentro_periodo')
 return empresas_csv

def empresas_x_indicadores():
 diretoria_atual = os.getcwd()
 empresas_csv = os.listdir(os.getcwd()+'/empresa_x_indicadores')
 return empresas_csv

# lista com todos os grupos
def grupos_empresa():
 diretoria_atual = os.getcwd()
 grupos_csv = os.listdir(diretoria_atual+'/empresa_dados_mes_ano_csv/grupos
     ')
 return grupos_csv

# lista com todas as empresas de um grupo
def empresas_grupo(grupo):
 diretoria_atual = os.getcwd()
 empresas_csv = os.listdir(diretoria_atual+'/empresa_dados_mes_ano_csv/
     grupos/'+grupo)
 return empresas_csv

# lista com todos os grupos
def grupos_empresa_cotacao_diaria():
```

```
 diretoria_atual = os.getcwd()
 grupos_csv = os.listdir(diretoria_atual+'/cotacao_diaria_deflacionada/
    grupos')
 return grupos_csv

# lista com todas as empresas de um grupo
def empresas_grupo_cotacao_diaria(grupo):
 diretoria_atual = os.getcwd()
 empresas_csv = os.listdir(diretoria_atual+'/cotacao_diaria_deflacionada/
    grupos/'+grupo)
 return empresas_csv

def exists_directory(directory):
 return os.path.exists(directory)

def is_directory(directory):
 return os.path.isdir(directory)

# copia um arquivo e cola em um diretorio a partir do diretorio atual
def copy_and_paste_arq(fonte,destino):
 #diretorio_atual = os.getcwd()
 #print str(os.getcwd())
 shutil.copy2(diretorio_atual + '/' + fonte, diretorio_atual + '/' +
    destino)
```

I.15 normaliza_empresa_dados_mes_ano_manipula_cotacao.py

```
import pandas as pd
import openpyxl
import matplotlib.pyplot as plt
import seaborn as sns
import xlsxwriter
import helper
import numpy as np
# encoding=utf8
import sys
reload(sys)  #
sys.setdefaultencoding('utf8')

def normaliza(df):
 #return (df-df.min())/(df.max()-df.min())
```

```
    result = df.copy()
    for feature_name in df.columns:
        max_value = df[feature_name].max()
        min_value = df[feature_name].min()
        min_max = max_value - min_value
        if min_max == 0 or np.isnan(min_max):
         result[feature_name] = 0
        else:
         result[feature_name] = (df[feature_name] - min_value) / min_max
    return result

def normalizaPercentual(df):
 #return 2*(df-df.min())/(df.max()-df.min())-1
 #https://stats.stackexchange.com/questions/178626/how-to-normalize-data-
    between-1-and-1
    result = df.copy()
    for feature_name in df.columns:
        max_value = df[feature_name].max()
        min_value = df[feature_name].min()
        min_max = max_value - min_value
        if min_max == 0 or np.isnan(min_max):
         result[feature_name] = 0
        else:
         result[feature_name] = (df[feature_name] - min_value) / min_max
    return 2*result-1

def planilhasCreate(nomeEmpresa):
 empresa = pd.read_csv("empresa_dados_mes_ano_csv/"+nomeEmpresa+".csv",
    index_col=0, sep=',', decimal=".") # carrega aquivo csv de
 empresa1 = pd.read_csv('empresa_dados_mes_ano_csv/cotacao_deslocada_1_mes/
    '+nomeEmpresa+'_deslocada1.csv', index_col=0, sep=',', decimal=".") #
    carrega aquivo csv de
 empresa3 = pd.read_csv('empresa_dados_mes_ano_csv/
    cotacao_deslocada_3_meses/'+nomeEmpresa+'_deslocada3.csv', index_col=0,
     sep=',', decimal=".") # carrega aquivo csv de
 empresa6 = pd.read_csv('empresa_dados_mes_ano_csv/
    cotacao_deslocada_6_meses/'+nomeEmpresa+'_deslocada6.csv', index_col=0,
     sep=',', decimal=".") # carrega aquivo csv de
 empresa9 = pd.read_csv('empresa_dados_mes_ano_csv/
    cotacao_deslocada_9_meses/'+nomeEmpresa+'_deslocada9.csv', index_col=0,
     sep=',', decimal=".") # carrega aquivo csv de
 empresa12 = pd.read_csv('empresa_dados_mes_ano_csv/
    cotacao_deslocada_12_meses/'+nomeEmpresa+'_deslocada12.csv', index_col
    =0, sep=',', decimal=".") # carrega aquivo csv de

```

```
percentual1 = pd.read_csv('empresa_dados_mes_ano_csv/
    cotacao_percentual_aumento1/'+nomeEmpresa+'_percentual_aumento1.csv',
    index_col=0, sep=',', decimal=".") # carrega aquivo csv de
percentual3 = pd.read_csv('empresa_dados_mes_ano_csv/
    cotacao_percentual_aumento3/'+nomeEmpresa+'_percentual_aumento3.csv',
    index_col=0, sep=',', decimal=".") # carrega aquivo csv de
percentual6 = pd.read_csv('empresa_dados_mes_ano_csv/
    cotacao_percentual_aumento6/'+nomeEmpresa+'_percentual_aumento6.csv',
    index_col=0, sep=',', decimal=".") # carrega aquivo csv de
percentual9 = pd.read_csv('empresa_dados_mes_ano_csv/
    cotacao_percentual_aumento9/'+nomeEmpresa+'_percentual_aumento9.csv',
    index_col=0, sep=',', decimal=".") # carrega aquivo csv de
percentual12 = pd.read_csv('empresa_dados_mes_ano_csv/
    cotacao_percentual_aumento12/'+nomeEmpresa+'_percentual_aumento12.csv',
     index_col=0, sep=',', decimal=".") # carrega aquivo csv de

empresa = normaliza(empresa)
empresa1 = normaliza(empresa1)
empresa3 = normaliza(empresa3)
empresa6 = normaliza(empresa6)
empresa9 = normaliza(empresa9)
empresa12 = normaliza(empresa12)

percentual1 = normalizaPercentual(percentual1)
percentual3 = normalizaPercentual(percentual3)
percentual6 = normalizaPercentual(percentual6)
percentual9 = normalizaPercentual(percentual9)
percentual12 = normalizaPercentual(percentual12)

empresa.to_csv('empresa_dados_mes_ano_csv_normalizado/'+nomeEmpresa+'_norm
    .csv', encoding='utf-8', sep=',', decimal=".")
empresa1.to_csv('empresa_dados_mes_ano_csv_normalizado/
    cotacao_deslocada_1_mes/'+nomeEmpresa+'_norm_deslocada1.csv', encoding=
    'utf-8', sep=',', decimal=".")
empresa3.to_csv('empresa_dados_mes_ano_csv_normalizado/
    cotacao_deslocada_3_meses/'+nomeEmpresa+'_norm_deslocada3.csv',
    encoding='utf-8', sep=',', decimal=".")
empresa6.to_csv('empresa_dados_mes_ano_csv_normalizado/
    cotacao_deslocada_6_meses/'+nomeEmpresa+'_norm_deslocada6.csv',
    encoding='utf-8', sep=',', decimal=".")
empresa9.to_csv('empresa_dados_mes_ano_csv_normalizado/
    cotacao_deslocada_9_meses/'+nomeEmpresa+'_norm_deslocada9.csv',
    encoding='utf-8', sep=',', decimal=".")
```

```
empresa12.to_csv('empresa_dados_mes_ano_csv_normalizado/
    cotacao_deslocada_12_meses/'+nomeEmpresa+'_norm_deslocada12.csv',
    encoding='utf-8', sep=',', decimal=".")

percentual1.to_csv('empresa_dados_mes_ano_csv_normalizado/
    cotacao_percentual_aumento1/'+nomeEmpresa+'_norm_percentual_aumento1.
    csv', encoding='utf-8', sep=',', decimal=".")
percentual3.to_csv('empresa_dados_mes_ano_csv_normalizado/
    cotacao_percentual_aumento3/'+nomeEmpresa+'_norm_percentual_aumento3.
    csv', encoding='utf-8', sep=',', decimal=".")
percentual6.to_csv('empresa_dados_mes_ano_csv_normalizado/
    cotacao_percentual_aumento6/'+nomeEmpresa+'_norm_percentual_aumento6.
    csv', encoding='utf-8', sep=',', decimal=".")
percentual9.to_csv('empresa_dados_mes_ano_csv_normalizado/
    cotacao_percentual_aumento9/'+nomeEmpresa+'_norm_percentual_aumento9.
    csv', encoding='utf-8', sep=',', decimal=".")
percentual12.to_csv('empresa_dados_mes_ano_csv_normalizado/
    cotacao_percentual_aumento12/'+nomeEmpresa+'_norm_percentual_aumento12.
    csv', encoding='utf-8', sep=',', decimal=".")



helper.create_directory('empresa_dados_mes_ano_csv_normalizado/')
helper.create_directory('empresa_dados_mes_ano_csv_normalizado/
    cotacao_deslocada_1_mes')
helper.create_directory('empresa_dados_mes_ano_csv_normalizado/
    cotacao_deslocada_3_meses')
helper.create_directory('empresa_dados_mes_ano_csv_normalizado/
    cotacao_deslocada_6_meses')
helper.create_directory('empresa_dados_mes_ano_csv_normalizado/
    cotacao_deslocada_9_meses')
helper.create_directory('empresa_dados_mes_ano_csv_normalizado/
    cotacao_deslocada_12_meses')
helper.create_directory('empresa_dados_mes_ano_csv_normalizado/
    cotacao_percentual_aumento1')
helper.create_directory('empresa_dados_mes_ano_csv_normalizado/
    cotacao_percentual_aumento3')
helper.create_directory('empresa_dados_mes_ano_csv_normalizado/
    cotacao_percentual_aumento6')
helper.create_directory('empresa_dados_mes_ano_csv_normalizado/
    cotacao_percentual_aumento9')
helper.create_directory('empresa_dados_mes_ano_csv_normalizado/
    cotacao_percentual_aumento12')
```

