UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Márcio Antônio de Freitas Júnior

Classificador LGBM Aplicado na Resolução do Desafio PLAsTiCC do Telescópio LSST

Uberlândia, Brasil

2019

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Márcio Antônio de Freitas Júnior

Classificador LGBM Aplicado na Resolução do Desafio PLAsTiCC do Telescópio LSST

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito exigido parcial à obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Marcelo Keese Albertini

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Ciência da Computação

Uberlândia, Brasil

2019

Márcio Antônio de Freitas Júnior

Classificador LGBM Aplicado na Resolução do Desafio PLAsTiCC do Telescópio LSST

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito exigido parcial à obtenção do grau de Bacharel em Ciência da Computação.

Trabalho aprovado. Uberlândia, Brasil, 12 de Julho de 2019:

Prof. Dr. Marcelo Keese Albertini
Orientador

Prof. Dr. Murillo Guimarães Carneiro

Prof. Dr. André Backes

Uberlândia, Brasil
2019

On the one hand, we all want to be happy. On the other hand, we all know the things that make us happy. But we don't do those things. Why? Simple. We are too busy. Too busy doing what? Too busy trying to be happy.

Matthew Kelly, The Rhythm of Life

Resumo

Em 2019 será construído o telescópio *Large Synoptic Survey Telescope* (LSST) que irá revolucionar a identificação de fontes astronômicas. Esse telescópio irá descobrir de 10 a 100 vezes mais objetos do que jamais foi descoberto, possivelmente incluindo classes de objetos ainda desconhecidos pela astronomia. Diante do volume de dados que serão gerados, algoritmos capazes de identificar e ordenar os objetos observados são cruciais. Para desenvolverem tais algoritmos, O LSST *Corporation* em conjunto com equipes de pesquisadores criaram o *The Photometric* LSST *Astronomical Time-Series Classification Challenge* (PLAsTiCC) com o intuito de obter colaboração da comunidade de *data science* e *machine learning*. Dito isso, esse trabalho de conclusão de curso constrói um modelo classificador para solucionar o problema proposto pelo PLAsTiCC.

Tal classificador emprega o algoritmo *Light Gradient Boosting Model* (LGBM) e executa as etapas de aprendizado e classificação sobre características extraídas do conjunto de dados fornecidos pela competição. Esses dados são simulações criadas a partir de observações astronômicas reais designadas a simular os dados reais que serão produzidos pelo LSST.

As predições do classificador atingiram a pontuação de 1,651 calculada pela competição através de uma *logarithmic loss function* e, se submetido dentro do prazo da competição, alcançaria a 704^a colocação dentre as predições dos 1094 times participantes.

Futuramente, se forem aplicados mais alguns métodos de pré-processamento e extração de características como, por exemplo, executar uma interpolação das observações de cada objeto e contar a quantidade de mínimos locais, e testar outros algoritmos classificadores, como as redes neurais artificiais, a possibilidade de alcançar boas colocações é muito alta.

Palavras-chave: *Machine learning*. classificação. LSST. PLAsTiCC. séries temporais.

Lista de ilustrações

Figura 1 – Exemplo de uma curva de luz usando a regressão pelo Processo Gaussiano. 25
Figura 2 – Matriz de confusão de Kyle Boone obtida através do *cross-validation* das predições do LightGBM. 26
Figura 3 – Exemplo de interpolação de uma curva de luz utilizando Celerite. . . . 28
Figura 4 – Exemplo de interpolação de uma curva de luz utilizando Bazin. 28
Figura 5 – Exemplo de interpolação de uma curva de luz utilizando Newling. . . . 29
Figura 6 – Histograma com análise dos fluxos dos objetos em um quantil de 0,05 a 0,95. 36
Figura 7 – Pontos com concentração de objetos representando as áreas DDFs. . . . 41
Figura 8 – Diferença das distribuições do *redshift* mensurado através da espectroscopia. 41
Figura 9 – Posição dos objetos mudando de cor de acordo com a mudança da data média da sua lacuna de observação. 42
Figura 10 – Classes divididas entre galácticas e extragalácticas. 43
Figura 11 – No eixo horizontal são identificados os objetos e no eixo vertical as médias, medianas e os desvios padrão dos fluxos de cada objeto de treino. 44
Figura 12 – Histograma da média, mediana e desvio padrão dos fluxos dos objetos de treino. 44
Figura 13 – Média dos fluxos do objetos de treino com coloração baseada na área de observação. 45
Figura 14 – Média dos fluxos objetos para cada classe separadamente. 52
Figura 15 – Proporção de cada classes nas duas áreas de observações. 53
Figura 16 – Disposição das fontes nas coordenadas *right ascension* e *declination*. . . 54
Figura 17 – Disposição das fontes nas coordenadas de latitude e longitude galácticas. 55
Figura 18 – Avermelhamento dos objetos de treino e suas distâncias modulares. . . 56
Figura 19 – Correlação entre as características extraídas do conjunto de treino. . . 56
Figura 20 – Média dos fluxos observados no *passband u* de cada objeto com coloração baseada em sua classe. 57
Figura 21 – Média dos fluxos observados de cada objeto em cada *passband* com coloração baseada em sua classe. 57
Figura 22 – Média dos fluxos observados de cada objeto em cada *passband* com coloração baseada em sua classe e eixo Y limitado. 58
Figura 23 – Média dos fluxos observados na parte inferior da média. 58
Figura 24 – Média dos fluxos observados na parte superior da média. 59
Figura 25 – Média dos fluxos observados limitados de -100 a 100 na área de DDF. 59
Figura 26 – Média dos fluxos observados limitados de -100 a 100 na área de WDF. 60

Figura 27 – Média dos fluxos observados, limitados de −200 a 200, das classes galácticas. 60
Figura 28 – Média dos fluxos observados, limitados de −200 a 200, das classes extragalácticas. 61
Figura 29 – Densidade e histograma das médias de cada classe diferenciadas pela cor. 61
Figura 30 – Densidade e gráfico em barras das médias das classes galácticas diferenciadas pela cor. 62
Figura 31 – Densidade e gráfico em barras das médias das classes extragalácticas diferenciadas pela cor. 63

Lista de tabelas

Tabela 1 – Tabela com a descrição das propriedades dos metadados da competição 23
Tabela 2 – Tabela com a descrição das propriedades dos dados da competição . . . 24
Tabela 3 – Tabela com a representação da evolução da pontuação de acordo com o acréscimo de características em determinada propriedade das observações adicionado a diferença entre as relações de proporção de partição do conjunto de treino e entre as características simples e por *passband* 49

Lista de abreviaturas e siglas

CSV	*Comma Separated Values*
CV	*Cross Validation*
DDF	*Deep Drilling Field*
EDA	*Exploratory Data Analysis*
GB	*Gigabyte*
GBDT	*Gradient Boosting Decision Tree*
h5	HDF versão 5
HDF	*Hierarchical Data Format*
JSON	*JavaScript Object Notation*
KNN	*K-Nearest Neighbors*
LB	*Leaderboard*
LGB	*Light Gradient Boosting*
LGBM	*Light Gradient Boosting Model*
LSST	*Large Synoptic Survey Telescope*
LSSTC	LSST *Corporation*
RAM	*Random Access Memory*
WDF	*Wide-Fast-Deep*

Sumário

1 INTRODUÇÃO 11

2 FUNDAMENTAÇÃO TEÓRICA 14
2.1 Astronomia 14
2.1.1 Propriedades das Observações 14
2.1.2 Propriedades das Fontes Observadas 15
2.2 Conceitos Computacionais 16
2.2.1 Árvores de Decisão 17
2.2.2 Light Boosting Gradient Model 17
2.2.3 Métodos de pre-processamento e classificação 18
2.2.4 Tecnologias Utilizadas 19
2.3 PLAsTiCC 20
2.4 Conjunto de Dados do PLAsTiCC 21
2.4.1 Ressalvas 22
2.5 Trabalhos Relacionados 22
2.5.1 Abordagem do Kyle Boone 24
2.5.2 Abordagem do Jean François Puget 26
2.5.3 Abordagem da Belinda Trotta 29
2.6 Considerações finais 30

3 DESENVOLVIMENTO 31
3.1 Estrutura do Código 34
3.1.1 Arquivo DataAnalysis.py 35
3.1.2 Arquivo ExtractFeatures.py 35
3.1.3 Arquivo LGBModel.py 37
3.1.4 Arquivos Auxiliares 38
3.2 Considerações finais 39

4 TESTES E EXPERIMENTOS 40
4.1 Análises Kaggle 40
4.2 Análise dos dados 42
4.2.1 A função *DataAnalysisInitia()l* 43
4.2.2 A função *DataAnalysis()* 45
4.2.3 A função *DataAnalysisSeparetePassbands()* 46
4.3 Testes 47
4.4 Considerações finais 51

5 CONCLUSÃO 64

REFERÊNCIAS 65

1 Introdução

Em 2019 será finalizada a construção do telescópio que irá revolucionar o entendimento dos homens sobre o espaço, descobrindo e mensurando milhões de objetos variantes no tempo, o *Large Synoptic Survey Telescope* (LSST) (JR et al., 2018). O LSST está sendo construído ao norte do deserto do Atacama, na montanha Cerro Panchon situada no Chile. O objetivo principal desse telescópio é conduzir uma pesquisa de 10 anos sobre o céu, que irá gerar um conjunto de 200 *petabytes* de imagens e informações, direcionado a responder as questões mais proeminentes sobre a estrutura e a evolução do universo e dos objetos neles existentes (LSSTC, 2019b).

Um dos maiores desafios provenientes do LSST é construir o *software* que irá processar os dados gerados por ele. São mais de 30 *terabytes* de medições, todas as noites, que serão processados e armazenados para produzir o maior *dataset* não-proprietário do mundo (LSSTC, 2019b). Uma das tarefas desse *software* é classificar cada objeto observado pelo telescópio em um tempo hábil utilizando de recursos factíveis. Para isto, o *LSST Dark Energy Science Collaboration* (LSSTC, 2019a) e o *LSST Transients and Variable Stars Science Collaboration* (BIANCO; STREET, 2019) desenvolveram o *Photometric LSST Astronomical Time-series Classification Challenge* (PLAsTiCC), um desafio publicado no *site* de competições Kaggle para premiar o criador do melhor algoritmo de classificação sobre o *dataset* de séries temporais astronômicas que simulam as futuras observações do telescópio LSST. Detalhes desse *dataset* são apresentados na seção 2.4 do Capítulo 2.

O objetivo deste trabalho de conclusão de curso é estudar aspectos práticos e computacionais relacionados à pergunta que o PLAsTiCC propõe: “Podemos classificar fontes astronômicas utilizando curvas de luz simuladas designadas a imitar os dados do LSST?” (JR et al., 2018). Com esse intuito, avaliaram-se empiricamente métodos propostos na literatura cujo objetivo era obter um modelo de classificação de objetos astronômicos.

Os métodos propostos na literatura foram treinados a partir de conjunto de dados que descrevem objetos astronômicos disponibilizado pelo PLAsTiCC. Esse conjunto contém observações e informações sobre fontes astronômicas geradas a partir de simulações computacionais baseadas em observações reais. Espera-se que um classificador treinado a partir desses dados simulados seja uma boa aproximação para dados observados no LSST.

Para cada observação, é registrado, entre algumas propriedades, uma medição do brilho do objeto simulado. A composição do brilho de um mesmo objeto ao longo de um período de tempo é denominada de curva de luz. O método de mensurar o brilho de um objeto astronômico é conhecido como fotometria (Seção 2.1). Maiores detalhes sobre esse método são descritos na seção 2.1 do Capítulo 2.

Por meio da fotometria é possível identificar um número maior de astros gastando menos tempo. Entretanto, há a desvantagem de carregar pouca informação se comparado com outra técnica chamada de espectroscopia. Enquanto a fotometria afere o fluxo de uma fonte astronômica, a espectroscopia obtém aspectos físicos como temperatura, densidade e a composição do objeto (KEPPLER; SARAIVA, 2014). Entretanto, os dados da competição são simulações de observações captadas através da fotometria, o que dificulta o processo de classificação citado na Seção 2.1.

Em resumo, os principais desafios para aprendizado automático de classificadores usando dados de fotometria são: identificar características específicas de cada classe a partir apenas do brilho; ausência de muitos dados; em um dado período o objeto em estudo não aparece no céu; baixa quantidade de dados de treino; alto número de classes de objetos; existência de objetos cuja classe é desconhecida no conjunto de treino; distribuição desigual das classes no conjunto de treino; características variantes entre objetos no conjunto de treino e teste mesmo na mesma classe; e amostragem tendenciosa.

Neste trabalho de conclusão de curso foram estudadas técnicas de pré-processamento e treinamento utilizadas pelas principais abordagens dos melhores competidores do PLAsTiCC. As técnicas de pré-processamento estudadas foram: preparação dos dados, através do tratamento dos valores nulos e da identificação e remoção dos valores extremos, e redução de dimensionalidade, através da construção de características. As técnicas de classificação estudadas foram: as etapas básicas de aprendizado e classificação, empregando o aprendizado supervisionado, devido à natureza do problema, e árvores de decisão simples e modificadas para efetuarem a segunda etapa do processo; e métodos *ensamble* e *boosting*.

O desempenho das técnicas estudadas foram medidas a partir dos métodos de *holdout* e *random subsampling* para cálculo de acurácia; seleção de modelos através da taxa de erro como métrica para escolha do melhor modelo.

Dentre as técnicas testadas neste trabalho de conclusão de curso, a mais bem sucedida no PLAsTiCC foi a árvore de decisão modificada quando seus parâmetros proporcionaram um alto nível de *boosting* balanceado com os limites de profundidade do modelo gerado e baixa taxa de aprendizado.

Este trabalho está organizado da seguinte forma. No capítulo 2 é apresentado com maiores detalhes os conceitos relacionados com as fontes astronômicas e sua propriedades, os conceitos computacionais e métricas de aprendizado de máquina, a competição PLAsTiCC e o conjunto de dados.

No capítulo 3 é relatado todo o processo desenvolvido durante esse trabalho, desde a análise inicial do conjunto de dados da competição até a avaliação do modelo de classificação final. Esse processo abrange toda a evolução da solução e a organização estrutural

do código.

No capítulo 4, é possível entender como estão distribuídos os dados do conjunto e perceber características e comportamentos antes desconhecidos. Nele é apresentado os resultados obtidos pela análise de dados e alguns disponíveis na competição. Por último, são abordados os resultados dos testes realizados de *holdout*, *random subsampling* e de seleção de modelos através da taxa de erro calculada pelo Kaggle.

Finalmente, o capítulo 5 apresenta as conclusões e possibilidades de linhas de estudos futuros que podem ser interessantes para estender este trabalho.

2 Fundamentação Teórica

Neste capítulo são apresentados os conceitos básicos relacionados a astronomia e a ciência da computação. Inicia-se com conceitos relacionados à astronomia tais como: fotometria, espectroscopia, intensidade específica e as propriedades astronômicas das observações e das fontes astronômicas compreendidas nos dados da competição. Posteriormente são apresentados os seguintes classificadores e conceitos computacionais: uma visão geral sobre mineração de dados, aprendizado de máquina, aprendizado supervisionado e classificação de dados, em seguida, é descrito os algoritmos de classificação árvore de decisão e o LGBM, adiante, os métodos de preprocessamento e classificação utilizados e por fim, as tecnologias utilizadas. Prosseguindo, a competição PLAsTiCC é apresentada de modo abrangente e, logo após, o conjunto de dados é descrito em cada arquivo que o compõe, diferenciado entre treino e teste, e detalhado cada propriedade de suas tabelas. Por fim, os trabalhos relacionados que propuseram abordagens que resultaram as melhores colocações na competição do Kaggle são descritos.

2.1 Astronomia

Astronomia é a ciência que estuda o Universo, seus astros e os fenômenos que neles ocorrem. É frequentemente considerada a mais antiga das ciências (KEPPLER; SARAIVA, 2014). Muito importante para a humanidade em tempos remotos como, por exemplo, estabelecer estações, prever a melhor época para o plantio, servir de bussola, a astronomia foi fundamental para a surgimento do problema abordado nesse trabalho. Foi através dela que se coletou todos os dados para a construção dos dados simulados e além disso, o grande telescópio LSST é um ferramenta para essa ciência. Nessa seção é explicado alguns conceitos astronômicos para entender as propriedades dos objetos observados e algumas características tanto dos objetos quanto das observações. Todos o conceitos apresentados nas seções seguintes foram retirados do livro Astronomia e Astrofísica (KEPPLER; SARAIVA, 2014).

2.1.1 Propriedades das Observações

Uma das técnicas para realizar a observação de um objeto é a espectroscopia. A espectroscopia é o estudo da luz através das cores que a compõem, obtidas através da difração de um feixe de luz proveniente do objeto observado através de um prisma ou uma rede de difração. Essa técnica forma o espectro do objeto em questão onde é possível obter de forma direta ou indireta quase todas as informações sobre as propriedades físicas do astro.

Outra técnica para observar a luz de um objeto é a fotometria, que mede a quantidade de radiação eletromagnética emitida por um objeto. Ao se considerar uma frequência visível para efetuar as medições, a quantidade de luz ou brilho é o que será mensurado. Anterior à invenção do telescópio o único meio de observação, portanto, mais significativo, era o olho humano, capaz de observar somente espectro de luz visível entre as cores vermelho e violeta. Devido a esse fato, o meio mais utilizado para aferir medições é o da fotometria. Atualmente, com o desenvolvimento de equipamentos tecnológicos avançados é possível realizar medições por todo o espectro eletromagnético, que vai desde a radiação gama até as ondas de rádio.

O dado medido através da Fotometria é chamado de fluxo. O fluxo é a energia por unidade de área por unidade de tempo que chega ao detector. O fluxo da radiação de um objeto cai com o quadrado da distância, assim, a quantidade de energia que chega na Terra é muito menor do que a energia que na superfície do astro. Ele também pode ser medido em um intervalo definido de frequência. Esses intervalos são chamados de *passbands* e são obtidos através de um filtro que possibilita a passagem de certas faixas de frequências. Uma sequência de fluxos, mensuradas em dado intervalo de tempo e de frequência de um mesmo objeto, compõem a chamada curva de luz desse objeto. A partir das curvas de luz é possível identificar astros e fenômenos como, por exemplo, supernovas e cefeidas.

2.1.2 Propriedades das Fontes Observadas

Para localizar os astros no céu, é preciso estabelecer um sistema de coordenadas. Dois sistemas são demarcados no conjunto de dados da competição através das coordenadas celestes e as coordenadas galácticas.

As coordenadas celestes usa a ideia das coordenadas geográficas, longitude e latitude (CDA-USP, 2000). Define-se uma grande esfera terrestre que contém em seu centro a esfera terrestre. Ao projetar o equador terrestre no firmamento é gerado o equador celeste. A prolongação do eixo de rotação da terra forma os polos celestes. Conhecendo os polos e o equador é fácil definir as linhas de latitudes. Os meridianos são linhas que vão do polo norte ao polo sul, podendo ser entendido como a projeção da longitude na esfera celeste. Foi necessário definir um meridiano como a longitude de 0^o, assim como na terra tem-se o meridiano de Greenwich. No céu determina-se um ponto vernal ou ponto γ (gama) como sendo a origem. Ele corresponde a intersecção do Sol com o equador celeste quando este passa do hemisfério celeste sul para o norte.

A declinação de um astro é definido pelo ângulo formado entre o equador celeste e o astro medido sobre seu meridiano. As declinações do hemisfério norte são positivas e do hemisfério sul negativas. A ascensão reta de um astro é o ângulo entre o ponto γ e o meridiano do astro sobre o equador celeste, no sentido para o leste.

As coordenadas galácticas são um sistema de coordenadas celestes cujo plano fundamental é o plano do disco da Via Láctea. Portanto, seus polos norte e sul não coincidem com os da Terra, assim como a longitude e a latitude tem uma variação com as coordenadas celestes.

O avermelhamento presente no conjunto de dados da competição, medido através da espectroscopia (*hostgal_specz*) ou da fotometria (*hostgal_photoz*), é a mudança da frequência dos fluxos devido ao efeito Doppler (CANALLE; MATSUURA, 2007). Quando um astro se aproxima ou afasta da Terra ocorre um desvio do espectro da sua radiação. Quando ele se aproxima, o deslocamento ocorre para comprimentos de ondas mais curtos. É dito que o deslocamento é para o azul. Quando o astro se afasta ocorre o oposto, o deslocamento para o vermelho, chamado de avermelhamento ou em inglês *redshift*.

Na atmosfera, existem vários componentes que difundem a luz em todas as direções como, por exemplo, moléculas, partículas sólidas de poeira e fumaça, causando uma extinção contínua, em todos os comprimentos de onda (KEPPLER; SARAIVA, 2014). Quanto maior o caminho que a luz tem de percorrer na atmosfera, maior será sua extinção.

Além da extinção atmosférica, existe também a extinção interestelar. Ele ocorre devido à poeira interestelar concentrada principalmente no plano da Galáxia e que também extingue e avermelha a luz dos astros (KEPPLER; SARAIVA, 2014).

2.2 Conceitos Computacionais

Data mining ou mineração de dados é a extração de conhecimento a partir de dados (HAN MICHELINE KAMBER, 2012), que tem relação com à essência do desafio da competição. É substancial aprender através dos dados de treino, extrair algum conhecimento e ainda ser capaz de aplicá-lo corretamente a outros dados, neste caso, os dados de teste. Posteriormente, esse processo é avaliado sendo dito quão bem sucedido ele foi ou o quanto foi capaz de aprender e aplicar este conhecimento. A avaliação dos resultados dessa pesquisa é feita através de um sistema de pontuação do site Kaggle que, após processar a resposta, retorna um *score* e posiciona o time ou o competidor em um sistema de *ranking*.

O aprendizado e sua aplicação é sucedida através do reconhecimento de padrões, para definir classes de objetos que se parecem e distribuir novos objetos dentre essas classes. Eles podem inicialmente estar definidos em uma pequena amostra de objetos, caracterizado como um aprendizado supervisionado, ou podem ser totalmente desconhecidos, aprendizado não supervisionado. Em ambos é necessário definir para os novos objetos a qual classe ele pertence (ZAKI; JR., 2013). Definida essa tarefa, usaremos os algoritmos para obter classificadores para realizá-la. Tais algoritmos estão presentes em *Machine Learning*, campo de pesquisa voltado para programas de computadores que automaticamente

aprendem a reconhecer padrões complexos e fazem decisões baseadas nas características de objetos (HAN MICHELINE KAMBER, 2012).

Classificadores são funções ou modelos que recebem características de um objeto como entrada e retornam uma classe ou a probabilidade daquele objeto pertencer a uma ou várias classes. Eles são construídos com base em algum método específico, como árvores de decisões ou redes neurais artificiais (HAN MICHELINE KAMBER, 2012), variando seus parâmetros de configuração dos classificadores para cada problema.

Os algoritmos de aprendizado de classificadores utilizam um conjunto de treino com objetos classificados para treinar, reconhecer os padrões de cada classe e ser capaz de predizer qualquer novo item com um certo grau de exatidão. Tal exatidão depende da precisão e da acurácia do modelo que varia de acordo com *dataset* de treino, dependendo da quantidade de elementos classificados, sua representatividade em relação aos dados que podem ser classificados, dos parâmetros definidos na construção do algoritmo e das características extraídas dos objetos.

2.2.1 Árvores de Decisão

Dentre os algoritmos classificadores adotados, um deles é a árvore de decisão. Essa estratégia de classificação funciona da seguinte forma: seja um conjunto de dados de treino C de dimensionalidade arbitrária que será particionado dentro do espaço de dados por um hiper plano gerando dois subconjuntos C_1 e C_2. De modo recursivo, cada uma dessas regiões também serão particionadas até que a maioria dos elementos desse subconjunto sejam da mesma classe, formando assim uma folha da árvore. A estrutura de divisões do conjunto resultante é o modelo de árvore de decisão. Para predizer a classe de um novo elemento, deve-se avaliar recursivamente a qual divisão ele pertence, começando pelo conjunto total, até que atinja uma folha da árvore, concluindo que a classe desse novo objeto é a mesma da folha que ele atingiu. (ZAKI; JR., 2013)

2.2.2 Light Boosting Gradient Model

O segundo algoritmo aplicado é o *Light Boosting Gradient Model* (LGBM). O LGBM é um algoritmo baseado em árvores de decisão que aplica um *boosting* gradiente. Foi designado a ser distribuído e eficiente com as seguintes vantagens: alta velocidade e eficiência no treinamento; baixo uso de memória; melhor acurácia; suporte para aprendizado paralelo e em GPU; e capaz de ligar com grandes escalas de dados (MICROSOFT, 2019a).

O *boosting* presente no LGBM consiste em utilizar várias cópias do classificador para aprenderem sobre o conjunto de dados de forma interativa. Para classificar um resultado, é dado um peso para cada predição desses classificadores e então é a calculada

a predição final. Para o aprendizado ocorre algo parecido. Cada tupla de treinamento, formado por um dado e sua classe, possui um peso. Assim que o primeiro classificador executa o processo de aprendizado, os pesos são alterados de modo a aumentar a atenção para as tuplas classificadas erroneamente por esse classificador. Assim, o próximo terá maiores chances de acertar o que o primeiro classificador e assim sucessivamente. O último classificador impulsionado (*boosted*), combina os votos de cada classificador individualmente, onde os pesos de cada classificador é uma função de sua acurácia (HAN MICHELINE KAMBER, 2012).

Quando voltado para classificar múltiplas classes, o LGBM pode utilizar algumas métricas, e uma delas é através da função objetivo *softmax* (MICROSOFT, 2019b). Essa função retorna um vetor que representa a distribuição de probabilidades de uma lista de potenciais resultados. A soma das probabilidades desse vetor é igual a 1.

2.2.3 Métodos de pre-processamento e classificação

As técnicas de pré-processamento aplicadas neste trabalho foram de limpeza e redução de dados. Na limpeza dos dados, foram tratados os valores ausentes nos dados e os chamados *outliers*. Os valores ausentes foram substituídos, após o processo de redução dos dados, pela tendência central (mediana) da propriedade que contém tais valores (HAN MICHELINE KAMBER, 2012). Os *outliers* são dados ruidosos que são distintos dos demais dados, se distanciam de sua distribuição geral (HAN MICHELINE KAMBER, 2012). Os *outliers* foram retirados para a análise dos dados através da seleção de um quantil de 90% do dados, removido os 5% menores dados e os 5% maiores. Entretanto, os *outliers* foram mantidos para o treinamento e classificação.

Para a redução de dados, foi aplicado a redução de dimensionalidade e em número, através da seleção e criação de atributos (HAN MICHELINE KAMBER, 2012). A redução de dados é o processo de reduzir os dados para um espaço menor que o original mantendo a integridade do mesmo. A redução de dimensionalidade envolve a escolha de atributos dos dados que serão considerados no processo de classificação, enquanto a redução de numerosidade substituem o volume dos dados por uma forma alternativa de representá-los. Podendo manter apenas a estimativa dos dados em vez dos dados reais.

Em conjunto com as técnicas de pré-processamento, também foram aplicadas algumas técnicas de classificação. Para o aprendizado supervisionado foi utilizado os algoritmos de árvore de decisão e LGBM, previamente descritos, e, para a avaliação das predições e a selecionar o melhor modelo, foi utilizado o *random subsampling* e a taxa de erro.

O *random subsampling* é uma variação do *holdout* o qual este método é repetido k vezes e o resultado final é a média das acurácias obtidas em cada interação (HAN MICHE-

LINE KAMBER, 2012). O *holdout* calcula a acurácia de um classificador. Nesse método, é passado um conjunto de dados quaisquer, corretamente rotulado, que é particionado em dois conjuntos independentes, um de treino e outro de teste. O conjunto de treino é utilizado para o aprendizado do modelo classificador que será avaliado pelo conjunto de teste. Essa avaliação é pessimista, pois, apenas uma porção do conjunto de treino é utilizada para o aprendizado do modelo.

Outro método que foi muito utilizado pelos competidores no Kaggle foi o *k-fold cross-validation*. Nesse método, os dados iniciais são aleatoriamente particionados em k subconjuntos, os *folds*, mutuamente exclusivos e de tamanhos aproximados. Treino e teste são executados k vezes. Na interação i, o subconjunto i é reservado para o teste enquanto todos os outros são aplicados no treinamento. Assim, todos os *folds* são utilizados para treinamento o mesmo número de vezes e em uma interação para o teste. Para classificação, a acurácia estimada é o número total de classificações corretas durantes as k interações, divido pelo número de tuplas nos dados iniciais (HAN MICHELINE KAMBER, 2012).

Entretanto, existe uma grande diferença entre o resultado do *cross-validation* com os resultados obtidos na competição (GRELLIER, 2019a) devido ao conjunto de dados de treino tendencioso que leva ao chamado *overfitting*. O *overffiting* é um erro de modelagem no qual a função não é capaz de lidar com certos ruídos dos dados reais ou não consegue generalizar um padrão para reconhecer uma certa classe em diferentes ambientes (TAN; KUMAR, 2005).