```
empresas = helper.empresas_csv() #retorna a lista de nomes da empresa com a
    extensao .csv
for empresa in empresas:
 nomeEmpresa = empresa.replace(".csv","") # tira o .csv do nome
 print("NORMALIZA empresa_dados_mes_ano " + nomeEmpresa + " Comecou!")
 planilhasCreate(nomeEmpresa)
 print(nomeEmpresa + ".csv Criado!" )
print("Sucesso!")
```

I.16 normaliza_empresa_dados_mes_ano_trimestral_colunas_selecionadas.py

```
# -*- coding: utf-8 -*-
import pandas as pd
import openpyxl
import matplotlib.pyplot as plt
import seaborn as sns
import xlsxwriter
import helper
import numpy as np
# encoding=utf8
import sys
reload(sys)  #
sys.setdefaultencoding('utf8')

def normaliza(df):
 result = df.copy()
   for feature_name in df.columns:
    mean = df[feature_name].iloc[0]
    if mean == 0 or np.isnan(mean):
      result[feature_name] = 0
    else:
      result[feature_name] = df[feature_name]*100 / mean
    return result

def planilhasCreate(nomeEmpresa):
 empresa = pd.read_csv("empresa_dados_mes_ano_csv_trimestral/"+nomeEmpresa+
    "_trimestre.csv", index_col=0, sep=',', decimal=".") # carrega aquivo
    csv de
 empresa_deslocada1 = pd.read_csv("empresa_dados_mes_ano_csv_trimestral/
    cotacao_deslocada_1_trimestre/"+nomeEmpresa+"_trimestre_deslocada1.csv"
    , index_col=0, sep=',', decimal=".") # carrega aquivo csv de
```

```
 empresa_percentual1 = pd.read_csv("empresa_dados_mes_ano_csv_trimestral/
     cotacao_percentual_aumento1/"+nomeEmpresa+"
     _trimestre_percentual_aumento1.csv", index_col=0, sep=',', decimal=".")
      # carrega aquivo csv de

 empresa = pd.DataFrame(empresa, columns=['Investimentos', 'Imobilizado','
     Intang vel','Capital Social Realizado','Reservas de Lucros','cotacao',
     'EV/EBIT','ROE','expecativa_compras_bens_maior_valor','
     expectativa_renda','Juro Real'])
 empresa = empresa.sort_index()
 empresa = normaliza(empresa)
 empresa_deslocada1 = pd.DataFrame(empresa_deslocada1, columns=['
     Investimentos', 'Imobilizado','Intang vel','Capital Social Realizado',
     'Reservas de Lucros','cotacao','EV/EBIT','ROE','
     expecativa_compras_bens_maior_valor','expectativa_renda','Juro Real'])
 empresa_deslocada1 = empresa_deslocada1.sort_index()
 empresa_deslocada1 = normaliza(empresa_deslocada1)
 empresa_percentual1 = pd.DataFrame(empresa_percentual1, columns=['
     Investimentos', 'Imobilizado','Intang vel','Capital Social Realizado',
     'Reservas de Lucros','cotacao','EV/EBIT','ROE','
     expecativa_compras_bens_maior_valor','expectativa_renda','Juro Real'])
 empresa_percentual1 = empresa_percentual1.sort_index()
 empresa_percentual1 = normaliza(empresa_percentual1)

 writer = pd.ExcelWriter('empresa_dados_mes_ano_trimestral/normalizado/'+
     nomeEmpresa+'_norm_trimestre.xlsx')
 empresa.to_excel(writer,'normal')
 empresa_deslocada1.to_excel(writer,'deslocada')
 empresa_percentual1.to_excel(writer,'percentual')
 writer.save()

helper.create_directory('empresa_dados_mes_ano_trimestral')
helper.create_directory('empresa_dados_mes_ano_trimestral/normalizado')
empresas = helper.empresas_csv() #retorna a lista de nomes da empresa com a
     extensao .csv
for empresa in empresas:
 nomeEmpresa = empresa.replace(".csv","") # tira o .csv do nome
 print("Processa empresa_dados_mes_ano " + nomeEmpresa + "_trimestre
     Comecou!")
 planilhasCreate(nomeEmpresa)
 print("NORMALIZA empresa_dados_mes_ano_csv_trimestral " + nomeEmpresa + "
     _trimestre.csv Criado!" )
print("Sucesso!")
```

I.17 normaliza_empresa_dados_mes_ano_trimestral_manipula_cotacao_mes.py

```
import pandas as pd
import openpyxl
import matplotlib.pyplot as plt
import seaborn as sns
import xlsxwriter
import helper
import numpy as np
# encoding=utf8
import sys
reload(sys)  #
sys.setdefaultencoding('utf8')

def normaliza(df):

 result = df.copy()
   for feature_name in df.columns:
    max_value = df[feature_name].max()
    min_value = df[feature_name].min()
    mean = df[feature_name].mean()
    min_max = max_value - min_value
    if min_max == 0 or np.isnan(min_max):
      result[feature_name] = 0
    else:
      result[feature_name] = (df[feature_name] - mean) / min_max
    return result

def planilhasCreate(nomeEmpresa):
 empresa = pd.read_csv("empresa_dados_mes_ano_csv_trimestral/"+nomeEmpresa+
     "_trimestre.csv", index_col=0, sep=',', decimal=".") # carrega aquivo
     csv de
 empresa_deslocada1 = pd.read_csv("empresa_dados_mes_ano_csv_trimestral/
     cotacao_deslocada_1_trimestre/"+nomeEmpresa+"_trimestre_deslocada1.csv"
     , index_col=0, sep=',', decimal=".") # carrega aquivo csv de
 empresa_percentual1 = pd.read_csv("empresa_dados_mes_ano_csv_trimestral/
     cotacao_percentual_aumento1/"+nomeEmpresa+"
     _trimestre_percentual_aumento1.csv", index_col=0, sep=',', decimal=".")
      # carrega aquivo csv de

 empresa = normaliza(empresa)
 empresa_deslocada1 = normaliza(empresa_deslocada1)
 empresa_percentual1 = normaliza(empresa_percentual1)
```

```
 empresa.to_csv('empresa_dados_mes_ano_csv_trimestral/normalizado/'+
     nomeEmpresa+'_norm_trimestre.csv', encoding='utf-8', sep=',', decimal="
     .")
 empresa_deslocada1.to_csv('empresa_dados_mes_ano_csv_trimestral/
     normalizado/cotacao_deslocada_1_trimestre/'+nomeEmpresa+'
     _norm_trimestre_deslocada1.csv', encoding='utf-8', sep=',', decimal="."
     )
 empresa_percentual1.to_csv('empresa_dados_mes_ano_csv_trimestral/
     normalizado/cotacao_percentual_aumento1/'+nomeEmpresa+'
     _norm_trimestre_percentual_aumento1.csv', encoding='utf-8', sep=',',
     decimal=".")

helper.create_directory('empresa_dados_mes_ano_csv_trimestral')
helper.create_directory('empresa_dados_mes_ano_csv_trimestral/normalizado')
helper.create_directory('empresa_dados_mes_ano_csv_trimestral/normalizado/
    cotacao_deslocada_1_trimestre')
helper.create_directory('empresa_dados_mes_ano_csv_trimestral/normalizado/
    cotacao_percentual_aumento1')
empresas = helper.empresas_csv() #retorna a lista de nomes da empresa com a
     extensao .csv
for empresa in empresas:
 nomeEmpresa = empresa.replace(".csv","") # tira o .csv do nome
 print("Processa empresa_dados_mes_ano " + nomeEmpresa + "_trimestre
     Comecou!")
 planilhasCreate(nomeEmpresa)
 print("NORMALIZA empresa_dados_mes_ano_csv_trimestral " + nomeEmpresa + "
     _trimestre.csv Criado!" )
print("Sucesso!")
```

I.18 plot_correlation_cotacao_ibovespa_with_date.py

```
import helper
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.dates as dates
import seaborn as sns
import numpy as np
# encoding=utf8
import sys
reload(sys)  #
sys.setdefaultencoding('utf8')
```

```
12 def plotCorrelationWithDate(nomeEmpresa):
   empresa_dados1 = pd.read_csv("empresa_dados_mes_ano_csv/"+nomeEmpresa+".
       csv",index_col=0, parse_dates=True, sep=';', decimal=",") # carrega
       aquivo csv de
14
   cotacao = empresa_dados1['cotacao']
16 juro_real = empresa_dados1['ibovespa']

18 empresa = empresa_dados1[['cotacao','ibovespa']] # retorna um objeto do
       tipo series
   correlacao = empresa.corr() # to_frame() transforma um objeto series em um
        dataframe
20 correlacao = correlacao.iloc[0,1]

22 #Cria diretorios caso nao existam
   diretorio = 'graficos/correlacao_date/cotacao_ibovespa'
24 helper.create_directory('graficos')
   helper.create_directory('graficos/correlacao_date')
26 helper.create_directory(diretorio)

28 nome_imagem = str(correlacao) + "_" + nomeEmpresa + "_cotacao_X_ibovespa"
   nome_imagem = helper.replace_barra_e_espaco(nome_imagem)
30 nome_imagem = helper.limita_tamanho(nome_imagem,120)

32 # Se ja existir arquivo interrompe
   if helper.exists_directory(diretorio + "/" + helper.remover_acentos(str(
       nome_imagem)) + ".png"):
34  print "Correlacao cotacao x juros reais ja processada: " + helper.
        remover_acentos(str(nome_imagem)) + ".png"
    return 0
36
   ##Configura graficos com dois eixos y
38 fig,ay1 = plt.subplots()

40 plt.gca().xaxis.set_major_formatter(dates.DateFormatter('%Y'))
   plt.gca().xaxis.set_major_locator(dates.DayLocator())
42 plt.gca().xaxis.set_label_text('Ano')
   plt.gca().xaxis.set_tick_params(rotation=10)
44
   ay1.set_ylabel('cotacao', color='r')
46 ay1.tick_params('y', colors='r')
   lns1 = plt.plot(empresa_dados1.index, cotacao, color='r')
48 ay2 = ay1.twinx()
   lns2 = ay2.plot(empresa_dados1.index, juro_real, color='b')
50 ay2.set_ylabel('ibovespa', color='b')
```

```
 ay2.tick_params('y', colors='b')
 # #END NOVO
 plt.ticklabel_format(style='sci',scilimits=(0,0),axis='y')
 plt.title('Correlacao: ' + str(correlacao))
 plt.savefig(diretorio + "/" + helper.remover_acentos(str(nome_imagem)) + "
     .png")
 plt.close()
 return 1

empresas = helper.empresas_csv() #retorna a lista de nomes da empresa com a
    extensao .csv
for empresa in empresas:
 nomeEmpresa = empresa.replace(".csv","") # tira o .csv do nome
 if plotCorrelationWithDate(nomeEmpresa) is 1:
  print "END: " + nomeEmpresa + "cotacao_X_ibovespa"
```

I.19 plot_correlation_cotacao_juros_real_with_date.py

```
import helper
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.dates as dates
import seaborn as sns
import numpy as np
# encoding=utf8
import sys
reload(sys)  #
sys.setdefaultencoding('utf8')

def plotCorrelationWithDate(nomeEmpresa):
 empresa_dados1 = pd.read_csv("empresa_dados_mes_ano_csv/"+nomeEmpresa+".
     csv",index_col=0, parse_dates=True, sep=';', decimal=",") # carrega
     aquivo csv de

 cotacao = empresa_dados1['cotacao']
 juro_real = empresa_dados1['Juro Real']

 empresa = empresa_dados1[['cotacao','Juro Real']] # retorna um objeto do
     tipo series
 correlacao = empresa.corr() # to_frame() transforma um objeto series em um
      dataframe
 correlacao = correlacao.iloc[0,1]
 #correlacao = str(correlacao[0])
```

```
#Cria diretorios caso nao existam
diretorio = 'graficos/correlacao_date/cotacao_juros_reais'
helper.create_directory('graficos')
helper.create_directory('graficos/correlacao_date')
helper.create_directory(diretorio)

nome_imagem = str(correlacao) + "_" + nomeEmpresa + "
    _cotacao_X_juros_reais"
nome_imagem = helper.replace_barra_e_espaco(nome_imagem)
nome_imagem = helper.limita_tamanho(nome_imagem,120)

# Se ja existir arquivo interrompe
if helper.exists_directory(diretorio + "/" + helper.remover_acentos(str(
    nome_imagem)) + ".png"):
  print "Correlacao cotacao x juros reais ja processada: " + helper.
      remover_acentos(str(nome_imagem)) + ".png"
  return 0

##Configura graficos com dois eixos y
fig,ay1 = plt.subplots()

plt.gca().xaxis.set_major_formatter(dates.DateFormatter('%Y'))
plt.gca().xaxis.set_major_locator(dates.DayLocator())
plt.gca().xaxis.set_label_text('Ano')
plt.gca().xaxis.set_tick_params(rotation=10)

ay1.set_ylabel('cotacao', color='r')
ay1.tick_params('y', colors='r')
lns1 = plt.plot(empresa_dados1.index, cotacao, color='r')
ay2 = ay1.twinx()
lns2 = ay2.plot(empresa_dados1.index, juro_real, color='b')
ay2.set_ylabel('juro_real', color='b')
ay2.tick_params('y', colors='b')
# #END NOVO
plt.ticklabel_format(style='sci',scilimits=(0,0),axis='y')
plt.title('Correlacao: ' + str(correlacao))
plt.savefig(diretorio + "/" + helper.remover_acentos(str(nome_imagem)) + "
    .png")
plt.close()
return 1

empresas = helper.empresas_csv() #retorna a lista de nomes da empresa com a
    extensao .csv
for empresa in empresas:
  nomeEmpresa = empresa.replace(".csv","") # tira o .csv do nome
```

```
 if plotCorrelationWithDate(nomeEmpresa) is 1:
63   print "END: " + nomeEmpresa + "cotacao_X_juro_real"
```

I.20 plot_correlation_cotacoes_with_date_two_enterprise.py

```
1 import helper
  import pandas as pd
3 import matplotlib.pyplot as plt
  import matplotlib.dates as dates
5 import seaborn as sns
  import numpy as np
7 # encoding=utf8
  import sys
9 reload(sys)  #
  sys.setdefaultencoding('utf8')
11
  def plotCorrelationWithDate(nomeEmpresa1,nomeEmpresa2):
13  empresa_dados1 = pd.read_csv("empresa_dados_mes_ano_csv/"+nomeEmpresa1+".
        csv",index_col=0, parse_dates=True, sep=';', decimal=",") # carrega
        aquivo csv de
    empresa_dados2 = pd.read_csv("empresa_dados_mes_ano_csv/"+nomeEmpresa2+".
        csv",index_col=0, parse_dates=True, sep=';', decimal=",") # carrega
        aquivo csv de
15
    cotacao_empresa1 = empresa_dados1['cotacao'] # retorna um objeto do tipo
        series
17  cotacao_empresa2 = empresa_dados2['cotacao']

19  correlacao = cotacao_empresa1.to_frame().corrwith(cotacao_empresa2.
        to_frame(),axis='index') # to_frame() transforma um objeto series em um
         dataframe

21  correlacao = str(correlacao[0])
    #Cria diretorios caso nao existam
23  diretorio = 'graficos/correlacao_date/cotacao'
    helper.create_directory('graficos')
25  helper.create_directory('graficos/correlacao_date')
    helper.create_directory(diretorio)
27
    nome_imagem = str(correlacao) + "_cotacao_" + nomeEmpresa1 + "_X_" +
        nomeEmpresa2
```

```
29  nome_imagem = helper.replace_barra_e_espaco(nome_imagem)
    nome_imagem = helper.limita_tamanho(nome_imagem,120)
31
    # Se ja existir arquivo interrompe
33  if helper.exists_directory(diretorio + "/" + helper.remover_acentos(str(
        nome_imagem)) + ".png"):
     print "Correlacao cotacao ja processada: " + helper.remover_acentos(str(
         nome_imagem)) + ".png"
35   return 0

37  ##Configura graficos com dois eixos y
    fig,ay1 = plt.subplots()
39
    plt.gca().xaxis.set_major_formatter(dates.DateFormatter('%Y'))
41  plt.gca().xaxis.set_major_locator(dates.DayLocator())
    plt.gca().xaxis.set_label_text('Ano')
43  plt.gca().xaxis.set_tick_params(rotation=10)

45  ay1.set_ylabel(nomeEmpresa1, color='r')
    ay1.tick_params('y', colors='r')
47  lns1 = plt.plot(empresa_dados1.index, cotacao_empresa1, color='r')
    ay2 = ay1.twinx()
49  lns2 = ay2.plot(empresa_dados2.index, cotacao_empresa2, color='b')
    ay2.set_ylabel(nomeEmpresa2, color='b')
51  ay2.tick_params('y', colors='b')
    # #END NOVO
53  plt.ticklabel_format(style='sci',scilimits=(0,0),axis='y')
    plt.title('Correlacao: ' + str(correlacao))
55  plt.savefig(diretorio + "/" + helper.remover_acentos(str(nome_imagem)) + "
        .png")
    plt.close()
57  return 1

59 empresas = helper.empresas_csv() #retorna a lista de nomes da empresa com a
       extensao .csv
   empresas2 = list(empresas) # cria copia da lista
61 for empresa1 in empresas:
    empresas2.remove(empresa1)
63  nomeEmpresa1 = empresa1.replace(".csv","") # tira o .csv do nome
    for empresa2 in empresas2:
65   nomeEmpresa2 = empresa2.replace(".csv","") # tira o .csv do nome
     if plotCorrelationWithDate(nomeEmpresa1,nomeEmpresa2) is 1:
67    print "END: " + nomeEmpresa1 + "_X_" + nomeEmpresa2
```

I.21 plot_correlation_group_with_date_two_enterprise.py

```
import helper
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.dates as dates
import seaborn as sns
# encoding=utf8
import sys
reload(sys)  #
sys.setdefaultencoding('utf8')

def plotCorrelationWithDate(grupo,nomeEmpresa1,nomeEmpresa2):
 empresa_dados1 = pd.read_csv("empresa_dados_mes_ano_csv/"+nomeEmpresa1+".
    csv",index_col=0, parse_dates=True, sep=';', decimal=",") # carrega
    aquivo csv de
 empresa_dados2 = pd.read_csv("empresa_dados_mes_ano_csv/"+nomeEmpresa2+".
    csv",index_col=0, parse_dates=True, sep=';', decimal=",") # carrega
    aquivo csv de

 correlacao = empresa_dados1.corrwith(empresa_dados2,axis='index')