Para a seleção através da comparação entre modelos, foi utilizado apenas a média da taxa de erro, que é igual ao oposto da acurácia média, 1 - acurácia(M), para um modelo M. Apesar dessa média ser apenas uma estimativa do erro real da classificação dos dados futuros (HAN MICHELINE KAMBER, 2012), ela foi utilizada, pois, é através de um conjunto de teste específico que a competição no Kaggle calcula a taxa de erro de um modelo. Sendo assim, se um modelo resulta em uma taxa de erro menor que outro modelo, nesse trabalho, o primeiro foi considerado melhor.

2.2.4 Tecnologias Utilizadas

As tecnologias utilizadas foram: a linguagem Python 3, alguns pacotes suportados por ela e dois tipos de arquivos de leitura e escrita. O Python foi escolhido por ter uma grande comunidade ativa relacionado ao aprendizado de máquina e ciência de dados e pela fácil integração de bibliotecas que facilitaram todo processo. Os pacotes utilizados foram o *pandas*, *numpy*, *Matplotlib*, *seaborn*, *scikit-learn*, *lightgbm*, as classes DataFrame, *tree*, *SimpleImputer* e, por último, os tipo de arquivos *h5* e CSV.

O pacote *numpy* (NUMFOCUS, 2019a) foi em grande parte utilizado para cálculos numéricos em vetores do tipo *numpy.array*, que tem uma velocidade maior de processa-

mento do que listas, por exemplo. O *pandas* (NUMFOCUS, 2019b) oferece leitura e escrita de arquivos do tipo CSV e *h5* transformando-os em objetos DataFrame. Esses objetos são similares a tabelas de banco de dados e possuem funções de busca e manipulação de dados que, em conjunto com o *numpy*, são rápidas. Além disso, oferece a criação de imagens e gráficos das tabelas através do pacote *Matplotlib* (HUNTER, 2007). O pacote *seaborn* também foi utilizado para a criação de gráficos. O *scikit-learn* contém algoritmos classificadores, métricas para cálculos de pontuação das predições, a classe *SimpleImputer* para preenchimento de valores nulos e alguns outros componentes para a construção e validação dos modelos classificadores. Por útlimo, o pacote *lightgbm* possui o algoritmo LGB e suas funções de aprendizado e treinamento, de classificação e outras métricas que não foram utilizadas nesse trabalho.

2.3 PLAsTiCC

O *Photometric LSST Astronomical Time-Series Classification Challenge* (PLAsTiCC), é uma competição para a comunidade científica desenvolver algoritmos que classifiquem eventos astronômicos, transientes e variáveis. O *Large Synoptic Survey Telescope* (LSST) irá descobrir dezenas de milhares de fenômenos transientes todas as noites e para lidar com essa grande quantidade de informações, algoritmos que classifiquem e ordene esses eventos e objetos, são cruciais.

O PLAsTiCC está hospedado no *site* de competições Kaggle. Nele, é possível encontrar diversas competições de diversos níveis de dificuldade na área de aprendizado de máquina. Os tipos mais comuns de competição são: pesquisa (*Research*), para quem está começando (*Getting Start*), para diversão (*Playground*) e competições com desafios completos de *machine learning* e, geralmente, envolve problemas de predição com propósitos comerciais (*Features*). A competição abordada nesse trabalho é classificada como *Feature Prediction Competition*, devido às suas características, e possuía uma premiação de 25,000.00 dólares distribuídos entre os 3 melhores times. O PLAsTiCC encerrou as submissões no dia 17 de dezembro de 2018 tendo como primeiro lugar o Kyle Boone (Seção 2.5).

Ainda no site do Kaggle, é possível fazer o *download* de *datasets* das competições que podem estar nos formatos de CSVs, JSON, em arquivos compactados, entre outros. Os dados necessários para a resolução do problema em questão também está na própria competição e foram adquiridos pelo Kaggle. No *site*, além das competições e dos *datasets*, é encontrado os chamados *Kernels*. Os *Kernels* são ambientes de computação em nuvem, capazes de reproduzir e corroborar análises (INC., 2019). Eles ajudam a entender como outros competidores estão trabalhando em suas pesquisas e a estar envolvido na comunidade Kaggle. Segundo o próprio site (INC., 2019), não há lugar melhor que

os *Kernels* do Kaggle para descobrir tamanho repositório público, de código aberto e de código reproduzível para *data science* e *machine learning*.

Há, porém, algumas regras a serem seguidas, tanto da competição quanto do próprio Kaggle, encontradas no Kaggle. Algumas regras interessantes são as obrigatoriedades dos ganhadores e as limitações para uso de dados externos à competição, ambas estabelecidas pelo PLAsTiCC. As predições que ganharam a competição e o código que as gerou devem ser *open-source* e o uso de dados externos devem ser previamente requisitados e autorizados pela competição.

Por fim, para determinar quais as melhores predições, cada uma delas é avaliada por uma função de perda logarítmica para múltiplas classes. O resultado é que cada classe tem a mesma importância para a pontuação final. Na competição (LSSTC, 2019c) é possível encontrar a fórmula utilizada na avaliação e uma breve descrição. Há ainda uma observação a respeito da não obrigatoriedade da soma das probabilidades para um dado objeto ter soma igual a 1, pois, as probabilidades são reescaladas. Cada linha de predição é dividida pela sua soma e são limitadas inferior e superiormente de modo a evitar os extremos do logaritmo.

2.4 Conjunto de Dados do PLAsTiCC

O *dataset* oferecido no PLAsTiCC tem grande peso em como o classificador é construído. Nele está contido as principais informações que é preciso para treinar e testar o classificador. Milhões de linhas de informações foram simuladas com base em observações astronômicas registradas para representar os dados reais de forma fiel. Os objetos simulados são divididos em eventos transientes, que ocorrem uma única vez e não voltam a se repetir como supernovas, e variáveis, variam seu brilho em ciclo repetitivo em um certo período como o caso das Cepheids (JR et al., 2018). Objetos que se movem no céu como asteroides foram descartados. Na competição e no artigo do PLAsTiCC é possível obter mais informações sobre o que está contido nos conjuntos de dados, quais as propriedades das tabelas e o que elas representam, além de discussões e artigos escritos pelos competidores no Kaggle, chamados pelo *site* de *kernels*. Nestes textos é possível encontrar dúvidas, artigos, códigos, explicações acerca dos conhecimentos que a competição envolve, entre outros assuntos. Parte dessas informações é descrita na seção 4.2 de análise dos dados.

Como descrito em (JR et al., 2018), o *dataset* é composto por 4 tabelas, separadas por treinamento, teste e ambas divididas em metadados e dados, as observações. As tabelas de treinamento são usadas para a preparação do classificador enquanto as de treino são utilizadas para avaliá-lo através da competição usando tanto os metadados quanto os dados para tal. As tabelas de metadados, também chamadas de cabeçalho,

possuem informações astronômicas dos objetos, que podem ser utilizadas para se ganhar algum conhecimento extra, e as tabelas de dados possuem os dados propriamente ditos, as curvas de luz, usados no classificador. Por último, as tabelas de dados contém as informações de curvas de luz de cada objeto composto por séries temporais de fluxos em seis filtros, incluindo ainda erros de fluxos neles próprios.

As tabelas de cabeçalho, inclui as propriedades de cada fonte observada e associando os objetos, presentes em ambos os conjuntos, através de um identificador único, o *object_id*. A descrição de cada propriedade é apresentada na Tabela 1.

O segundo tipo de tabela, chamada de tabela de dados ou *data files*, possui informações sobre os objetos astronômicos, o *object_id*, presente na tabela de cabeçalho para que possam ser associados, e o valor de seu brilho em diferentes faixas de frequência (*passbands*) que, em função do tempo, constituem as curvas de luz. Cada linha da tabela corresponde a uma observação feita em um certo tempo utilizando um determinado *passband*. A relação de observações de DDF e WFD é de 1 para 11 respectivamente, enquanto a área entre os dois no LSST será de 1 para 400. As informações incluídas são descritas na Tabela 2:

2.4.1 Ressalvas

Três ressalvas sobre os dados são citadas no artigo do PLAsTiCC (JR et al., 2018). A primeira é sobre as grandes lacunas de tempo sem observações devido as limitações físicas do método utilizado. Como o telescópio é fixado em um lugar na Terra, ele pode não conseguir captar imagens de um astro em determinadas épocas do ano. Também, algumas noites podem apresentar nuvens que bloqueiam a passagem da luz dos astros. A segunda ressalva é que os objetos galácticos, presentes na Via Láctea, são caracterizados por apresentarem um *redshift* igual a 0. E a terceira e última é a presença de fluxos negativos. Que devido ao método de estimar o brilho das fontes astronômicas e as variações estatísticas, o fluxo do objeto pode ser mensurado com um valor negativo, quando na verdade seu valor é próximo a 0. O fluxo mensurado é subtraído por um valor padrão para cada *passband* para cada região do céu, por essa razão poder haver fluxos negativos nos dados (TEAM, 2019).

2.5 Trabalhos Relacionados

Os trabalhos usados como referencial para este trabalho de conclusão de curso foram os melhores 20 colocados com as discussões mais avaliadas e referenciadas do PLAsTiCC. Ao todo foram 1094 soluções submetidas na competição.

Dentre eles, foram escolhidos os trabalhos que disponibilizaram os melhores textos, com explicações e códigos de seus modelos. Destacando-se os trabalhos do primeiro

Tabela 1 – Tabela com a descrição das propriedades dos metadados da competição

object_id:	Identificador único que relaciona os objetos descritos nos dados e no cabeçalho;
ra:	Ascensão reta (*right ascension*), coordenada celeste (Seção 2.1.2). Corresponde à longitude celeste, medida pelo angulo formado entre o meridiano da fonte astronômica e o ponto γ (Gama), sobre o equador celeste e no sentido para o leste;
decl:	Declinação (*declination*), coordenada celeste (Seção 2.1.2). Corresponde à latitude celeste, medida pelo ângulo formado entre a fonte e o equador celeste sobre o meridiano da fonte;
gal_l:	Longitude galáctica (Seção 2.1.2), distância angular entre a fonte e o centro galáctico, situado sobre a constelação *Sagittarius*, sobre o plano formado pela Via Láctea (Unidade de medida em graus);
gal_b:	Latitude galáctica (Seção 2.1.2), ângulo do objeto com o equador galáctico medido em graus e variando de 0 a 90º para o Norte e de 0 a -90º para o Sul;
ddf:	Variável *booleana* que representa em qual área de pesquisa o objeto foi observado. Se igual a 1, o objeto foi advém da área DDF, se 0, então ele advém da área WDF. É considerável que, apesar da área de DDF estar contida dentro da área de WFD, ela contém erros significativamente menores;
hostgal_specz:	*Redshift* ou envermelhamento (Seção 2.1.2) do objeto captado pela espectroscopia. É extremamente preciso e é voltado para o treinamento do classificador, sendo assim, apenas uma pequena parcela do conjunto de teste possui esta informação;
hostgal_photoz:	*Redshift* (Seção 2.1.2) do objeto captado pela fotometria. Ele deveria ser semelhante ao *hostgal_specz* porém a incerteza, o erro, desta medida é bem maior;
hostgal_photoz_err:	Grau de incerteza do *hostgal_photoz* baseado nas projeções de pesquisa do LSST;
dist_mod:	Distância, em módulo, da fonte de brilho calculada através do *redshift*, armazenado pela propriedade *hostgal_photoz*;
MWEBV:	Equivalente a MW E(B-V), é uma propriedade da Via Láctea que indica quanto de luminosidade é absorvida, na área onde o objeto está localizado, pela poeira da nossa galáxia. Ela é usada para determinar o escurecimento dependendo do *passband* bem como o avermelhamento da fonte astronômica;
target:	Classe a qual a fonte pertence, preenchido somente nas tabelas de treinamento. Encontrar a classe correta para as fontes é o objetivo das tabelas de testes.

Tabela 2 – Tabela com a descrição das propriedades dos dados da competição

object_id	Identificador único de cada objeto;
mjd	A data de quando foi feita a observação em *Modified Julian Date* (MJD). As unidade de medida são dias e o ponto de ínicio da medição é datada na meia noite do dia 17 de novembro em 1957;
passband	Marca a faixa de frequência em que a observação foi feita. Sendo um número inteiro de 0 a 5 inclusive, cada um representando os *passbands* u, g, r, i, z, y respectivamente;
flux	Fluxo da fonte, ou o brilho, captada em uma das passbands. Ele é corrigido pelo MWEBV mas quando os níveis de absorção de luz devido à poeira, a incerteza se torna maior que a correção;
flux_err	Incerteza do fluxo medido;
detected	Booleano que é 1 quando o objeto tem seu brilho a uma distância maior que 3sigma relativo ao modelo de referência, ou seja, difere do modelo mais que 3 vezes o desvio padrão;

colocado, Kyle Boone (BOONE, 2019b), do Jean François (PUGET, 2019a), quinto colocado, e da Belinda Trotta (TROTTA, 2019a), 14^a colocada. Esse três autores usaram o modelo de classificação LGBM(*Light Gradient Boosting Model*). Também utilizaram esse modelo os (ERDEM, 2019; LIU, 2019; GARRETA, 2019; CHEN, 2019) proponentes das aborgagens da 4^a, 8^a, 9^a e 11^a posições na competição respectivamente. O grande número de competidores nas melhores posições usando esse algoritmo reforça a escolha do LGBM para ser usado como um dos algoritmos classificadores avaliados neste trabalho.

2.5.1 Abordagem do Kyle Boone

Kyle Boone (BOONE, 2019b) focou em classificar as supernovas, resultando em um excelente trabalho. Na sua abordagem as supernovas e suas variações foram corretamente classificadas e outras classes são preditas por consequência.

Em síntese, sua solução envolve quatro grandes aspectos: aplicar técnica para aumentar quantidade de dados sobre o conjunto de treino degradando as curvas de luz para se aproximar com as propriedades do conjunto de teste; aplicar o processo gaussiano para predizer as curvas de luz; extrair características dos dados tratados e não tratados; e treinar um único modelo LGBM com 5-*fold cross-validation*.

Boone empregou a regressão do Processo Gaussiano para extrair as características de cada objeto. Esse procedimento levou cerca de 3 dias para o processamento e gerou modelos para as curvas de luz bem amostradas. Além de serem para os 6 diferentes *passbands*, os modelos se saíram muito bem apesar das incertezas com um *flux_err* elevado, detalhes sobre os atributos dos objetos na Seção 2.4. Boone também mostra um exemplo de como são os modelos na Figura 1.

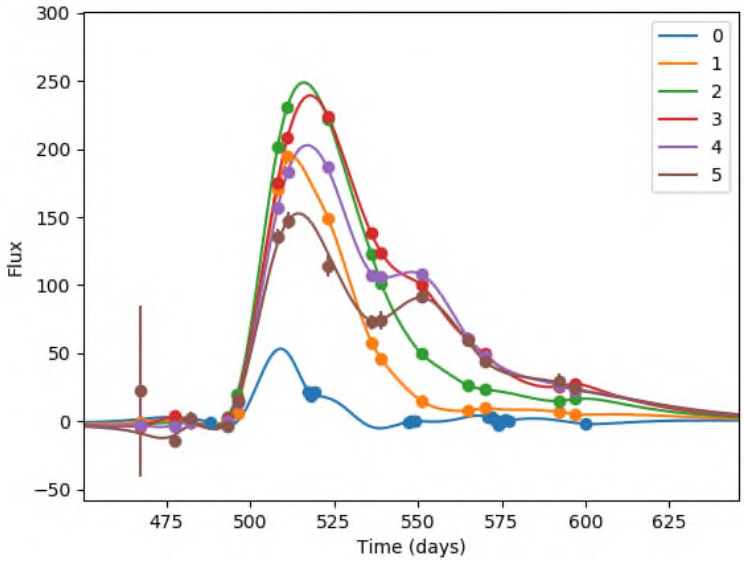

Figura 1 – Exemplo de uma curva de luz usando a regressão pelo Processo Gaussiano.

Fonte: https://www.kaggle.com/c/PLAsTiCC-2018/discussion/75033 (Junho 2019)

As características que diferenciam as supernovas, segundo o autor, são os picos de brilho e a largura de sua curva de luz, portanto, diversas características que representam tais propriedades foram colocadas em seu modelo de predição. Ele também inclui características que permitem dizer quão bem o GP construiu a curva de luz, uma vez que em amostras pouco volumosas o GP não obtém grandes resultados. De modo a identificar classes que não são supernovas ele também usufruiu de uma detecção e contagem de picos.

Para lidar com a diferença entre o conjunto de treino e o conjunto de teste (O'BRIEN, 2019), as curvas de luz de cada objeto de treino foram modificadas 40 vezes para resultar em novas curvas de luz que se parecessem com as piores amostras curvas de luz do conjunto de teste. Inclui-se nesse processo: alteração no brilho dos objetos galácticos, que pertencem à nossa galáxia Via-Láctea; alteração do *redshift* dos objetos extragalácticos (incluindo dilatação do tempo e mudança de brilho); adicionados grandes intervalos sem observações como nos dados reais, que advém do momento da observação em relação à data do ano; escolhidos novos *hostgal_photoz* e *hostgal_photoz_err* baseado na relação entre *hostgal_specz* e *hostgal_photoz* e simulada a detecção do objeto para escolher quais desses objetos serão incluidos no *dataset*;

Após feita essa modificação das curvas de luz, o *dataset* de treino passou a ter cerca de 270 mil objetos e se tornou mais representativo em relação aos dados de teste. Prosseguindo, Kyle Boone treinou um modelo LightGBM () com *5-fold cross validation* usando o conjunto gerado e certificou-se que as 40 variações, ou degradações de cada objeto, estivessem em um mesmo grupo. Como resultado, o modelo, após sintonizado (processo de *tunning*), obteve um CV (*Cross Validation*) em torno de 0,4 sobre o conjunto de treino original, com uma matriz de confusão representado pela Figura 2.

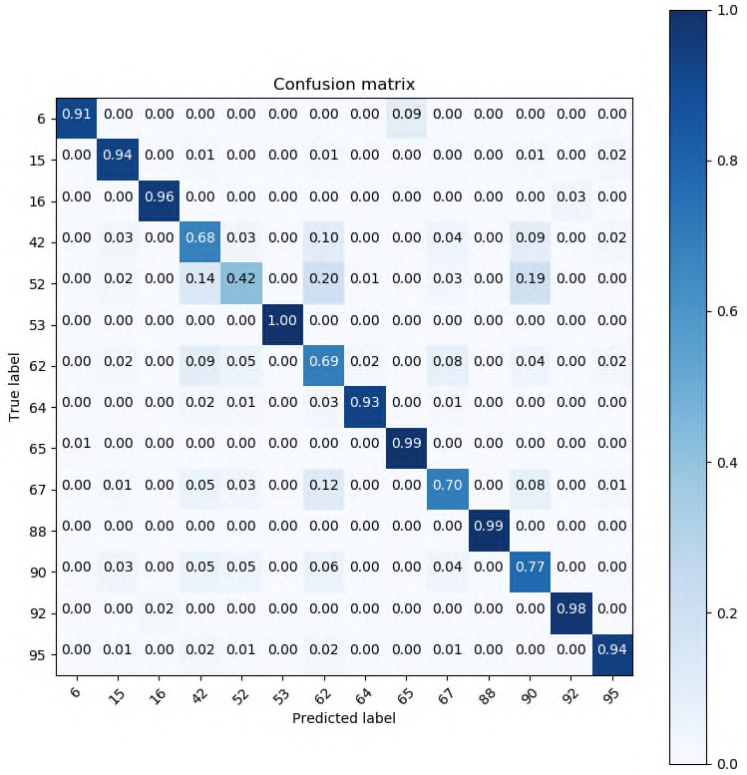

Figura 2 – Matriz de confusão de Kyle Boone obtida através do *cross-validation* das predições do LightGBM.

Fonte: https://www.kaggle.com/c/PLAsTiCC-2018/discussion/75033 (Junho 2019)

Finalmente, para contornar o problema de classificar os objetos da classe 99 que não estão presentes no *training dataset* foi escolhido um *flat score* (uma pontuação de valor única) para classificá-los. Usando isso, Boone obteve o que ele considera seu melhor *score* de 0,726 no *ranking* público de desempenho de classificadores. Tentando melhorar ainda mais seu modelo, Boone utilizou um método chamado de *probing*, que consiste em conseguir alguma informação dos resultados através da submissão de respostas e avaliando seus *scores*. Com isso ele obteve sua pontuação final de 0,670 no LB público.

2.5.2 Abordagem do Jean François Puget

De acordo com Jean François Puget (PUGET, 2019a), as principais dificuldades do PLAsTiCC em relação a outras competições são: séries temporais amostradas desigualmente; *Dataset* de treino tendencioso; *dataset* de treino pequeno; e dados de treino não possui todas as classes a serem preditas.

Antes de resolver esses problemas, Puget criou uma base com transformações das características dos dados das curvas de luz. Essas transformações foram computadas usando o Pandas pois, de acordo com Puget, os pacotes amplamente disponíveis *light gatspy*, *cesium* e *tsfresh* (VANDERPLAS, 2019; CESIUM, 2019; CHRIST; NEUFFER,

2019) para extração de dados não tiveram bom desempenho e tem alto custo computacional. Uma maneira de fazer essas extrações de forma eficiente pode ser encontrado em outra discussão aberta pelo próprio autor em (PUGET, 2019c).

Nesse pré-processamento, para cada *object_id*, as características foram extraídas por cada *passband* e olhado o fluxo completo do objeto

Algumas características dos dados criadas a partir das extrações e que provaram ser úteis são:

- Desvio padrão, assimetria e curtose. Com o objetivo de diferenciar as curvas com picos (supernovas) das outras. Também caracteriza um pouco a forma do pico;
- *Ratios*. Se calcula o *ratios* pela divisão do fluxo máximo de cada *passband* e de cada objeto e divide pelo maior dentre eles;
- Características baseadas na diferença de observações subsequentes (Δ);
- Magnitude. Com a dificuldade de estimar o fluxo absoluto visto que apenas temos acesso a diferença com um *template* desconhecido;
- Diferença Mjd (Seção 2.4); Propriedade muito importante como mostrado em (SIONKOWSKI, 2019)
- Fluxo normalizado pela divisão do fluxo máximo de cada objeto. A sua magnitude oferece a informação sobre a escala mais apropriada.
- Predição das curvas de luz. Foi usado as predições *Bazin* e *Newling* para preencher os valores nulos das observações entre a primeira observação e a última.

Com o intuito de aumentar o tamanho da base de dados para o aprendizado do modelo, Puget gerou variantes de cada elemento de treino usando a propriedade *flux_err*. A geração de variantes foi realizada com base em um ruído Gaussiano com o desvio padrão do *flux_err*.

Essa geração obteve 5 versões aleatórias de cada curva de luz, com o cuidado de sempre manter essas variações em *folds* iguais, obtendo uma melhora de 0,003 na pontuação do ranking de desempenho classificadores. Esse planejamento resultou em uma pontuação de 0,855 no ranking público incorporando os resultados de um modelo LGB com uma MLP (*Multilayer Perceptron*) treinada com um subconjunto das características geradas para o LGB.

Para contornar o problema das observações amostradas de forma desigual e com espaços de tempo sem nenhum dado foi feito a interpolação das curvas de luz. Puget tentou *Bazin*, *Newling* e processo Gaussiano. O processo Gaussiano foi mais promissor,

confirmado também pela solução do Boone (Seção 2.5.1). As Figuras 3, 4 e 5 são exemplos da interpolação de curvas de luz.

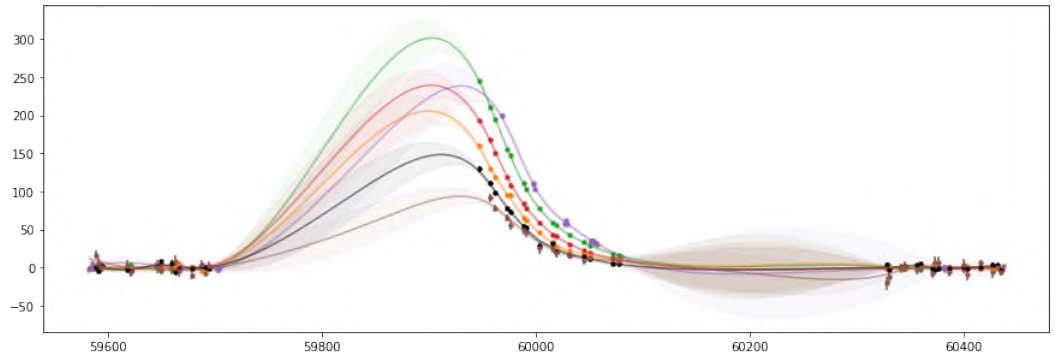

Figura 3 – Exemplo de interpolação de uma curva de luz utilizando Celerite.
Fonte: https://www.kaggle.com/c/PLAsTiCC-2018/discussion/75050 (Junho 2019)

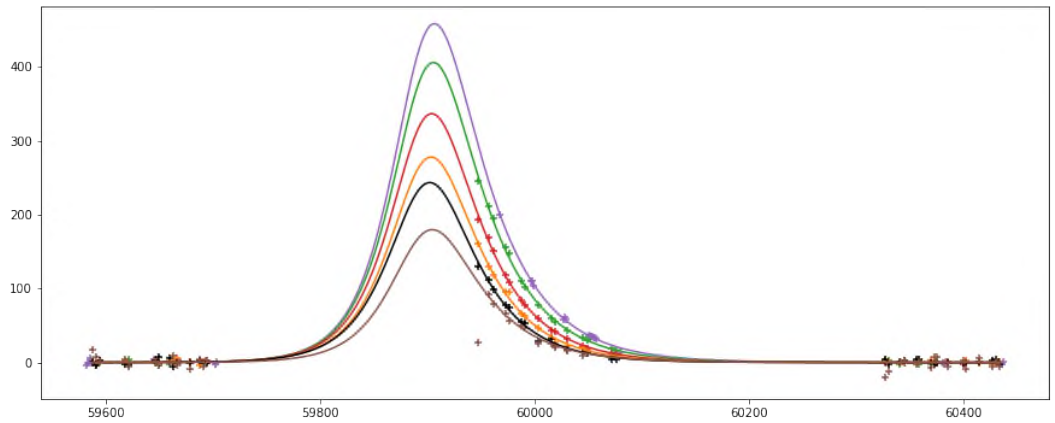

Figura 4 – Exemplo de interpolação de uma curva de luz utilizando Bazin.
https://www.kaggle.com/c/PLAsTiCC-2018/discussion/75050 (Junho 2019)

Pelo fato das distribuições das classes serem diferentes e por não conter nenhum elemento da classe 99 para o aprendizado, o conjunto de treino se torna tendencioso em relação ao de teste. Como exemplo, objetos observados na área de DDF são mais frequentes no conjunto de treino, porém, um ajuste com pesos não resultou em um ganho de acurácia. Outro atributo que influencia o viés do aprendizado é a propriedade *hostgal photoz*. Se comparado a distribuição dessa propriedade para cada classe com a do conjunto inteiro de treino, será visto uma distinção de duas classes das demais. O que não faz sentido no ponto de vista físico, segundo Puget, pois, a distribuição das classes pelo espaço teoricamente é uniforme. Para contornar este problema, essa propriedade foi trocada por um binário para identificar objetos galácticos e extragalácticos.

Uma possível solução para a pequena quantidade de objetos de treino é a criação de dados a partir dos existentes. Contudo, é um desafio fazê-lo sem tornar os dados mais

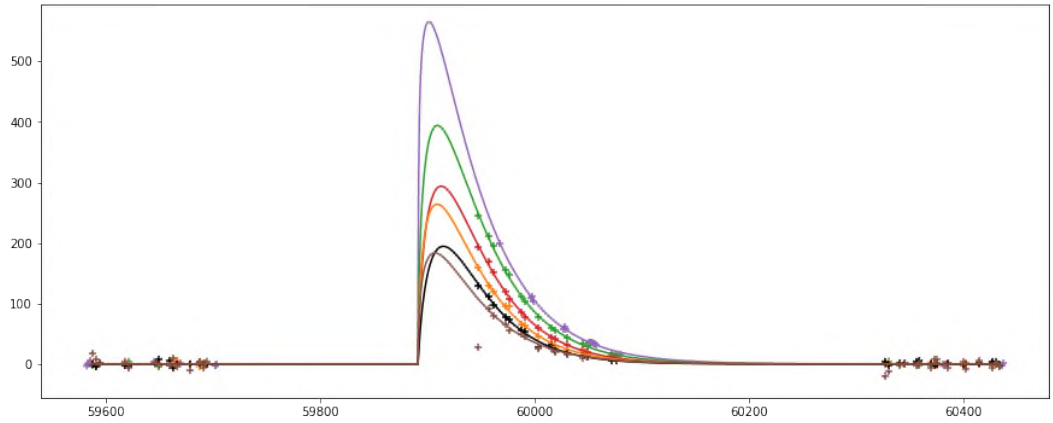

Figura 5 – Exemplo de interpolação de uma curva de luz utilizando Newling.
UrlFont: https://www.kaggle.com/c/PLAsTiCC-2018/discussion/75050 (Junho 2019)

tendenciosos. Usando ruído Gaussiano com o *flux_err* é fácil, porém, é necessário ter a curva de luz completa e interpolar curvas ao longo do tempo é complexo.