 #Cria diretorios caso nao existam
 diretorio = 'graficos/correlacao_date/grupos/' + grupo + "/" +
    nomeEmpresa1 + '_x_' + nomeEmpresa2
 diretorio_variacao = 'graficos/correlacao_date/grupos/' + grupo + "/" +
    nomeEmpresa2 + '_x_' + nomeEmpresa1
 helper.create_directory('graficos')
 helper.create_directory('graficos/correlacao_date')
 helper.create_directory('graficos/correlacao_date/grupos')
 helper.create_directory('graficos/correlacao_date/grupos/'+grupo)
 helper.create_directory(diretorio)

 for row_title, val in correlacao.iteritems(): # percorre a linhas das
    correlacao
  # percorre cada coluna da linha
  if val > 0.5 or val < -0.5:
   nome_imagem = str(val) + "_" + row_title + '_' + nomeEmpresa1 + "_X_" +
      nomeEmpresa2
   nome_imagem = helper.replace_barra_e_espaco(nome_imagem)
   nome_imagem = helper.limita_tamanho(nome_imagem,100)

   #Configura graficos com dois eixos y
```

```
    fig,ay1 = plt.subplots()

    plt.gca().xaxis.set_major_formatter(dates.DateFormatter('%Y'))
    plt.gca().xaxis.set_major_locator(dates.DayLocator())
    plt.gca().xaxis.set_label_text('Ano')
    plt.gca().xaxis.set_tick_params(rotation=10)

    ay1.set_ylabel(nomeEmpresa1+'_'+row_title, color='r')
    ay1.tick_params('y', colors='r')
    lns1 = plt.plot(empresa_dados1.index, empresa_dados1[row_title], color='
        r')
    ay2 = ay1.twinx()
    lns2 = ay2.plot(empresa_dados1.index, empresa_dados2[row_title], color='
        b')
    ay2.set_ylabel(nomeEmpresa2+'_'+row_title, color='b')
    ay2.tick_params('y', colors='b')
    #END NOVO
    plt.ticklabel_format(style='sci',scilimits=(0,0),axis='y')
    plt.title('Correlacao: ' + str(val))
    plt.savefig(diretorio + "/" + helper.remover_acentos(nome_imagem) + ".
        png")
    plt.close()

grupos = helper.grupos_empresa() #retorna a lista de nomes da empresa com a
    extensao .csv
for grupo in grupos:
 print 'Grupo: '+ grupo
 empresas = helper.empresas_grupo(grupo)
 empresas2 = list(empresas) # cria copia da lista
 for empresa1 in empresas:
  empresas2.remove(empresa1)
  nomeEmpresa1 = empresa1.replace(".csv","") # tira o .csv do nome
  for empresa2 in empresas2:
   nomeEmpresa2 = empresa2.replace(".csv","") # tira o .csv do nome
   diretorio = 'graficos/correlacao_date/grupos/' + grupo + "/" +
       nomeEmpresa1 + '_x_' + nomeEmpresa2
   diretorio_variacao = 'graficos/correlacao_date/grupos/' + grupo + "/" +
       nomeEmpresa2 + '_x_' + nomeEmpresa1
   # verifica se o diretorio ja para essas duas empresas,caso ja exista
       significa que as empresas ja foram processadas em dupla
   if helper.exists_directory(diretorio) or helper.exists_directory(
       diretorio_variacao):
    print 'Correlacao ja processada:' + nomeEmpresa1 + "_x_" + nomeEmpresa2
    continue
   plotCorrelationWithDate(grupo,nomeEmpresa1,nomeEmpresa2)
```

```
71      print 'Processou: '+ nomeEmpresa1 + "_x_" + nomeEmpresa2

73  #plotCorrelationWithDate('jbss3','brfs3')
```

I.22 plot_correlation_with_date.py

```
1   import helper
    import pandas as pd
3   import matplotlib.pyplot as plt
    import matplotlib.dates as dates
5   import seaborn as sns
    # encoding=utf8
7   import sys
    reload(sys)  #
9   sys.setdefaultencoding('utf8')

11  # Retorna o valor(x) somado ao modulo do parametro minimo(min) e + 1 para
        valores zero, pois nao existe log de 0
    def sum_min_abs(x,minimo):
13      return x + minimo*(-1) + 1

15  def plotCorrelationWithDate(nomeEmpresa):
     empresa_correlacao = pd.read_csv("correlacao_csv/"+nomeEmpresa+".csv",
         index_col=0, parse_dates=True, sep=';', decimal=",") # carrega aquivo
         csv de
17   empresa_dados = pd.read_csv("empresa_dados_mes_ano_csv/"+nomeEmpresa+".csv
         ",index_col=0, parse_dates=True, sep=';', decimal=",") # carrega aquivo
          csv de

19   #Cria diretorios caso nao existam
     diretorio = 'graficos/correlacao_date/' + nomeEmpresa
21   helper.create_directory('graficos')
     helper.create_directory('graficos/correlacao_date')
23   helper.create_directory(diretorio)
     rows_list_processed = [] #listas de linhas que ja foram processadas
25   for row_title, row in empresa_correlacao.iterrows(): # percorre a linhas
         das correlacao
      #adiciona linha que sera processa na lista de processadas
27    rows_list_processed.append(row_title)

29    # Retira colunas precessadas
      # Exemplo:
```

```
31  # o Ativo Total tem correlacao dentro do estipulado com o Ativo
        Circulante.
    # Quando a linha do Ativo Total eh processada a o grafico com o Ativo
        circulante eh gerado Ativo_total_X_Ativo_circulante
33  # caso essa funcao nao existisse
    # Quando a linha do Ativo circulante fosse processada seria criado
        novamente o grafico entre essas duas colunas
        Ativo_circulante_X_Ativo_total
35  # que eh igual ao grafico anteriormente criado
    row = row.drop(rows_list_processed)
37
    # percorre cada coluna da linha
39  for column_title, val in row.items():
     if val > 0.5 or val < -0.5:
41    nome_imagem = str(val) + "_" + row_title + "_X_" + column_title
      nome_imagem = helper.replace_barra_e_espaco(nome_imagem)
43    nome_imagem = helper.limita_tamanho(nome_imagem,120)

45    empresa_dados_plot = empresa_dados[[row_title,column_title]] # Retorna
          um dataframe com as colunas a serem plotadas

47    #Configura graficos com dois eixos y
      fig,ay1 = plt.subplots()
49
      plt.gca().xaxis.set_major_formatter(dates.DateFormatter('%Y'))
51    plt.gca().xaxis.set_major_locator(dates.DayLocator())
      plt.gca().xaxis.set_label_text('Ano')
53    plt.gca().xaxis.set_tick_params(rotation=10)

55    ay1.set_ylabel(row_title, color='r')
      ay1.tick_params('y', colors='r')
57    lns1 = plt.plot(empresa_dados_plot.index, empresa_dados_plot[row_title
          ], color='r')
      ay2 = ay1.twinx()
59    lns2 = ay2.plot(empresa_dados_plot.index, empresa_dados_plot[
          column_title], color='b')
      ay2.set_ylabel(column_title, color='b')
61    ay2.tick_params('y', colors='b')
      #END NOVO
63    plt.ticklabel_format(style='sci',scilimits=(0,0),axis='y')
      plt.title('Correlacao: ' + str(val))
65    plt.savefig(diretorio + "/" + helper.remover_acentos(nome_imagem) + ".
          png")
      plt.close()
67
```

```
plotCorrelationWithDate('abev3')
```

I.23 plot_correlation_with_date_ibovespa.py

```
import helper
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.dates as dates
import seaborn as sns
import numpy as np
# encoding=utf8
import sys
reload(sys)  #
sys.setdefaultencoding('utf8')

# Retorna o valor(x) somado ao modulo do parametro minimo(min) e + 1 para
    valores zero, pois nao existe log de 0
def sum_min_abs(x,minimo):
    return x + minimo*(-1) + 1

def plotCorrelationWithDate(nomeEmpresa):
 empresa_correlacao = pd.read_csv("correlacao_csv/"+nomeEmpresa+".csv",
     index_col=0, parse_dates=True, sep=';', decimal=",") # carrega aquivo
     csv de
 empresa_dados = pd.read_csv("empresa_dados_mes_ano_csv/"+nomeEmpresa+".csv
     ",index_col=0, parse_dates=True, sep=';', decimal=",") # carrega aquivo
      csv de

 #Cria diretorios caso nao existam
 diretorio = 'graficos/correlacao_date/indicadores_x_ibovespa/' +
     nomeEmpresa
 helper.create_directory('graficos')
 helper.create_directory('graficos/correlacao_date')
 helper.create_directory('graficos/correlacao_date/indicadores_x_ibovespa')
 helper.create_directory(diretorio)
 #quantidade_dentro_padroes_correlacao = 0
 row_title = 'ibovespa'
 for column_title, val in empresa_correlacao[row_title].iteritems(): #
     percorre a linhas das correlacao
  if np.isnan(val): # se for null passa pra proxima iteracao
   continue
  nome_imagem = str(val) + "_" + row_title + "_X_" + column_title
  nome_imagem = helper.replace_barra_e_espaco(nome_imagem)
```

```
    nome_imagem = helper.limita_tamanho(nome_imagem,120)
34
    empresa_dados_plot = empresa_dados[[row_title,column_title]] # Retorna um
        dataframe com as colunas a serem plotadas
36
    #Configura graficos com dois eixos y
38  fig,ay1 = plt.subplots()