Para superar o desafio de classificar a classe 99 (Seção 2.4), foi utilizado a abordagem proposta pelo Olivier Grellier (GRELLIER, 2019b) que se baseia nas probabilidades resultantes de um dado objeto pertencer às outras classes.

Os métodos que Jean F. Puget aplicou em seu modelo e que não obtiveram melhora na pontuação dos resultados foram: *auto encoder*, detecção de anomalias, algoritmo KNN, rede neural convolucional, *k_corretion* e a validação cruzada 2.2.3. Esse último foi marcante, pois, muitos competidores (GRELLIER, 2019a) enfrentaram dificuldades para empregar a validação cruzada no treinamento para melhorar a acurácia e, ao mesmo tempo, evitar o *overfitting* do modelo. Além da diferença de distribuição de teste e treino tornando as pontuações resultantes da validação cruzada muito distantes da pontuação calculada pelo PLAsTiCC (GRELLIER, 2019a).

2.5.3 Abordagem da Belinda Trotta

A solução mais clara e aparentemente simples é a da Belinda Trotta (TROTTA, 2019a). Em um repositório github (TROTTA, 2019b) ela dispôs de um texto em forma de artigo explicando todo o desenvolvimento, os problemas encontrados e como eles foram tratados de forma precisa e bem estruturada. Além disso, Trotta dispõe todo o seu código usado para o tratamento dos dados e a construção do modelo de predição que utiliza, também, o algoritmo LGBM. O código muito bem estruturado e de fácil compreensão foi utilizado para o entendimento de algumas abordagens comuns entre os melhores competidores e para a predição da classe 99, explicada no capítulo 3 e que obteve maiores pontuações para a predição de tal classe.

Ao seguir os caminhos para a mineração de dados proposto pelo livro *Data Mining*

(HAN MICHELINE KAMBER, 2012), a estrutura obtida se pareceu mais com a estrutura construída pela Belinda Trotta. Por esse fato e pela sua qualidade, o seu trabalho foi de grande importância e considerado, por este trabalho de conclusão de curso, a melhor da soluções apresentadas.

2.6 Considerações finais

Esse capítulo contém todos os conceitos mais específicos relacionados ao problema e à solução abordados neste trabalho de conclusão curso. Ele colabora e muito o entendimento do que foi feito e foi fundamental para a busca de métodos que poderiam ser utilizados e de trabalhos relacionados que poderiam auxiliar na busca pela solução do problema.

3 Desenvolvimento

Antes de construir um classificador de dados e logo executar as predições, primeiro foi necessário entender o problema que o PLAsTiCC apresenta e deseja solução. Baseando-se nos artigos publicados (JR et al., 2018; TEAM, 2019) e nas discussões dos competidores (Seção 2.5) as tecnologias e uma solução foram selecionados. A linguagem escolhida foi o Python (Seção 2.2.4) empregando como solução um classificador com aprendizado supervisionado (Seção 2.2) que utiliza o algoritmo LGBM, abordado na Seção 2.2.2 dos Conceitos Computacionais.

Outros modelos, utilizando os algoritmos de árvore de decisão e o próprio LGBM, foram construídos com a finalidade de compará-los e executar alguns testes. Muitos dos competidores ainda utilizaram redes neurais como o algoritmo classificador, porém, os textos mais claros e explicativos com códigos mais compreensíveis utilizaram o algoritmo LGBM, em razão disso foi preferível esse último. Após definir a linguagem e o modelo, foi necessário definir a máquina que executaria todo o programa. Apenas duas foram utilizadas, a primeira é uma máquina física executando o sistema Windows 10 e, a segunda, uma máquina virtual executando Ubuntu 18.4 hospedada na nuvem. Apesar dos 64 GB de RAM, a máquina virtual não foi útil para esse trabalho e deixou de ser utilizada.

A primeira etapa para começar a construção do projeto foi analisar os dados do conjunto de treinamento em busca de características determinantes que ajudassem no processo de classificação. Para isso, cada sequência de observações de um mesmo objeto em cada *passband* foi redimensionada em uma única linha da tabela, que contém a média dos fluxos das observações. O que torna possível a construção de gráficos com um nível de abstração maior, com uma visão mais ampla da distribuição da base de treino. Uma alternativa a essa abordagem seria começar analisando os metadados que correspondem a uma entrada para cada objeto, que são mais fáceis de examinar. Contudo, uma vez que os *kernels* (APERS; NARAYAN, 2019; MELLO, 2019; BOONE, 2019a; PUGET, 2019b), contém análises como, a relação da quantidade de objetos por classe, a relação entre a medida do *redshift* através da fotometria e da espectroscopia, a localização das áreas de DDF e WDF em coordenadas galácticas, a separação das classes presentes na Via-Láctea das extragalácticas e o porquê de existirem lacunas de tempo sem observações em praticamente todos os objetos, foi preferível analisar diretamente os dados e seu comportamento.

Em seguida, foi construído um modelo simples de classificação utilizando um algoritmo de árvore de decisão (Seção 2.2.1) e passando como parâmetros, de treino e de teste, subdivisões do conjunto de treino. Utilizando apenas esse conjunto, foi possível alcançar uma acurácia de 59%, fixando uma relação de 2,5% e 97,5% do tamanhos dos conjuntos

de treino e de teste respectivamente. Mudando o processamento do conjunto de treino, melhor detalhado na Seção 4.3, para extrair mais características foi possível obter uma acurácia de 65% mantendo a relação de tamanho entre os subconjuntos gerados.

Apesar de não ser uma acurácia tão elevada, foi um resultado razoável. Portanto, mantendo a extração de características utilizadas no modelo de árvore de decisão, foi construído o modelo LBGM que encontrou diversos desafios. Dentre eles, os principais desafios foram: a extração de características do *dataset* de teste realizada através de *chunks*, partes do arquivo de tamanho menor, causando a separação de observações de um mesmo objeto em *chunks* diferentes, os *ids* dos objetos no conjunto de teste apesar de agrupados não estavam ordenados como nos dados do cabeçalho, manipulação de *DataFrames* do pacote Pandas, dados incompletos, a escolha dos valores dos parâmetros de entrada para treinamento do modelo, o preenchimento dos dados nulos após a extração de características e a predição da classe 99 desconhecida.

A extração de características de treino e teste são semelhantes, no entanto, devido ao tamanho do conjunto de teste e à memória limitada da máquina que executou o modelo, a leitura desse conjunto foi particionada em frações de 6 milhões de linhas cada. Assim, foi contornado o problema do tamanho do conjunto de teste. Contudo, provocou a separação de observações de um mesmo objeto em partições diferentes. Essa separação cria resultados duplicados e divergentes com relação à característica real de um objeto processado.

Para solucionar o novo problema dos resultados duplicados, é verificado se o *id* do primeiro objeto do próximo *chunk* a ser processado é igual ao *id* do último objeto da última parte processada. Se sim, então esse último objeto processado é descartado de sua lista e é adicionado as observações de tal objeto no início do *chunk* atual até finalizar toda a leitura. É necessário manter as observações do último objeto de todo *chunk* processado para que seja possível adicionar essas observações no processamento subsequente de modo a recalcular as características daquele objeto com os seus dados por completo.

Posteriormente ao ler todo o conjunto de teste, foi verificado que a extração estava retornando valores nulos devido à diferença entre a ordem dos elementos. A ordenação dos objetos estava afetando o processo, pois, ao ler as primeiras 10 mil observações de 20 objetos, por exemplo, era lido os 20 primeiros objetos do conjunto de metadados. Os 10 iniciais objetos correspondiam, mas os 10 seguintes não, por isso o transtorno. Foi descoberta com certa demora, pois, inicialmente, o modelo fazia a leitura de cerca de 10 milhões de linhas utilizando 3 *chunks* de modo a poupar tempo e dentro desse intervalo os objetos dos dados e metadados se mantinham iguais.

Para a leitura dos arquivos no formato CSV (Seção 2.2.4), foi usado o pacote Pandas (NUMFOCUS, 2019b) do Python que retorna um objeto do tipo *DataFrame* 2.2.4. O Pandas oferece uma função de leitura de CSV tanto para ler o arquivo inteiro

quanto de modo iterativo, usando *chunks*, além de ter uma função chamada *pivot_table*. O *pivot_table* é usado para agrupar dados de acordo com alguma propriedade ou coluna (identificador de cada objeto), criar colunas com base em alguma característica dos objetos (os *passbands*) e executar outras funções ou cálculos sobre os valores agrupados (o *flux*, *flux_err*, *mjd*, *detected* medidos). Os cálculos executados foram: a média, mediana, variância e o desvio padrão sobre os fluxos e seus erros; a amplitude sobre o tempo das observações usando a diferença entre o máximo e o mínimo; e a soma das detecções. Todas as funções para os cálculos das características foram retiradas do pacote *numpy*, muito usado para computação científica, na linguagem Python, com suporte para grandes vetores ou matrizes multi-dimensionais (NUMFOCUS, 2019a).

Os dados incompletos de observações não registradas em determinados espaços de tempo resultou em características vazias para alguns objetos, o que pode não ser aceito por alguns algoritmos classificadores. Uma boa solução para isso seria utilizar de um *imputer* para cada objeto que substitui os valores nulos por algum valor dependendo da métrica escolhida. Ao menos utilizar o *imputer* para cada coluna de todo o conjunto, de modo a obter valores estimados não influenciados por outras características. Outra solução seria predizer toda a curva de luz, em um determinado intervalo de tempo, para cada objeto em cada um dos 6 *passbands*, através de regressões lineares, por exemplo.

Em vista das prováveis soluções, a saída implementada foi substituir as características ausentes de cada objeto, após o processamento perdendo alguns dados, pela mediana dessa característica no conjunto. Para tanto, a função *SimpleImputer* do pacote *scikit-learn* sobrepôs os valores ausentes utilizando a métrica mediana. Apesar da perda de alguns dados, essa abordagem se mostrou melhor, através da pontuação obtida, do que uma abordagem mais simples que não extrai as características de cada *passband* de cada objeto.

Após todo o processamento dos dados, ainda é necessário definir os parâmetros que o algoritmo classificador irá receber antes de ser treinado e executar suas predições. Começando com os parâmetros presentes na solução do Grellier (GRELLIER, 2019b), foi possível construir um modelo inicial, avaliar os resultados obtidos através do Kaggle e, testar e melhorar os parâmetros até que encontrada a combinação com o melhor resultado:

```
params = {
    'boosting_type': 'gbdt',
    'objective': 'multiclass',
    'num_class': 14,
    'metric': 'multi_logloss',
    'learning_rate': 0.03,
    'num_iterations': 230,
```

```
    'max_depth': 6
}
```

Os quatro primeiros parâmetros são de configuração (MICROSOFT, 2019b). O primeiro define o tipo de *boosting* como GBDT, o segundo define o objetivo para classificar múltiplas classes usando o algoritmo *softmax*. Assim, o resultado da classificação de cada objeto é a probabilidade dele pertencer a cada uma das classes. Se somado essas probabilidades o resultado será igual a 1,0. O terceiro parâmetro informa ao algoritmo a quantidade de classes existentes nos conjuntos enquanto o quarto define a métrica para calcular a potuação ou avaliação do modelo.

Os últimos três parâmetros são para regular a acurácia e o *overfitting*. O número de iterações de *boosting* relativamente alto para aumentar a acurácia e para controlar o *overfitting*, uma taxa de aprendizado baixa e um limite de profundidade de 6 (MICROSOFT, 2019c).

Por fim, para predizer a classe 99, a abordagem da Belinda Trotta resultou em predições com pontuações mais elevadas se comparado com a abordagem proposta por Grellier. Em síntese, essa abordagem consiste em determinar a probabilidade de um objeto pertencer à classe 99 como sendo o oposto da maior probabilidade de o objeto pertencer a alguma classe. Portanto:

```
predict_99 = 1 - max(predictions)
```

Partindo dessa abordagem, o resultado médio das probabilidades de todos os objetos para ser essa classe foi de 40%, uma probabilidade muito alta que precisou ser normalizada. Portanto, cada estimativa dessa classe de cada objeto passou pela função de normalização:

```
0.14 * predict_99 / numpy.mean(predict_99)
```

Então a média foi modificada para 0,14 como mostrado em (GRELLIER, 2019b). Essa estratégia resultou em um aumento de 0,035 na avaliação dos resultados. Normalizar os valores para que a soma fosse igual a 1 não alterou os resultados e, alterar a média para outros valores ou não alterá-la tem um resultado pior do que obtido modificando a média para 0,14.

3.1 Estrutura do Código

O código desenvolvido é separado em 3 módulos principais: a análise de dados, a extração de características e os modelos de classificação. Esses módulos ainda usufruem

de 3 outros arquivos complementares: instalação dos pacotes necessários, configuração das variáveis de ambiente e de otimização, e leitura dos dados. Cada um tem uma função específica, descrita nesta seção, para que juntos executem todas as ações mostradas no início deste capítulo. Todos os arquivos estão disponíveis no GitHub (*https://github.com/marcio55afr/PLAsTiCC_Solution*) caso seja interessante. Para executar o projeto, é preciso baixar os arquivos da competição através do Kaggle ou do *site* do PLAsTiCC (LSSTC, 2019d).

3.1.1 Arquivo DataAnalysis.py

Esse módulo é responsável por analisar todos os dados e metadados de treino com intenção de descobrir comportamentos, obter informações sobre as distribuições das fontes astronômicas, encontrar diferenças entre classes e observar como algumas características estão dispostas. Toda essa análise tem o propósito de compreender melhor o problema, abordado pelo PLAsTiCC, de modo a encontrar uma solução adequada. Ela direciona a extração de características dos objetos e poderia auxiliar, se feito, um processo de *augmentation* (Seção 2.2.3) ou de interpolação das curvas de luz.