40  plt.gca().xaxis.set_major_formatter(dates.DateFormatter('%Y'))
    plt.gca().xaxis.set_major_locator(dates.DayLocator())
42  plt.gca().xaxis.set_label_text('Ano')
    plt.gca().xaxis.set_tick_params(rotation=10)
44
    ay1.set_ylabel(row_title, color='r')
46  ay1.tick_params('y', colors='r')
    lns1 = plt.plot(empresa_dados_plot.index, empresa_dados_plot[row_title],
        color='r')
48  ay2 = ay1.twinx()
    lns2 = ay2.plot(empresa_dados_plot.index, empresa_dados_plot[column_title
        ], color='b')
50  ay2.set_ylabel(column_title, color='b')
    ay2.tick_params('y', colors='b')
52  #END NOVO
    plt.ticklabel_format(style='sci',scilimits=(0,0),axis='y')
54  plt.title('Correlacao: ' + str(val))
    plt.savefig(diretorio + "/" + helper.remover_acentos(nome_imagem) + ".png
        ")
56  plt.close()

58 plotCorrelationWithDate('abev3')
```

I.24 separa_teste_treino_empresa_dados.py

```
import pandas as pd
2 import openpyxl
import matplotlib.pyplot as plt
4 import seaborn as sns
import xlsxwriter
6 import helper
import numpy as np
8 # encoding=utf8
import sys
10 reload(sys)  #
```

```
sys.setdefaultencoding('utf8')

def planilhasCreate(nomeEmpresa):
 empresa = pd.read_csv("empresa_dados_mes_ano_csv/"+nomeEmpresa+".csv",
     index_col=0, sep=',', decimal=".") # carrega aquivo csv de
 empresa1 = pd.read_csv('empresa_dados_mes_ano_csv/cotacao_deslocada_1_mes/
     '+nomeEmpresa+'_deslocada1.csv', index_col=0, sep=',', decimal=".") #
     carrega aquivo csv de
 empresa3 = pd.read_csv('empresa_dados_mes_ano_csv/
     cotacao_deslocada_3_meses/'+nomeEmpresa+'_deslocada3.csv', index_col=0,
      sep=',', decimal=".") # carrega aquivo csv de
 empresa6 = pd.read_csv('empresa_dados_mes_ano_csv/
     cotacao_deslocada_6_meses/'+nomeEmpresa+'_deslocada6.csv', index_col=0,
      sep=',', decimal=".") # carrega aquivo csv de
 empresa9 = pd.read_csv('empresa_dados_mes_ano_csv/
     cotacao_deslocada_9_meses/'+nomeEmpresa+'_deslocada9.csv', index_col=0,
      sep=',', decimal=".") # carrega aquivo csv de
 empresa12 = pd.read_csv('empresa_dados_mes_ano_csv/
     cotacao_deslocada_12_meses/'+nomeEmpresa+'_deslocada12.csv', index_col
     =0, sep=',', decimal=".") # carrega aquivo csv de


 percentual1 = pd.read_csv('empresa_dados_mes_ano_csv/
     cotacao_percentual_aumento1/'+nomeEmpresa+'_percentual_aumento1.csv',
     index_col=0, sep=',', decimal=".") # carrega aquivo csv de
 percentual3 = pd.read_csv('empresa_dados_mes_ano_csv/
     cotacao_percentual_aumento3/'+nomeEmpresa+'_percentual_aumento3.csv',
     index_col=0, sep=',', decimal=".") # carrega aquivo csv de
 percentual6 = pd.read_csv('empresa_dados_mes_ano_csv/
     cotacao_percentual_aumento6/'+nomeEmpresa+'_percentual_aumento6.csv',
     index_col=0, sep=',', decimal=".") # carrega aquivo csv de
 percentual9 = pd.read_csv('empresa_dados_mes_ano_csv/
     cotacao_percentual_aumento9/'+nomeEmpresa+'_percentual_aumento9.csv',
     index_col=0, sep=',', decimal=".") # carrega aquivo csv de
 percentual12 = pd.read_csv('empresa_dados_mes_ano_csv/
     cotacao_percentual_aumento12/'+nomeEmpresa+'_percentual_aumento12.csv',
      index_col=0, sep=',', decimal=".") # carrega aquivo csv de

 empresa_teste = empresa.iloc[:12, :]
 empresa_treino = empresa.iloc[12:,:]
 empresa1_teste = empresa1.iloc[:12, :]
 empresa1_treino = empresa1.iloc[12:, :]
 empresa3_teste = empresa3.iloc[:12, :]
 empresa3_treino = empresa3.iloc[12:, :]
 empresa6_teste = empresa6.iloc[:12, :]
```

```
empresa6_treino = empresa6.iloc[12:, :]
empresa9_teste = empresa9.iloc[:12, :]
empresa9_treino = empresa9.iloc[12:, :]
empresa12_teste = empresa12.iloc[:12, :]
empresa12_treino = empresa12.iloc[12:, :]


percentual1_teste = percentual1.iloc[:12, :]
percentual1_treino = percentual1.iloc[12:, :]
percentual3_teste = percentual3.iloc[:12, :]
percentual3_treino = percentual3.iloc[12:, :]
percentual6_teste = percentual6.iloc[:12, :]
percentual6_treino = percentual6.iloc[12:, :]
percentual9_teste = percentual9.iloc[:12, :]
percentual9_treino = percentual9.iloc[12:, :]
percentual12_teste = percentual12.iloc[:12, :]
percentual12_treino = percentual12.iloc[12:, :]


empresa_teste.to_csv('empresa_dados_mes_ano_csv/teste/'+nomeEmpresa+'
    _teste.csv', encoding='utf-8', sep=',', decimal=".")
empresa1_teste.to_csv('empresa_dados_mes_ano_csv/teste/
    cotacao_deslocada_1_mes/'+nomeEmpresa+'_teste_deslocada1.csv', encoding
    ='utf-8', sep=',', decimal=".")
empresa3_teste.to_csv('empresa_dados_mes_ano_csv/teste/
    cotacao_deslocada_3_meses/'+nomeEmpresa+'_teste_deslocada3.csv',
    encoding='utf-8', sep=',', decimal=".")
empresa6_teste.to_csv('empresa_dados_mes_ano_csv/teste/
    cotacao_deslocada_6_meses/'+nomeEmpresa+'_teste_deslocada6.csv',
    encoding='utf-8', sep=',', decimal=".")
empresa9_teste.to_csv('empresa_dados_mes_ano_csv/teste/
    cotacao_deslocada_9_meses/'+nomeEmpresa+'_teste_deslocada9.csv',
    encoding='utf-8', sep=',', decimal=".")
empresa12_teste.to_csv('empresa_dados_mes_ano_csv/teste/
    cotacao_deslocada_12_meses/'+nomeEmpresa+'_teste_deslocada12.csv',
    encoding='utf-8', sep=',', decimal=".")
empresa_treino.to_csv('empresa_dados_mes_ano_csv/treino/'+nomeEmpresa+'
    _treino.csv', encoding='utf-8', sep=',', decimal=".")
empresa1_treino.to_csv('empresa_dados_mes_ano_csv/treino/
    cotacao_deslocada_1_mes/'+nomeEmpresa+'_treino_deslocada1.csv',
    encoding='utf-8', sep=',', decimal=".")
empresa3_treino.to_csv('empresa_dados_mes_ano_csv/treino/
    cotacao_deslocada_3_meses/'+nomeEmpresa+'_treino_deslocada3.csv',
    encoding='utf-8', sep=',', decimal=".")
```

```
empresa6_treino.to_csv('empresa_dados_mes_ano_csv/treino/
    cotacao_deslocada_6_meses/'+nomeEmpresa+'_treino_deslocada6.csv',
    encoding='utf-8', sep=',', decimal=".")
empresa9_treino.to_csv('empresa_dados_mes_ano_csv/treino/
    cotacao_deslocada_9_meses/'+nomeEmpresa+'_treino_deslocada9.csv',
    encoding='utf-8', sep=',', decimal=".")
empresa12_treino.to_csv('empresa_dados_mes_ano_csv/treino/
    cotacao_deslocada_12_meses/'+nomeEmpresa+'_treino_deslocada12.csv',
    encoding='utf-8', sep=',', decimal=".")

percentual1_teste.to_csv('empresa_dados_mes_ano_csv/teste/
    cotacao_percentual_aumento1/'+nomeEmpresa+'_teste_percentual_aumento1.
    csv', encoding='utf-8', sep=',', decimal=".")
percentual3_teste.to_csv('empresa_dados_mes_ano_csv/teste/
    cotacao_percentual_aumento3/'+nomeEmpresa+'_teste_percentual_aumento3.
    csv', encoding='utf-8', sep=',', decimal=".")
percentual6_teste.to_csv('empresa_dados_mes_ano_csv/teste/
    cotacao_percentual_aumento6/'+nomeEmpresa+'_teste_percentual_aumento6.
    csv', encoding='utf-8', sep=',', decimal=".")
percentual9_teste.to_csv('empresa_dados_mes_ano_csv/teste/
    cotacao_percentual_aumento9/'+nomeEmpresa+'_teste_percentual_aumento9.
    csv', encoding='utf-8', sep=',', decimal=".")
percentual12_teste.to_csv('empresa_dados_mes_ano_csv/teste/
    cotacao_percentual_aumento12/'+nomeEmpresa+'_teste_percentual_aumento12
    .csv', encoding='utf-8', sep=',', decimal=".")
percentual1_treino.to_csv('empresa_dados_mes_ano_csv/treino/
    cotacao_percentual_aumento1/'+nomeEmpresa+'_treino_percentual_aumento1.
    csv', encoding='utf-8', sep=',', decimal=".")
percentual3_treino.to_csv('empresa_dados_mes_ano_csv/treino/
    cotacao_percentual_aumento3/'+nomeEmpresa+'_treino_percentual_aumento3.
    csv', encoding='utf-8', sep=',', decimal=".")
percentual6_treino.to_csv('empresa_dados_mes_ano_csv/treino/
    cotacao_percentual_aumento6/'+nomeEmpresa+'_treino_percentual_aumento6.
    csv', encoding='utf-8', sep=',', decimal=".")
percentual9_treino.to_csv('empresa_dados_mes_ano_csv/treino/
    cotacao_percentual_aumento9/'+nomeEmpresa+'_treino_percentual_aumento9.
    csv', encoding='utf-8', sep=',', decimal=".")
percentual12_treino.to_csv('empresa_dados_mes_ano_csv/treino/
    cotacao_percentual_aumento12/'+nomeEmpresa+'
    _treino_percentual_aumento12.csv', encoding='utf-8', sep=',', decimal="
    .")

helper.create_directory('empresa_dados_mes_ano_csv/teste')
helper.create_directory('empresa_dados_mes_ano_csv/teste/
   cotacao_deslocada_1_mes/')
```

```
helper.create_directory('empresa_dados_mes_ano_csv/teste/
    cotacao_deslocada_3_meses')
helper.create_directory('empresa_dados_mes_ano_csv/teste/
    cotacao_deslocada_6_meses')
helper.create_directory('empresa_dados_mes_ano_csv/teste/
    cotacao_deslocada_9_meses')
helper.create_directory('empresa_dados_mes_ano_csv/teste/
    cotacao_deslocada_12_meses')
helper.create_directory('empresa_dados_mes_ano_csv/teste/
    cotacao_percentual_aumento1')
helper.create_directory('empresa_dados_mes_ano_csv/teste/
    cotacao_percentual_aumento3')
helper.create_directory('empresa_dados_mes_ano_csv/teste/
    cotacao_percentual_aumento6')
helper.create_directory('empresa_dados_mes_ano_csv/teste/
    cotacao_percentual_aumento9')
helper.create_directory('empresa_dados_mes_ano_csv/teste/
    cotacao_percentual_aumento12')
helper.create_directory('empresa_dados_mes_ano_csv/treino')
helper.create_directory('empresa_dados_mes_ano_csv/treino/
    cotacao_deslocada_1_mes/')
helper.create_directory('empresa_dados_mes_ano_csv/treino/
    cotacao_deslocada_3_meses')
helper.create_directory('empresa_dados_mes_ano_csv/treino/
    cotacao_deslocada_6_meses')
helper.create_directory('empresa_dados_mes_ano_csv/treino/
    cotacao_deslocada_9_meses')
helper.create_directory('empresa_dados_mes_ano_csv/treino/
    cotacao_deslocada_12_meses')
helper.create_directory('empresa_dados_mes_ano_csv/treino/
    cotacao_percentual_aumento1')
helper.create_directory('empresa_dados_mes_ano_csv/treino/
    cotacao_percentual_aumento3')
helper.create_directory('empresa_dados_mes_ano_csv/treino/
    cotacao_percentual_aumento6')
helper.create_directory('empresa_dados_mes_ano_csv/treino/
    cotacao_percentual_aumento9')
helper.create_directory('empresa_dados_mes_ano_csv/treino/
    cotacao_percentual_aumento12')
empresas = helper.empresas_csv() #retorna a lista de nomes da empresa com a
    extensao .csv
for empresa in empresas:
 nomeEmpresa = empresa.replace(".csv","") # tira o .csv do nome
 print("Separa teste e treino empresa_dados_mes_ano " + nomeEmpresa + "
    Comecou!")
```

```
104     planilhasCreate(nomeEmpresa)
        print(nomeEmpresa + ".csv Criado!" )
106 print("Sucesso!")
```

I.25 separa_teste_treino_empresa_dados_normalizado.py

```
1  import pandas as pd
   import openpyxl
3  import matplotlib.pyplot as plt
   import seaborn as sns
5  import xlsxwriter
   import helper
7  import numpy as np
   # encoding=utf8
9  import sys
   reload(sys)  #
11 sys.setdefaultencoding('utf8')

13 def planilhasCreate(nomeEmpresa):
    empresa = pd.read_csv("empresa_dados_mes_ano_csv_normalizado/"+nomeEmpresa
        +"_norm.csv", index_col=0, sep=',', decimal=".") # carrega aquivo csv
        de
15  empresa1 = pd.read_csv('empresa_dados_mes_ano_csv_normalizado/
        cotacao_deslocada_1_mes/'+nomeEmpresa+'_norm_deslocada1.csv', index_col
        =0, sep=',', decimal=".") # carrega aquivo csv de
    empresa3 = pd.read_csv('empresa_dados_mes_ano_csv_normalizado/
        cotacao_deslocada_3_meses/'+nomeEmpresa+'_norm_deslocada3.csv',
        index_col=0, sep=',', decimal=".") # carrega aquivo csv de
17  empresa6 = pd.read_csv('empresa_dados_mes_ano_csv_normalizado/
        cotacao_deslocada_6_meses/'+nomeEmpresa+'_norm_deslocada6.csv',
        index_col=0, sep=',', decimal=".") # carrega aquivo csv de
    empresa9 = pd.read_csv('empresa_dados_mes_ano_csv_normalizado/
        cotacao_deslocada_9_meses/'+nomeEmpresa+'_norm_deslocada9.csv',
        index_col=0, sep=',', decimal=".") # carrega aquivo csv de
19  empresa12 = pd.read_csv('empresa_dados_mes_ano_csv_normalizado/
        cotacao_deslocada_12_meses/'+nomeEmpresa+'_norm_deslocada12.csv',
        index_col=0, sep=',', decimal=".") # carrega aquivo csv de


21

    percentual1 = pd.read_csv('empresa_dados_mes_ano_csv_normalizado/
        cotacao_percentual_aumento1/'+nomeEmpresa+'_norm_percentual_aumento1.
        csv', index_col=0, sep=',', decimal=".") # carrega aquivo csv de
```

```
23 percentual3 = pd.read_csv('empresa_dados_mes_ano_csv_normalizado/
       cotacao_percentual_aumento3/'+nomeEmpresa+'_norm_percentual_aumento3.
       csv', index_col=0, sep=',', decimal=".") # carrega aquivo csv de
   percentual6 = pd.read_csv('empresa_dados_mes_ano_csv_normalizado/
       cotacao_percentual_aumento6/'+nomeEmpresa+'_norm_percentual_aumento6.
       csv', index_col=0, sep=',', decimal=".") # carrega aquivo csv de
25 percentual9 = pd.read_csv('empresa_dados_mes_ano_csv_normalizado/
       cotacao_percentual_aumento9/'+nomeEmpresa+'_norm_percentual_aumento9.
       csv', index_col=0, sep=',', decimal=".") # carrega aquivo csv de
   percentual12 = pd.read_csv('empresa_dados_mes_ano_csv_normalizado/
       cotacao_percentual_aumento12/'+nomeEmpresa+'_norm_percentual_aumento12.
       csv', index_col=0, sep=',', decimal=".") # carrega aquivo csv de
27
   empresa_teste = empresa.iloc[:12, :]
29 empresa_treino = empresa.iloc[12:,:]
   empresa1_teste = empresa1.iloc[:12, :]
31 empresa1_treino = empresa1.iloc[12:, :]
   empresa3_teste = empresa3.iloc[:12, :]
33 empresa3_treino = empresa3.iloc[12:, :]
   empresa6_teste = empresa6.iloc[:12, :]
35 empresa6_treino = empresa6.iloc[12:, :]
   empresa9_teste = empresa9.iloc[:12, :]
37 empresa9_treino = empresa9.iloc[12:, :]
   empresa12_teste = empresa12.iloc[:12, :]
39 empresa12_treino = empresa12.iloc[12:, :]


41
   percentual1_teste = percentual1.iloc[:12, :]
43 percentual1_treino = percentual1.iloc[12:, :]
   percentual3_teste = percentual3.iloc[:12, :]
45 percentual3_treino = percentual3.iloc[12:, :]
   percentual6_teste = percentual6.iloc[:12, :]
47 percentual6_treino = percentual6.iloc[12:, :]
   percentual9_teste = percentual9.iloc[:12, :]
49 percentual9_treino = percentual9.iloc[12:, :]
   percentual12_teste = percentual12.iloc[:12, :]
51 percentual12_treino = percentual12.iloc[12:, :]


53
   empresa_teste.to_csv('empresa_dados_mes_ano_csv_normalizado/teste/'+
       nomeEmpresa+'_norm_teste.csv', encoding='utf-8', sep=',', decimal=".")
55 empresa1_teste.to_csv('empresa_dados_mes_ano_csv_normalizado/teste/
       cotacao_deslocada_1_mes/'+nomeEmpresa+'_norm_teste_deslocada1.csv',
       encoding='utf-8', sep=',', decimal=".")
```