Nesse módulo foram criadas duas grandes funções e uma função inicial. Uma das funções maiores utiliza os dados processados com características geradas separadamente para cada *passband* enquanto a outra utiliza as características mais simples, que não considera os *passbands* separadamente. Ambas produzem gráficos de dispersão, histogramas e gráficos em barras através da classe *DataFrame* do Pandas e do pacote Matplotlib (HUNTER, 2007). A terceira função foi criada para começar as análises, chamada de *DataAnalysisInitial()*. A Figura 6 é um exemplo de um histograma produzido pela função inicial. Foi produzido através do código Python abaixo e apresenta a distribuição da média, mediana e do desvio padrão do conjunto de treino:

```
pyplot.histogram( TrainingData['flux_mean'], bins=100, graphic_1)
pyplot.histogram( TrainingData['flux_median'], bins=100, graphic_2)
pyplot.histogram( TrainingData['flux_std'], bins=100, graphic_3)
pyplot.show()
pyplot.close()
```

3.1.2 Arquivo ExtractFeatures.py

O módulo de extração de características, ou *features*, é dividido em três grupos distintos de funções. Um desses grupos de funções criam uma tabela de *features* desconsiderando os filtros das observações, essa tabela é citada nesse trabalho como as características simples, e um segundo grupo cria as características para cada *passband* separadamente.

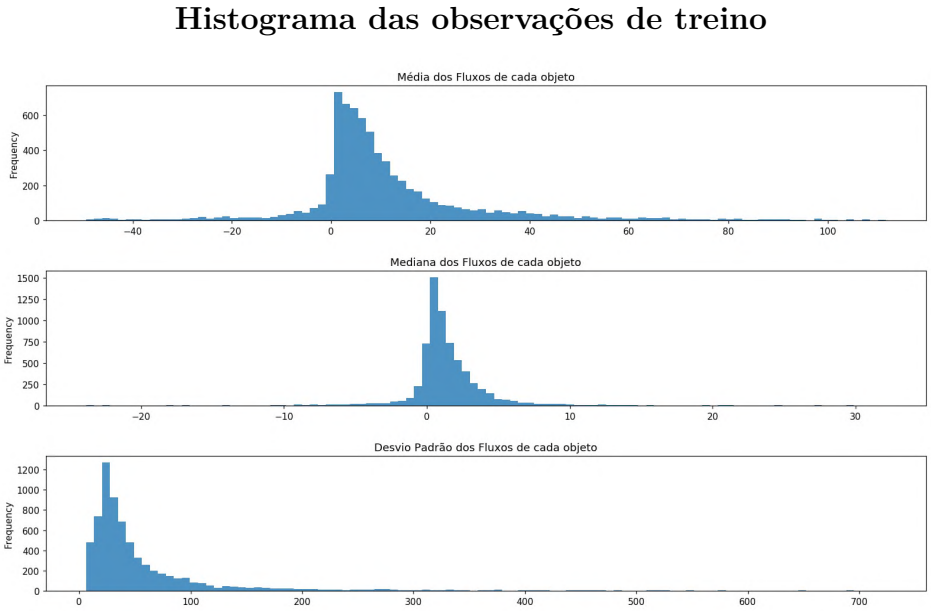

Figura 6 – Histograma com análise dos fluxos dos objetos em um quantil de 0,05 a 0,95.

O último grupo calcula algumas características específicas, como, por exemplo, a média dos fluxos, usadas na análise de dados.

Os dois primeiros grupos atuam de maneira similar, cada um possui uma função para extrair os dados de treino, uma para extrair os dados de teste e uma terceira função que recebe um conjunto de dados, contendo as observações seja de treino ou de teste, e um cabeçalho de informações correspondentes. Essa função agrupa esse conjunto de dados com base no identificador do objeto presente na observação; faz os cálculos estatísticos de modo a criar as características considerando ou não cada *passband* separado; renomeia todas colunas da tabela produzida; verifica a existência de um objeto sem características; faz a junção da tabela produzida com os metadados; e, por fim, retorna esse grande conjunto em forma de um *DataFrame*.

A diferença entre os dois grupos primordiais é a extração de características simples ou para cada *passband*. Para mais, existe outra diferença entre a função de extração de treino e de teste em cada um dos grupos, que é a leitura dos arquivos de dados. Para o treinamento, é possível ler os dados e os metadados todos de uma vez e calcular todas as características diretamente. Enquanto que para o teste, apenas a tabela de cabeçalho e mais uma parte dos dados, chamada de *chunk*, é realizável.

Essa leitura em *chunks* criou o problema de particionamentos das observações de um mesmo objeto citada no começo dessa Seção. Além da diferença de ordenação entre os dados e os metadados, o que tornou necessário encontrar os objetos dos metadados correspondentes com cada objeto dos *chunks* para serem enviados corretamente à função de extração.

Os resultados das extrações são escritos em arquivo de extensão h5 (Seção 2.2.4) com a intenção de poupar tempo, persistindo as características em um arquivo. Além de otimizar a leitura em tempo e memória, pois, um arquivo h5 ocupa um espaço de memória menor se comparado a outros formatos (Seção 2.2.4).

O terceiro e último grupo é composto por funções complementares utilizadas pelos outros módulos. São apenas três funções, duas calculam a média, mediana e o desvio padrão do fluxo de cada objeto, e a terceira faz a contagem de objetos por classes. Dentre as duas primeiras, uma considera cada *passband* separadamente e a outra apenas o *id* dos objetos. Todas elas atuam somente sobre o conjunto de dados e metadados de treino.

3.1.3 Arquivo LGBModel.py

O terceiro módulo é onde estão construídos os modelos classificadores. Nele existem 4 funções cada uma contendo um classificador que variam entre árvores de decisões e modelos LGBs. Começando pela mais simples, a função *DTree_using_TrainingDataSet()* lê o arquivo de características do conjunto de treino e o divide em treino e teste, retirando o *target* desse conjunto de teste criado. Em seguida, essa função treina uma árvore de decisão com o *dataset* de treino menor e executa a classificação sobre o *dataset* de teste criado por ela. Ela também avalia as predições com base nos rótulos removidos anteriormente e repete esse processo 1000 vezes com diferentes particionamentos, através do método *random subsampling* (Seção 2.2.3). As partições são igualmente proporcionais e todas derivadas do conjunto de características dos dados de treino original.

Ainda é possível obter um relatório através do pacote *scikit-learn* e o pior e a melhor pontuação das predições através do próprio código escrito. As pontuações de cada predição é calculada a partir do método *score()* pertencente à classe *DecisionTreeClassifier* também do pacote *scikit-learn*. Esse modelo atua somente com as características simples, pois, os possíveis valores nulos presentes no outro conjunto não são tratados.

A segunda função foi construída para tratar valores nulos na tabela de teste e possui uma árvore de decisão como algoritmo classificador. Essa função executa todo o processo de aprendizado e classificação da seguinte forma: faz a leitura do arquivo de características do conjunto de treino; em seguida, realiza o aprendizado com essas características; lê o arquivo de teste extraído; utiliza o *SimpleImputer* para preencher os valores ausentes no conjunto de teste; faz a predição da classe de todos os objetos de teste; prediz a classe 99 com base na abordagem da Belinda Trotta; ordena as predições de acordo com o *object_id*; e escreve todo o resultado em um arquivo CSV para submissão e avaliação dos resultados.

Esse segundo modelo que utiliza uma árvore de decisão, foi um modelo para testar o processo de ler de classificar os dados do teste e submeter os resultados no Kaggle, de

modo que não obteve resultados significantes. Porém, criou uma base para a construção dos próximos modelos. Os resultados foram escritos em formato CSV, pois, é como a competição exige para aceitar a submissão dos resultados e calcular uma pontuação.

As outras duas funções atuam como a citada anteriormente. A diferença está no algoritmo utilizado e na criação de seus parâmetros. Nessas duas funções é empregado o algoritmo LGBM e fazem todo o processo descrito de leituras, treinamento, predições, predição da classe 99, ordenação e escrita dos resultados em um CSV para submissão.

A diferença entre essas duas funções é as características utilizadas para treinamento e predição. Enquanto a função *LightBGM_Classifier()* trabalha sobre as *features* simples, a *LightBGM_Classifier_ByPassbands()* age sobre as *features* extraídas para cada *passband* além de fazer a substituição dos valores ausentes nesse conjunto. Como esperado, este último modelo se saiu melhor que todos os outros, apesar da perda de informações que não foi eliminada do processo de extração de características.

3.1.4 Arquivos Auxiliares

Os três arquivos restantes, *InstallPackages.bash*, *Config.py* e *ReadDataSet.py* existem para controle e modularização das funções e variáveis usadas nos outros módulos, além de comandos para executar o projeto em outras máquinas. O primeiro arquivo contém um *script* que instala os pacotes do Python necessários para a execução correta dos módulos em um ambiente Linux ou Windows, dependendo do ambiente é necessário adaptar alguns códigos. O Python 3 e o *pip* (Seção 2.2.4) são pré-requisitos para a instalação do restante dos pacotes. O arquivo de configuração possui variáveis de ambientes, variáveis para leitura dos arquivos de modo eficiente, como a definição da tipagem de cada coluna das tabelas de dados e metadados, e o *import* de pacotes usados nos módulos, como, por exemplo, o *numpy*, *pandas* e outros. Por último, o terceiro arquivo auxiliar contém funções para a leitura dos arquivos de dados, de metadados e de características previamente extraídas.

Através dessa estrutura é possível identificar os principais conceitos aplicados neste trabalho de conclusão de curso: análise de dados, preprocessamento de dados e aprendizado de máquina. Esse formato ficou mais próximo do trabalho feito pela Belinda Trotta, explicado na Fundamentação Teórica (Capítulo 2) e que atingiu uma das melhores posições da competição. Nos campos citados, muitos métodos ainda poderiam ser aplicados de modo a melhorar significativamente o modelo. Muitos desses métodos foram utilizados nas melhores soluções do Kaggle como, por exemplo, aplicar *augmentation*, predizer as curvas de luz por completa, extrair características específicas de cada classe, usar algoritmos para calcular automaticamente os parâmetros e utilizar abordagens diferentes para predizer a classe desconhecida.

3.2 Considerações finais

A estrutura adotada para o desenvolvimento desse trabalho seguiu a ordem de como é apresentados os conceitos de mineração de dados pelo livro *Data Mining* (HAN MICHELINE KAMBER, 2012). Ele apresenta o que é a mineração de dados, conceitos de análise de dados, o pré-processamento e então os algoritmos classificadores. O que deixou o todo o trabalho bem organizado e uma estrutura para uma evolução e aplicação de novos métodos.

4 Testes e Experimentos

Neste capítulo são descritos os principais experimentos realizados no modelo que forneceram informações sobre o problema em questão, sobre os conjuntos de dados e abordagens que aumentaram diretamente a pontuação das predições. Nele está presente toda a análise de dados realizada e a evolução de como foi feita certas escolhas e mudanças no projeto, como, por exemplo, o uso das características extraídas para cada *passband* separadamente e como os parâmetros do classificador influenciam seus resultados. Adicionalmente, são relatadas análises, informações e fatos interessantes que foram concebidos e expostas em discussões e *kernels* (Seção 2.3) da competição.

4.1 Análises Kaggle

Para começar a analisar os dados e o problema em sua totalidade, o *kernel* disponível no Kaggle *Starter Kit* (TEAM, 2019) foi indispensável. Ele faz uma introdução de todo o problema, do objetivo da competição e de suas dificuldades. Aborda aspectos interessantes, como o porquê da existência de fluxos negativos nos *datasets*, do envermelhamento (*redshift*) dos objetos astrofísicos, do erro das medições e diferenças entre o *redshift* da espectroscopia e da fotometria. Esse *kernel* contém explicações sobre as curvas de luz os objetos, como o envermelhamento está relacionado com a expansão do universo e como as supernovas do tipo *Ia* mostram a expansão crescente do universo. Além disso, esse *kernel* também mostra como a poeira cósmica da Via Láctea influencia as observações simuladas, representada pelo atributo *MWEBV* do *dataset*.

Após essa grande quantidade de informações a pesquisa pôde ser iniciada com um direcionamento claro, além de contar com a ajuda de outras análises mais específicas e voltadas para os conjunto de dados. Duas análises que ajudaram bastante porém não tão abrangentes quanto o *kernel Starter Kit* foram a *The Astronomical (complete) EDA - PLAsTiCC dataset* (SILVA, 2019) e a análise *Dataset overview — Exploration and comments* (MELLO, 2019). A primeira delas verifica o brilho por tempo de cada classe e as diferenças nas distribuições entre as áreas de pesquisa DDF e WDF, enquanto a segunda, localiza os campos DDFs nas áreas em destaque, como mostra a Figura 7, evidencia a distribuição de classes nesses campos de DDF e evidencia a diferença entre o *host_specz* e o *host_photoz*.

Um amplo leque de análises estão disponíveis através dos *kernels* e das discussões, cada tendo a sua contribuição, e demais três se destacam e merecem ser citadas. Especificando cada vez mais em um assunto, a análise *Train set vs Test set Differences EDA* (O'BRIEN, 2019) aponta as diferenças entre a distribuição de alguns atributos entre os

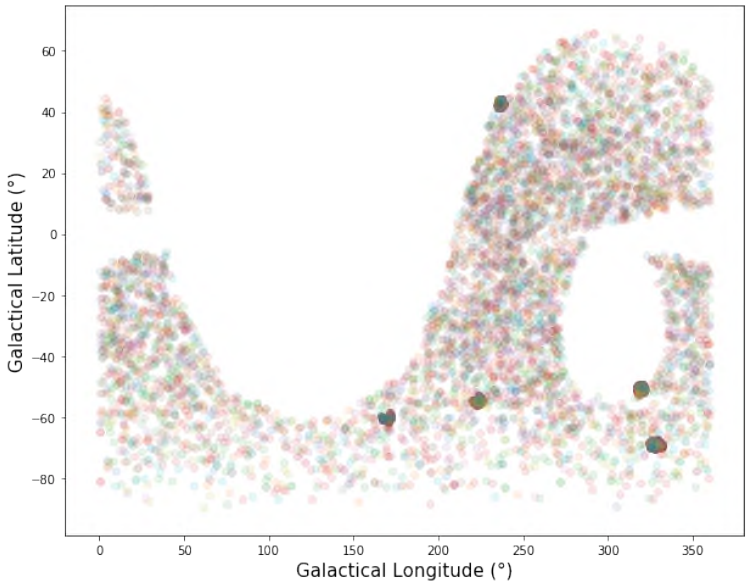

Figura 7 – Pontos com concentração de objetos representando as áreas DDFs.

Fonte: https://www.kaggle.com/hrmello/dataset-overview-exploration-and-comments (Julho 2019)

objetos de treino e de teste, exemplo na Figura 8. O texto *Where is the gap?* (PUGET, 2019b) estabelece a relação entre as lacunas de observações em determinado período do tempo e suas coordenadas celestes, Figura 9. *Galactic vs Extragalactic Objects* investiga sobre a diferença entre as classes galácticas e extragalácticas (Figura 10) e levanta questões sobre a classe desconhecida (Seção 2.4.1).