```
empresa3_teste.to_csv('empresa_dados_mes_ano_csv_normalizado/teste/
    cotacao_deslocada_3_meses/'+nomeEmpresa+'_norm_teste_deslocada3.csv',
    encoding='utf-8', sep=',', decimal=".")
empresa6_teste.to_csv('empresa_dados_mes_ano_csv_normalizado/teste/
    cotacao_deslocada_6_meses/'+nomeEmpresa+'_norm_teste_deslocada6.csv',
    encoding='utf-8', sep=',', decimal=".")
empresa9_teste.to_csv('empresa_dados_mes_ano_csv_normalizado/teste/
    cotacao_deslocada_9_meses/'+nomeEmpresa+'_norm_teste_deslocada9.csv',
    encoding='utf-8', sep=',', decimal=".")
empresa12_teste.to_csv('empresa_dados_mes_ano_csv_normalizado/teste/
    cotacao_deslocada_12_meses/'+nomeEmpresa+'_norm_teste_deslocada12.csv',
     encoding='utf-8', sep=',', decimal=".")
empresa_treino.to_csv('empresa_dados_mes_ano_csv_normalizado/treino/'+
    nomeEmpresa+'_norm_treino.csv', encoding='utf-8', sep=',', decimal=".")
empresa1_treino.to_csv('empresa_dados_mes_ano_csv_normalizado/treino/
    cotacao_deslocada_1_mes/'+nomeEmpresa+'_norm_treino_deslocada1.csv',
    encoding='utf-8', sep=',', decimal=".")
empresa3_treino.to_csv('empresa_dados_mes_ano_csv_normalizado/treino/
    cotacao_deslocada_3_meses/'+nomeEmpresa+'_norm_treino_deslocada3.csv',
    encoding='utf-8', sep=',', decimal=".")
empresa6_treino.to_csv('empresa_dados_mes_ano_csv_normalizado/treino/
    cotacao_deslocada_6_meses/'+nomeEmpresa+'_norm_treino_deslocada6.csv',
    encoding='utf-8', sep=',', decimal=".")
empresa9_treino.to_csv('empresa_dados_mes_ano_csv_normalizado/treino/
    cotacao_deslocada_9_meses/'+nomeEmpresa+'_norm_treino_deslocada9.csv',
    encoding='utf-8', sep=',', decimal=".")
empresa12_treino.to_csv('empresa_dados_mes_ano_csv_normalizado/treino/
    cotacao_deslocada_12_meses/'+nomeEmpresa+'_norm_treino_deslocada12.csv'
    , encoding='utf-8', sep=',', decimal=".")

percentual1_teste.to_csv('empresa_dados_mes_ano_csv_normalizado/teste/
    cotacao_percentual_aumento1/'+nomeEmpresa+'
    _norm_teste_percentual_aumento1.csv', encoding='utf-8', sep=',',
    decimal=".")
percentual3_teste.to_csv('empresa_dados_mes_ano_csv_normalizado/teste/
    cotacao_percentual_aumento3/'+nomeEmpresa+'
    _norm_teste_percentual_aumento3.csv', encoding='utf-8', sep=',',
    decimal=".")
percentual6_teste.to_csv('empresa_dados_mes_ano_csv_normalizado/teste/
    cotacao_percentual_aumento6/'+nomeEmpresa+'
    _norm_teste_percentual_aumento6.csv', encoding='utf-8', sep=',',
    decimal=".")
percentual9_teste.to_csv('empresa_dados_mes_ano_csv_normalizado/teste/
    cotacao_percentual_aumento9/'+nomeEmpresa+'
```

```
    _norm_teste_percentual_aumento9.csv', encoding='utf-8', sep=',',
    decimal=".")
percentual12_teste.to_csv('empresa_dados_mes_ano_csv_normalizado/teste/
    cotacao_percentual_aumento12/'+nomeEmpresa+'
    _norm_teste_percentual_aumento12.csv', encoding='utf-8', sep=',',
    decimal=".")
percentual1_treino.to_csv('empresa_dados_mes_ano_csv_normalizado/treino/
    cotacao_percentual_aumento1/'+nomeEmpresa+'
    _norm_treino_percentual_aumento1.csv', encoding='utf-8', sep=',',
    decimal=".")
percentual3_treino.to_csv('empresa_dados_mes_ano_csv_normalizado/treino/
    cotacao_percentual_aumento3/'+nomeEmpresa+'
    _norm_treino_percentual_aumento3.csv', encoding='utf-8', sep=',',
    decimal=".")
percentual6_treino.to_csv('empresa_dados_mes_ano_csv_normalizado/treino/
    cotacao_percentual_aumento6/'+nomeEmpresa+'
    _norm_treino_percentual_aumento6.csv', encoding='utf-8', sep=',',
    decimal=".")
percentual9_treino.to_csv('empresa_dados_mes_ano_csv_normalizado/treino/
    cotacao_percentual_aumento9/'+nomeEmpresa+'
    _norm_treino_percentual_aumento9.csv', encoding='utf-8', sep=',',
    decimal=".")
percentual12_treino.to_csv('empresa_dados_mes_ano_csv_normalizado/treino/
    cotacao_percentual_aumento12/'+nomeEmpresa+'
    _norm_treino_percentual_aumento12.csv', encoding='utf-8', sep=',',
    decimal=".")



helper.create_directory('empresa_dados_mes_ano_csv_normalizado/teste')
helper.create_directory('empresa_dados_mes_ano_csv_normalizado/teste/
    cotacao_deslocada_1_mes/')
helper.create_directory('empresa_dados_mes_ano_csv_normalizado/teste/
    cotacao_deslocada_3_meses')
helper.create_directory('empresa_dados_mes_ano_csv_normalizado/teste/
    cotacao_deslocada_6_meses')
helper.create_directory('empresa_dados_mes_ano_csv_normalizado/teste/
    cotacao_deslocada_9_meses')
helper.create_directory('empresa_dados_mes_ano_csv_normalizado/teste/
    cotacao_deslocada_12_meses')
helper.create_directory('empresa_dados_mes_ano_csv_normalizado/teste/
    cotacao_percentual_aumento1')
helper.create_directory('empresa_dados_mes_ano_csv_normalizado/teste/
    cotacao_percentual_aumento3')
```

```
helper.create_directory('empresa_dados_mes_ano_csv_normalizado/teste/
    cotacao_percentual_aumento6')
helper.create_directory('empresa_dados_mes_ano_csv_normalizado/teste/
    cotacao_percentual_aumento9')
helper.create_directory('empresa_dados_mes_ano_csv_normalizado/teste/
    cotacao_percentual_aumento12')
helper.create_directory('empresa_dados_mes_ano_csv_normalizado/treino')
helper.create_directory('empresa_dados_mes_ano_csv_normalizado/treino/
    cotacao_deslocada_1_mes/')
helper.create_directory('empresa_dados_mes_ano_csv_normalizado/treino/
    cotacao_deslocada_3_meses')
helper.create_directory('empresa_dados_mes_ano_csv_normalizado/treino/
    cotacao_deslocada_6_meses')
helper.create_directory('empresa_dados_mes_ano_csv_normalizado/treino/
    cotacao_deslocada_9_meses')
helper.create_directory('empresa_dados_mes_ano_csv_normalizado/treino/
    cotacao_deslocada_12_meses')
helper.create_directory('empresa_dados_mes_ano_csv_normalizado/treino/
    cotacao_percentual_aumento1')
helper.create_directory('empresa_dados_mes_ano_csv_normalizado/treino/
    cotacao_percentual_aumento3')
helper.create_directory('empresa_dados_mes_ano_csv_normalizado/treino/
    cotacao_percentual_aumento6')
helper.create_directory('empresa_dados_mes_ano_csv_normalizado/treino/
    cotacao_percentual_aumento9')
helper.create_directory('empresa_dados_mes_ano_csv_normalizado/treino/
    cotacao_percentual_aumento12')
empresas = helper.empresas_csv() #retorna a lista de nomes da empresa com a
    extensao .csv
for empresa in empresas:
 nomeEmpresa = empresa.replace(".csv","") # tira o .csv do nome
 print("Separa teste e treino empresa_dados_mes_ano_csv_normalizado " +
    nomeEmpresa + " Comecou!")
 planilhasCreate(nomeEmpresa)
 print(nomeEmpresa + ".csv Criado!" )
print("Sucesso!")
```

I.26 trimestre.py

```
import xlrd
import os

def empresas_csv():
```

```
    diretoria_atual = os.getcwd()
    empresas_csv = os.listdir(os.getcwd()+'/cotacao')
    return empresas_csv

class Variaveis:

    def __init__(self,cotacao,quantidadeAcoes,nomeEmpresa):
        self.cotacao = cotacao
        self.quantidadeAcoes = quantidadeAcoes
        self.planilhaBalanco = xlrd.open_workbook("balanco/"+nomeEmpresa+".xls")
        self.balanco = self.planilhaBalanco.sheet_by_index(0)
        self.demonstrativo = self.planilhaBalanco.sheet_by_index(1)

    def getEbit(self,coluna):
        return self.demonstrativo.cell_value(6,coluna)+self.demonstrativo.
            cell_value(7,coluna)+self.demonstrativo.cell_value(8,coluna)

    def getLucroLiquido(self,coluna):
        return self.demonstrativo.cell_value(25,coluna)

    def getLucroBruto(self,coluna):
        return self.demonstrativo.cell_value(6,coluna)

    def getReceitaLiquida(self,coluna):
        return self.demonstrativo.cell_value(4,coluna)

    def getPatrimonioLiquido(self,coluna):
        return self.balanco.cell_value(47,coluna)

    def getAtivoTotal(self,coluna):
        return self.balanco.cell_value(2,coluna)

    def getAtivoCirculante(self,coluna):
        return self.balanco.cell_value(3,coluna)

    def getPassivoCirculante(self,coluna):
        return self.balanco.cell_value(27,coluna)

    def getDividasDeCurtoElongoPrazo(self,coluna):
        return self.balanco.cell_value(28,coluna) + self.balanco.cell_value(29,
            coluna) + self.balanco.cell_value(30,coluna) + self.balanco.cell_value
            (31,coluna) + self.balanco.cell_value(33,coluna) + self.balanco.
            cell_value(38,coluna)

    def getDividendos(self,coluna):
```

```
46      return self.balanco.cell_value(30,coluna) + self.balanco.cell_value(31,
            coluna) + self.balanco.cell_value(33,coluna)