Figura 8 – Diferença das distribuições do *redshift* mensurado através da espectroscopia.

Fonte: https://www.kaggle.com/obrienmitch94/train-set-vs-test-set-differences-eda (Julho 2019)

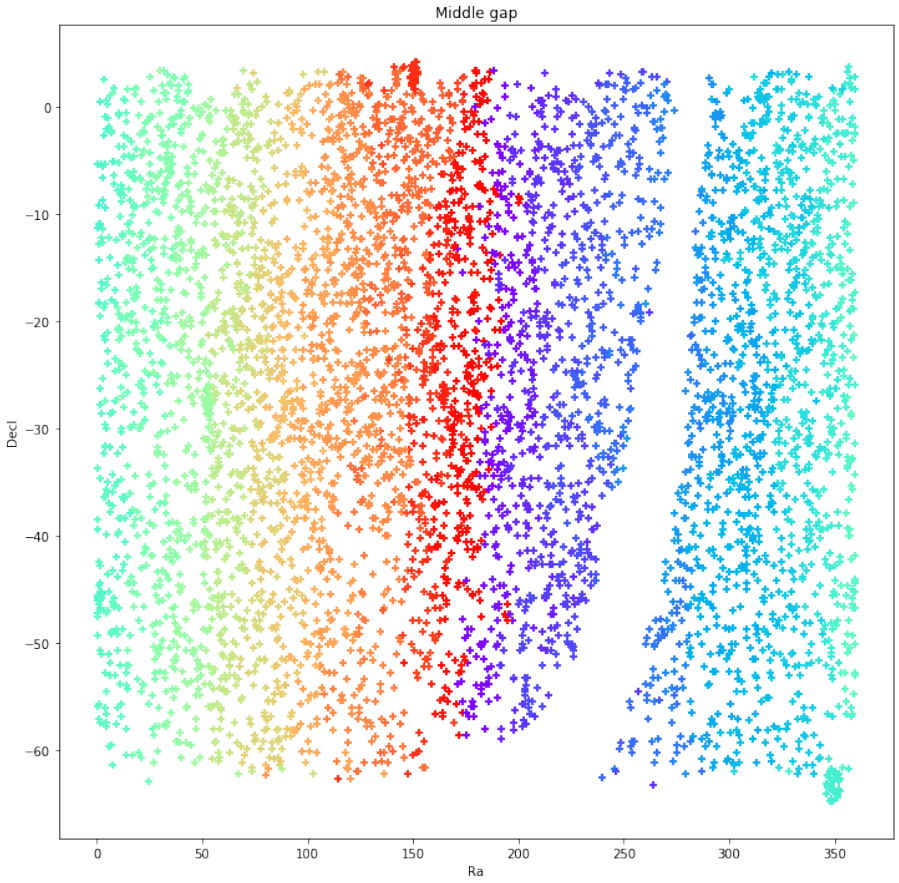

Figura 9 – Posição dos objetos mudando de cor de acordo com a mudança da data média da sua lacuna de observação.

Fonte: https://www.kaggle.com/c/PLAsTiCC-2018/discussion/71292 (Julho 2019)

4.2 Análise dos dados

Como exposto na Seção 3.1, a análise desenvolvida durante esta pesquisa foi implementada no aquivo *DataAnalysis* composto de três funções principais. Cada uma delas abordou algum questionamento ou curiosidade com o objetivo de entender os dados e melhorar a extração de características elaborada no módulo de extração. A primeira função, *DataAnalysisInitial()*, analisa a média dos fluxos dos objetos de treino bem como suas classes e o campo em que ele foi observado (DDF ou WDF). A segunda função é a *DataAnalysis()*, ela examina as posições dos objetos no espaço de acordo com sua classe, a relação entre as medições do envermelhamento dos objetos e a correlação entre as características simples extraídas dos dados com os metadados. Por último a *DataAnalysisSeparetePassbands()*, que busca encontrar diferenças entre as classes dos objetos a partir da média dos fluxos em diferentes *passbands* utilizando diferentes abordagens.

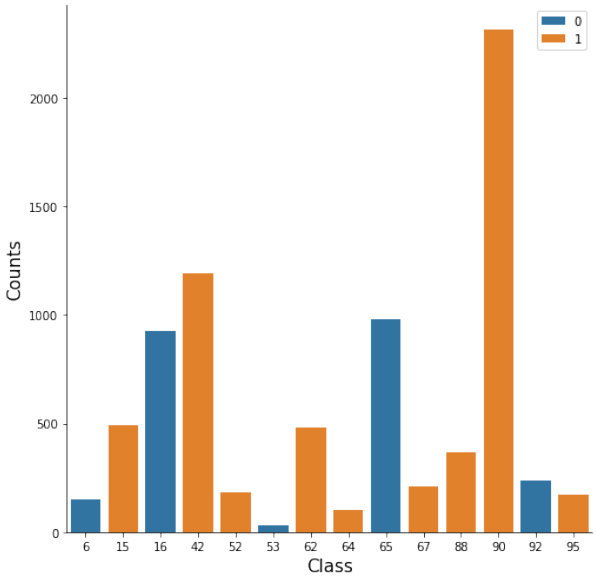

Figura 10 – Classes dividas entre galácticas e extragalácticas.

Fonte: https://www.kaggle.com/kyleboone/naive-benchmark-galactic-vs-extragalactic/notebook (Julho 2019)

4.2.1 A função *DataAnalysisInitia()I*

Com o intuito de iniciar a análise e observar as distribuições de forma simples foi utilizado as *features* mais simples para ser possível ver a distribuição dos objetos. As Figuras 11 e 12 mostram a distribuição da média, mediana e desvio padrão do fluxo dos objetos de treino excluindo os *outliers* para uma visualização melhor. Em ambos é mostrado uma concentração nos valores próximos a zero e positivos além de uma diferença dos valores no gráfico de dispersão a partir do objeto 2 000. O *id* é um número único para cada objeto de modo a normalizar a construção do gráfico.

A Figura 13 evidencia a diferença vista nos gráficos anteriores tem relação direta com o atributo DDF de cada objeto. Dependendo de qual área foi feita a observação, existe uma tendência para os valores das médias dos fluxos. Se o objeto foi avistado na área *Wide-Fast-Deep* então a média de seu fluxo tende a ser um pouco maior em relação aos objetos da área de DDF.

Observando como estão dispostas a média dos fluxos dos objetos em cada classe separadamente, percebe-se pequenas diferenças. A Figura 14 mostra que as classes 6, 16, 88 e 92 possui as médias dos fluxos mais esparsas, que as médias das classes 15, 42 e 90 um pouco esparsas e de valores positivos, e que as classes restantes contêm valores próximos a 0 na maioria de seus objetos. Ainda nesse mesmo gráfico, é perceptível uma diferença da quantidade de objetos, de mesma classe, entre a áreas DDF e WDF. As classes 6, 15 e 64 são mais densas na área de pesquisa WDF como constatado na figura 15.

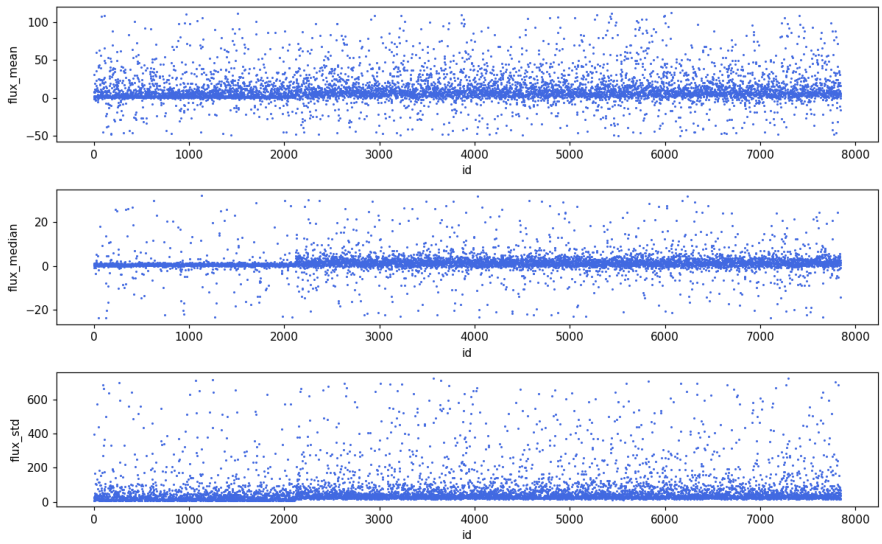

Figura 11 – No eixo horizontal são identificados os objetos e no eixo vertical as médias, medianas e os desvios padrão dos fluxos de cada objeto de treino.

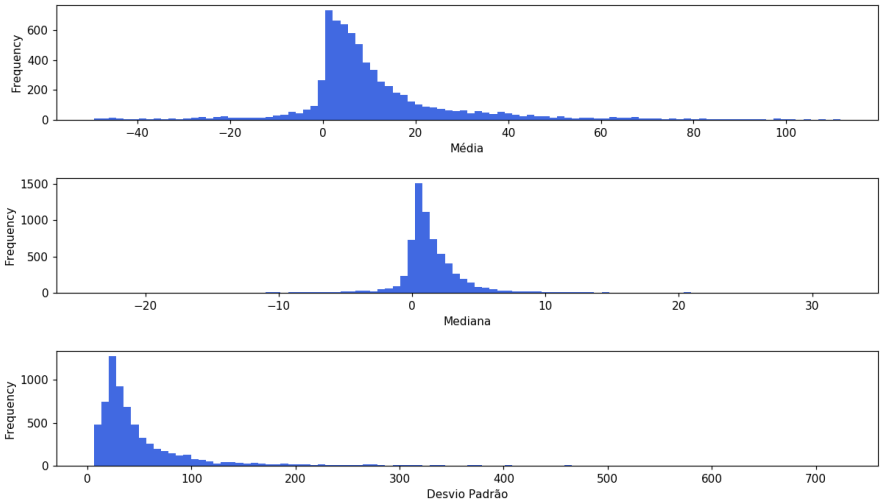

Figura 12 – Histograma da média, mediana e desvio padrão dos fluxos dos objetos de treino.

Na relação de proporção é exposto diferença em mais algumas classes. No gráfico dessa relação, as classes 6, 15, 16, 53 e 64 são mais presentes na *Wide-Fast-Deep*, a 52 e a 90 mais presentes na *Deep Driling Field* enquanto o restante das classes estão igualmente distribuídas.

Finalizado então a primeira análise, é obtido uma visão geral sobre uma das características extraídas e relação entre as áreas de pesquisa das observações e as classes dos objetos. Foram criadas imagens para a mediana e o desvio padrão, contudo, os resultados

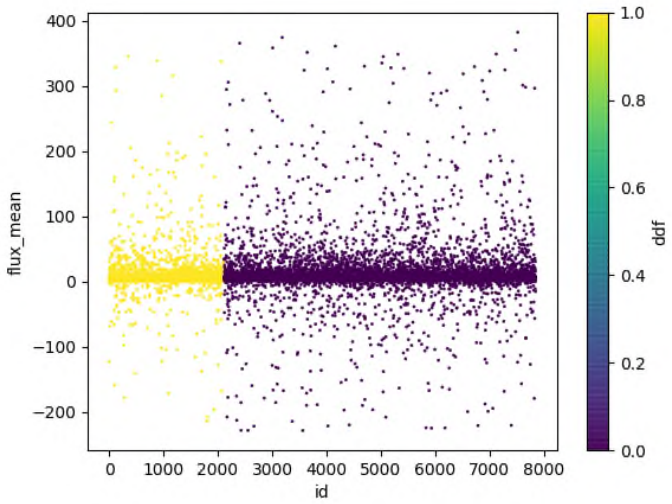

Figura 13 – Média dos fluxos do objetos de treino com coloração baseada na área de observação.

foram equivalentes aos atingidos, por isso não foram incluídas nesse documento.

4.2.2 A função *DataAnalysis()*

Alterando o foco para os metadados, a função *DataAnalysis()* constrói gráficos que exibem a disposição das fontes no céu em um plano bidimensional, o crescimento exponencial do avermelhamento com o aumento da distância das fontes astronômicas e a correlação entre as características simples extraídas.

Nas Figuras 16 e 17 estão dispostos todos os objetos do conjunto de treino nas coordenadas celestes e galácticas e separados por classes. Foram feitos com o propósito de analisar alguns pontos citados nos *kernels* sobre as diferenças entre as distribuições das classes (O'BRIEN, 2019). Algumas observações podem ser constatadas, como a ausente de objetos em algumas classes entre as latitudes 100 e 200 e a quantidade desbalanceada de objetos em cada classe. Duas áreas das coordenadas galácticas não possuem observações devido às limitações físicas para realizar as medições.

A Figura 18 relaciona o avermelhamentos medido tanto através da fotometria e da espectroscopia. Relembrando que as medições através da fotometria são imprecisas e apresenta divergência com um agrupamento existente com *redshift* próximo a 0 porém alta distância modular (MELLO, 2019). O próximo gráfico mostra a correlação entre as características simples extraídas do treino com apenas a média, mediana e o desvio padrão (Figura 19).

Apesar de interessante e esclarecer algumas dúvidas sobre os objetos das observações, o resultado produzido por essa função não foi conclusivo. Ela também foi utilizada

para analisar a relação do avermelhamento com outros atributos das fontes com o intuito de fazer uma predição da propriedade *hostgal_specz*. Abordagem esta não utilizada nesse trabalho.

4.2.3 A função *DataAnalysisSeparetePassbands()*

Após uma tentativa ineficiente de gerar características para os diferentes *passbands* do conjunto de teste, foi analisado no conjunto de treino se existe uma separação das classes em decorrência desses filtros. Também foi utilizado a localização do objeto (interno ou externo à Via Láctea), a área da observação (DDF ou WDF) e um gráfico de densidade para tentar encontrar algo relevante para a classificação.

A primeira figura dessa função, a Figura 20, foi gerada de modo a concentrar o foco em apenas um filtro. Ela exibe muitos objetos de todas as classes, o que resulta em um gráfico de dispersão confuso. Esse gráfico foi gerado a partir da média do fluxo dos objetos e o seu eixo Y foi limitado entre -1000 e 1000 para evitar os *outliers*.

Prosseguindo, como na figura de exemplo, foram gerados os gráficos mas agora para todos os *passbands* e em diferentes partes do plano cartesiano com diferentes limites do eixo Y. Em busca por diferenças significativas entre as classes de modo a justificar o uso dessas características para cada *passaband*.