48  def getValorMercado(self):
        return self.cotacao * self.quantidadeAcoes
50
    def getDividaBruta(self,coluna):
52      return self.balanco.cell_value(31,coluna) + self.balanco.cell_value(38,
            coluna)

54  def getDisponibilidades(self,coluna):
        return self.balanco.cell_value(4,coluna) + self.balanco.cell_value(5,
            coluna)
56
    def getDividaLiquida(self,coluna):
58      return self.getDividaBruta(coluna) - self.getDisponibilidades(coluna)

60  def getValorFirma(self,coluna):
        return self.getValorMercado() + self.getDividaLiquida(coluna)
62
    def getCaixa(self,coluna):
64      return self.balanco.cell_value(4,coluna)

66  def getFornecedores(self,coluna):
        return self.balanco.cell_value(29,coluna)
68
    def getAtivosTotaisPorAcao(self,coluna):
70      try:
            return self.getAtivoTotal(coluna)/self.quantidadeAcoes
72      except ZeroDivisionError:
            return 0
74
    def getCapitalDeGiro(self,coluna):
76      return self.getAtivoCirculante(coluna) - self.getPassivoCirculante(coluna
            )

78  def getCapitalDeGiroPorAcao(self,coluna):
        try:
80          return self.getCapitalDeGiro(coluna)/self.quantidadeAcoes
        except ZeroDivisionError:
82          return 0
    def getAtivoCirculanteLiquido(self,coluna):
84      return self.getAtivoCirculante(coluna) - self.
            getDividasDeCurtoElongoPrazo(coluna)
```

```
86  def getAtivoCirculanteLiquidoPorAcao(self,coluna):
     try:
88    return self.getAtivoCirculanteLiquido(coluna)/self.quantidadeAcoes
     except ZeroDivisionError:
90    return 0
    def getDividendoPorAcao(self,coluna):
92   try:
      return self.getDividendos(coluna)/self.quantidadeAcoes
94   except ZeroDivisionError:
      return 0
96  # Lucro Liquido nos ultimos 12 meses
    def getLucroLiquidoUltimos12meses(self,coluna):
98   return self.getLucroLiquido(coluna) + self.getLucroLiquido(coluna+1) +
        self.getLucroLiquido(coluna+2) + self.getLucroLiquido(coluna+3)

100 def getReceitaLiquidaUltimos12meses(self,coluna):
     return self.getReceitaLiquida(coluna) + self.getReceitaLiquida(coluna+1)
        + self.getReceitaLiquida(coluna+2) + self.getReceitaLiquida(coluna+3)
102
    # Receita Liquida por acao nos ultimos 12 meses
104 def getReceitaLiquidaPorAcao(self,coluna):
     try:
106   return self.getReceitaLiquidaUltimos12meses(coluna)/self.quantidadeAcoes
     except ZeroDivisionError:
108   return 0

110 def getEbitUltimos12meses(self,coluna):
     return self.getEbit(coluna) + self.getEbit(coluna+1) + self.getEbit(
        coluna+2) + self.getEbit(coluna+3)
112
    def getEbitUltimos12mesesPorAcao(self,coluna):
114  try:
      return self.getEbitUltimos12meses(coluna)/self.quantidadeAcoes
116  except ZeroDivisionError:
      return 0
118 def getLucroBrutoUltimos12meses(self,coluna):
     return self.getLucroBruto(coluna) + self.getLucroBruto(coluna+1) + self.
        getLucroBruto(coluna+2) + self.getLucroBruto(coluna+3)
120
    # INDICADORES FUNDAMENTALISTAS
122
    # LP (Lucro por acao Ultimos 12 meses)
124 def getLucroPorAcao(self,coluna):
     try:
126   return self.getLucroLiquidoUltimos12meses(coluna)/self.quantidadeAcoes
```

```
    except ZeroDivisionError:
128     return 0
    # VPA (Patrimonio Liquido por acao)
130 def getPatrimonioLiquidoPorAcao(self,coluna):
    try:
132     return self.getPatrimonioLiquido(coluna)/self.quantidadeAcoes
    except ZeroDivisionError:
134     return 0
    # Marg. Bruta (Lucro Bruto/ Receita Liquida) - ultimos 12 meses
136 def getMargBruta(self,coluna):
    try:
138     return self.getLucroBrutoUltimos12meses(coluna)/self.
            getReceitaLiquidaUltimos12meses(coluna)
    except ZeroDivisionError:
140     return 0

142 # Marg. Ebit (EBIT/Receita Liquida) - ultimos 12 meses
    def getMargEBIT(self,coluna):
144 try:
    return self.getEbitUltimos12meses(coluna)/self.
            getReceitaLiquidaUltimos12meses(coluna)
146 except ZeroDivisionError:
    return 0
148 # Marg Liquida (Lucro Liquido/ Receita Liquida) ultimos 12 meses
    def getMargLiquida(self,coluna):
150 try:
    return self.getLucroLiquidoUltimos12meses(coluna)/self.
            getReceitaLiquidaUltimos12meses(coluna)
152 except ZeroDivisionError:
    return 0
154

    # EBIT/ATIVO (EBIT ultimos 12 meses/ Ativos totais ultimo trimestre)
156 def getEBITporAtivos(self,coluna):
    try:
158     return self.getEbitUltimos12meses(coluna)/self.getAtivoTotal(coluna)
    except ZeroDivisionError:
160     return 0

162 # ROIC = EBIT ultimos 12 meses/ (Ativos Totais - caixa - fornecedores)
        ultimo trimestre
    def getROIC(self,coluna):
164 try:
    return self.getEbitUltimos12meses(coluna)/(self.getAtivoTotal(coluna)-
            self.getCaixa(coluna)-self.getFornecedores(coluna))
166 except ZeroDivisionError:
```

```
    return 0

# ROE = Lucro Liquido ultimos 12 meses/ Patrimonio Liquido ultimo
    trimestre
def getROE(self,coluna):
 try:
  return self.getLucroLiquidoUltimos12meses(coluna)/self.
      getPatrimonioLiquido(coluna)
 except ZeroDivisionError:
  return 0

# Liquidez Corr (Ativo Circulante/Passivo Circulante)
def getLiquidezCorr(self,coluna):
 try:
  return self.getAtivoCirculante(coluna)/self.getPassivoCirculante(coluna)
 except ZeroDivisionError:
  return 0

# Div Br / Patrim (Divida Bruta / Patrimonio Liquido)
def getDividaBrutaPorPatrimonioLiquido(self,coluna):
 try:
  return self.getDividaBruta(coluna)/self.getPatrimonioLiquido(coluna)
 except ZeroDivisionError:
  return 0
######################

# P/L
def getPrecoAcaoPorLucro(self,coluna):
 try:
  return self.cotacao/self.getLucroPorAcao(coluna)
 except ZeroDivisionError:
  return 0

# P/VP
def getPrecoAcaoPorPatrimonioLiquidoPorAcao(self,coluna):
 try:
  return self.cotacao/self.getPatrimonioLiquidoPorAcao(coluna)
 except ZeroDivisionError:
  return 0

# P/EBIT
def getPrecoAcaoPorEBITUltimo12mesesPorAcao(self,coluna):
 try:
  return self.cotacao/self.getEbitUltimos12mesesPorAcao(coluna)
 except ZeroDivisionError:
```

```
210         return 0
211
212     # PSR
213     def getPrecoAcaoPorReceitaLiquidaPorAcao(self,coluna):
214         try:
215             return self.cotacao/self.getReceitaLiquidaPorAcao(coluna)
216         except ZeroDivisionError:
217             return 0
218
219     # P/Ativos
220     def getPrecoAcaoPorAtivosTotais(self,coluna):
221         try:
222             return self.cotacao/self.getAtivosTotaisPorAcao(coluna)
223         except ZeroDivisionError:
224             return 0
225
226     # P/Cap. Giro
227     def getPrecoAcaoPorCapitalGiroPorAcao(self,coluna):
228         try:
229             return self.cotacao/self.getCapitalDeGiroPorAcao(coluna)
230         except ZeroDivisionError:
231             return 0
232
233     # P/Ativ Circ Liq
234     def getPrecoAcaoPorAtivoCirculanteLiquidoPorAcao(self,coluna):
235         try:
236             return self.cotacao/self.getAtivoCirculanteLiquidoPorAcao(coluna)
237         except ZeroDivisionError:
238             return 0
239
240     # Div. Yield
241     def getDividendoYield(self,coluna):
242         try:
243             return self.getDividendoPorAcao(coluna)/self.cotacao
244         except ZeroDivisionError:
245             return 0
246
247     # EV/EBIT
248     def getValorDaFirmaPorEBITUltimos12meses(self,coluna):
249         try:
250             return self.getValorFirma(coluna)/self.getEbitUltimos12meses(coluna)
251         except ZeroDivisionError:
252             return 0
253
254     # Giros Ativos
```

```
def getGirosAtivos(self,coluna):
 try:
  return self.getReceitaLiquidaUltimos12meses(coluna)/self.getAtivoTotal(
      coluna)
 except ZeroDivisionError:
  return 0
```