O primeiro gráfico foi gerado a fim de obter uma visão mais distantes da dispersão dos dados, representado pela Figura 21. Nela é enfatizada uma diferença entre os *outliers* de cada filtro e o agrupamento dos dados próximos a 0 em todos os filtros. Existe uma distinção sutil dos *passbands g, z* e *y*, nos quais, o primeiro deles, possui a classe de cor rosa (12) um pouco mais esparsa que o restante, enquanto, nas outras duas, acontece o mesmo com a classe de cor verde (6). Vale ressaltar que o objetivo dessas análises é encontrar uma distinção das classes a partir dos *passband*, sendo irrelevante, neste caso, descobrir qual a classe que está separada das demais.

Aproximando-se da média geral é possível identificar um contraste entre as áreas de DDF, eixo x de 0 a 2000 aproximadamente, e WDF (Figura 22). Poucos pontos azuis, distantes da média geral, são encontrados nos *passbands u* e *y* como nos demais filtros.

Buscando distinções em outras áreas do gráfico, foram criadas as Figuras 23 e 24 que mostram a parte inferior e superior do gráfico presente na Figura 22. Nessas imagens é possível identificar uma diferença maior no filtro *u* que, na parte negativa quase não possui médias e na parte positiva contém médias de objetos distintos dos demais.

Aproximando ainda mais da média geral dos fluxos, a dispersão torna-se mais diversificada e difícil de distinguir características. Mesmo separando as áreas de DDF e WDF, Figuras 25 e 26 respectivamente, as diferenças de proporção das classes em diferentes filtros parecem não existir.

Ainda tentando encontrar diferenças nas proximidades do centro da média geral, de valor 0, foram separadas a classes galácticas das extragalácticas e observado o campo da dispersão (Figuras 27 e 28). Essa tentativa também não gerou bons resultados.

Por fim, foram feitos gráficos de densidade de alguns *passbands*, um histograma com a quantidade de cada classe dividindo os objetos em 100 grupos, chamados de *bins*, e gráficos em barras representando a mesma informação que o histograma.

Continuando a analisar a característica de média extraída do conjunto de observações de treino, a Figura 29 apresenta um gráfico de densidade com poucas variações entre as classes e um histograma um tanto quanto confuso. Essa imagem foi calculada com as observações do filtro i e contém todas as classes. Para simplificar e esclarecer essas informações, as classes galácticas e extragalácticas foram separadas, observadas na Figura 30, e o histograma foi substituído por um gráfico em barras.

O último gráfico (Figura 31) exibe a função densidade e o gráfico em barras para classes extragalácticas observadas no *passband* u. Neste existe uma divergência entre as classes, sendo melhor apresentada pelo gráfico em barras, nos valores negativos.

A etapa de análise dos dados foi finalizada e foi bem aproveitada. Algumas das análises obtidas através do Kaggle puderam ser confirmadas, uma visão geral do conjunto de dados e do problema que o cerca foram expostas e a diferença das classes apresentadas em alguns *passbands* diferentes foi demonstrada através dos resultados da função *DataAnalysisSeparetePassbands()*. Essa diferença indica que as características extraídas em *passbands* diferentes podem obter melhor pontuações, o que foi observado durante os testes e é descrito na Seção 4.3.

4.3 Testes

Nessa seção é apresentado alguns dos testes executados. A maioria são testes que buscam uma melhoria na pontuação de algum dos modelos classificadores criados através da mudança das características usadas no treinamentos ou dos parâmetros de configuração.

Para testar as características foi aplicado um método chamado *random sampling* 2.2.3 ao modelo de árvore de decisão. O conjunto de treino e de teste utilizados neste teste são subdivisões das características extraídas do conjunto de treino.

Aplicado o método de *random sampling* sobre as características extraídas do conjunto de treino, a média do fluxo e do erro do fluxo, e com uma relação de particionamento de 99% para teste, o resultado obtido foi:

```
              precision    recall  f1-score   support
    accuracy                           0,42      7770
   macro avg       0,30      0,28      0,27      7770
weighted avg       0,45      0,42      0,43      7770

pontuação média = 0,4017252252252247
```

Acrescentando a amplitude do MJD e a contagem e média da propriedade *detected* o resultado foi:

```
              precision    recall  f1-score   support
    accuracy                           0,48      7770
   macro avg       0,41      0,38      0,36      7770
weighted avg       0,51      0,48      0,47      7770

pontuação média = 0,44620437580437555
```

A amplitude dos fluxos e do erro dos fluxo foram acrescentadas porém não resultou em melhoras significativas.

```
              precision    recall  f1-score   support
    accuracy                           0,47      7770
   macro avg       0,39      0,36      0,33      7770
weighted avg       0,50      0,47      0,46      7770

pontuação média = 0,44969703989704024
```

Ao acrescentar os valores da mediana, desvio padrão e variância a pontuação aumentou novamente:

```
              precision    recall  f1-score   support
    accuracy                           0,48      7770
   macro avg       0,40      0,40      0,38      7770
weighted avg       0,52      0,48      0,48      7770

pontuação média = 0.4681908622908617
```

Relação de particionamento (% para teste)	Tipo de extração	Colunas	Características	Pontuação média (Random Sambling)
99%	Simples	Fluxo/Erro	+ média	0,4017252252252247
		MJD	+ amplitude	0,44620437580437555
		Detectado	+ contagem + média	
		Fluxo/Erro	+ amplitude	0,44969703989704024
		Fluxo/Erro	+ mediana, + desvio padrão + variância	0.4681908622908617
	Para cada *passband*	As mesmas colunas	características finais	0,42199279279279217
30%	Simples	As mesmas colunas	características finais	0,5986157112526596
	Para cada *passband*	As mesmas colunas	características finais	0,6380203821656079

Tabela 3 – Tabela com a representação da evolução da pontuação de acordo com o acréscimo de características em determinada propriedade das observações adicionado a diferença entre as relações de proporção de partição do conjunto de treino e entre as características simples e por *passband*

Utilizando tais características encontradas através dos testes para cada *passband* separadamente a pontuação foi de:

```
                precision    recall  f1-score   support

    accuracy                           0,50      7770
   macro avg       0,35      0,34      0,33      7770
weighted avg       0,51      0,50      0,49      7770

pontuação média = 0,42199279279279217
```

Estranhamente a pontuação foi bem menor, porém, se comparado com um subconjunto de treino de 70% e o de teste de 30%, a pontuação com as características por *passband* supera a pontuação com as caracteísticas simples. O resultando em uma pontuação média de 0,6380203821656079 para a primeira e de 0,5986157112526596 para a segunda.

De modo resumido, a evolução das pontuações de acordo com o acréscimo de características extraídas é representada pela Tabela 4.3.

Sendo assim, as melhores características encontradas para cada propriedade das observações após os testes de *random sampling* foram:

```
'flux' : mínimo, máximo, média, mediana, variância, desvio padrão
'flux_err' : mínimo, máximo, média, mediana, variância, desvio padrão
'mjd' : amplitude
'detected' : média, soma
```

Todas essas características foram calculadas separadamente para cada *passband* e empregadas no modelo de classificação que utiliza o algoritmo LGBM. Esse algoritmo precisa de parâmetros bem definidos para obter bons resultados, caso contrário, o treinamento pode ficar tendencioso em relação ao conjunto de treino ou ter uma baixa acurácia. Dito isso, os primeiros parâmetros foram adquiridos de um dos *kernels* do PLAsTiCC escrito pelo Olivier Grellier (GRELLIER, 2019b), foram estes:

```
parametros_lgb = {
    'boosting_type': 'gbdt',
    'objective': 'multiclass',
    'num_class': 14,
    'metric': 'multi_logloss',
    'learning_rate': 0.03,
    'subsample': .9,
    'colsample_bytree': .7,
    'reg_alpha': .01,
    'reg_lambda': .01,
    'min_split_gain': 0.01,
    'min_child_weight': 10,
    'n_estimators': 1000,
    'silent': -1,
    'verbose': -1,
    'max_depth': 3
}
```

Apesar de terem bons resultados para o modelo de Grellier e outros competidores terem utilizado desses parâmetros, para o modelo construído através das características descritas nessa Seção, eles não resultaram em uma pontuação baixa na competição. A pontuação foi de 2,80, ao passo que a pontuação ao predizer todas as classes com a mesma porcentagem é de 2,70.

Após algumas tentativas e submissões de resultados, chegou-se a melhor pontuação utilizando os parâmetros:

```
params = {
        'boosting_type': 'gbdt',
        'objective': 'multiclass',
        'num_class': 14,
        'metric': 'multi_logloss',
        'learning_rate': 0.01,
        'num_iterations': 150,
        'max_depth': 3
    }
```

Com esses parâmetros e as melhores características encontradas, o modelo de classificação com o algoritmo LGBM alcançou a pontuação de 1,651. Um resultado satisfatório que chegaria à colocação de 704 lugar, caso submetido dentro do tempo da competição.

4.4 Considerações finais

Diante das análises realizadas e dos testes aplicados, foi possível construir e melhorar a extração de características dos conjuntos de dados e progredir as pontuações das predições do modelo construído até chegar à pontuação final. Em paralelo, a hipótese inicial de que as características extraídas em cada filtro separadamente geraria um aprendizado do modelo melhor foi constatada.

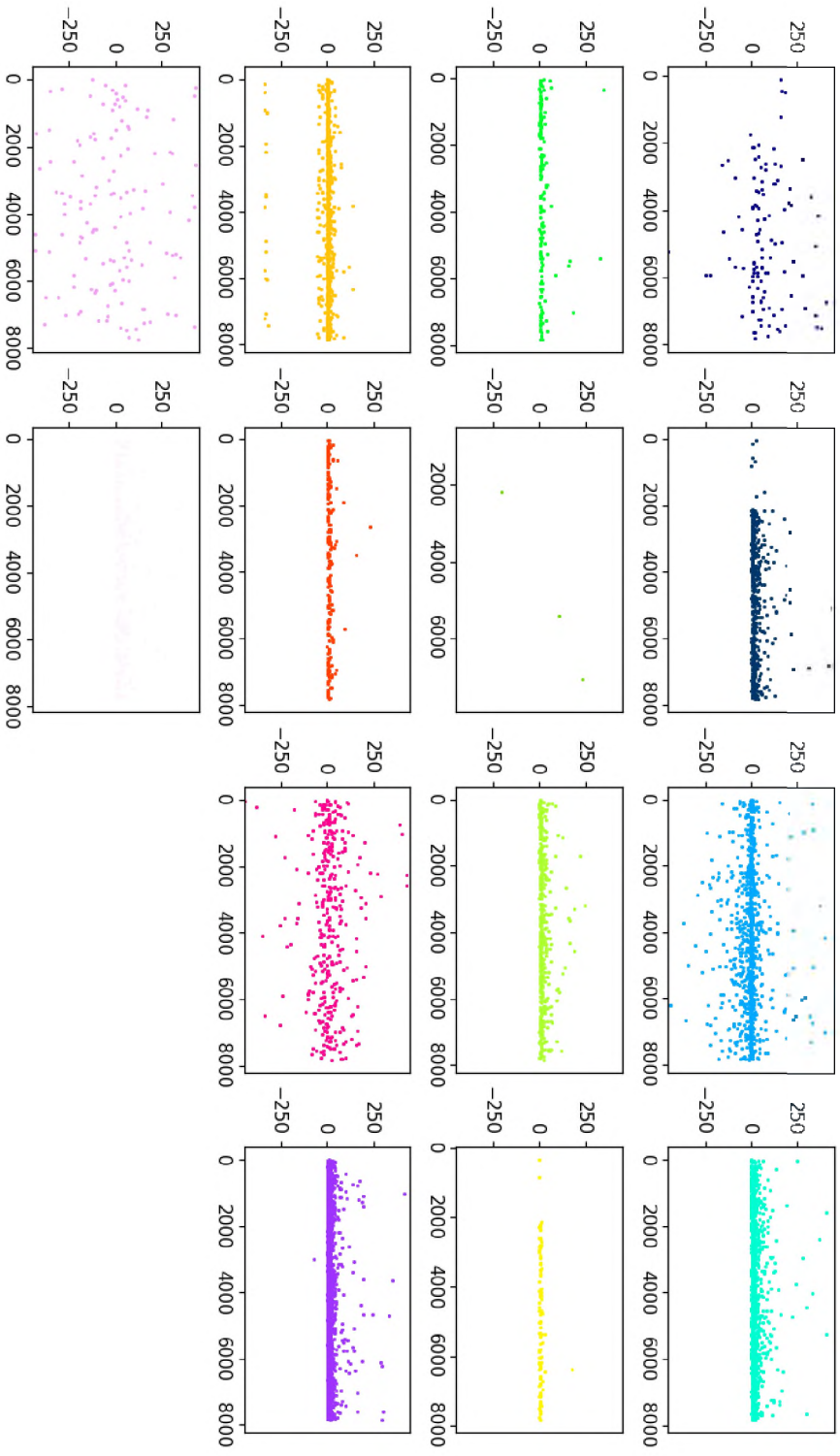

Figura 14 – Média dos fluxos objetos para cada classe separadamente.

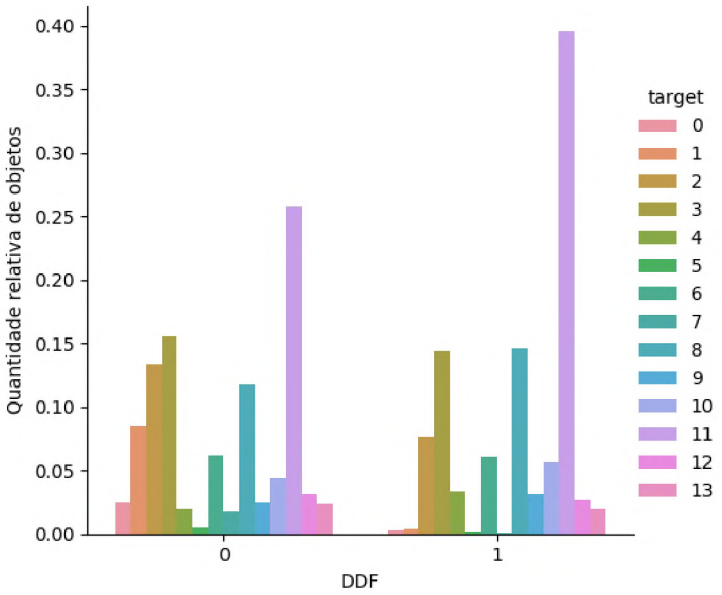

Figura 15 – Proporção de cada classes nas duas áreas de observações.

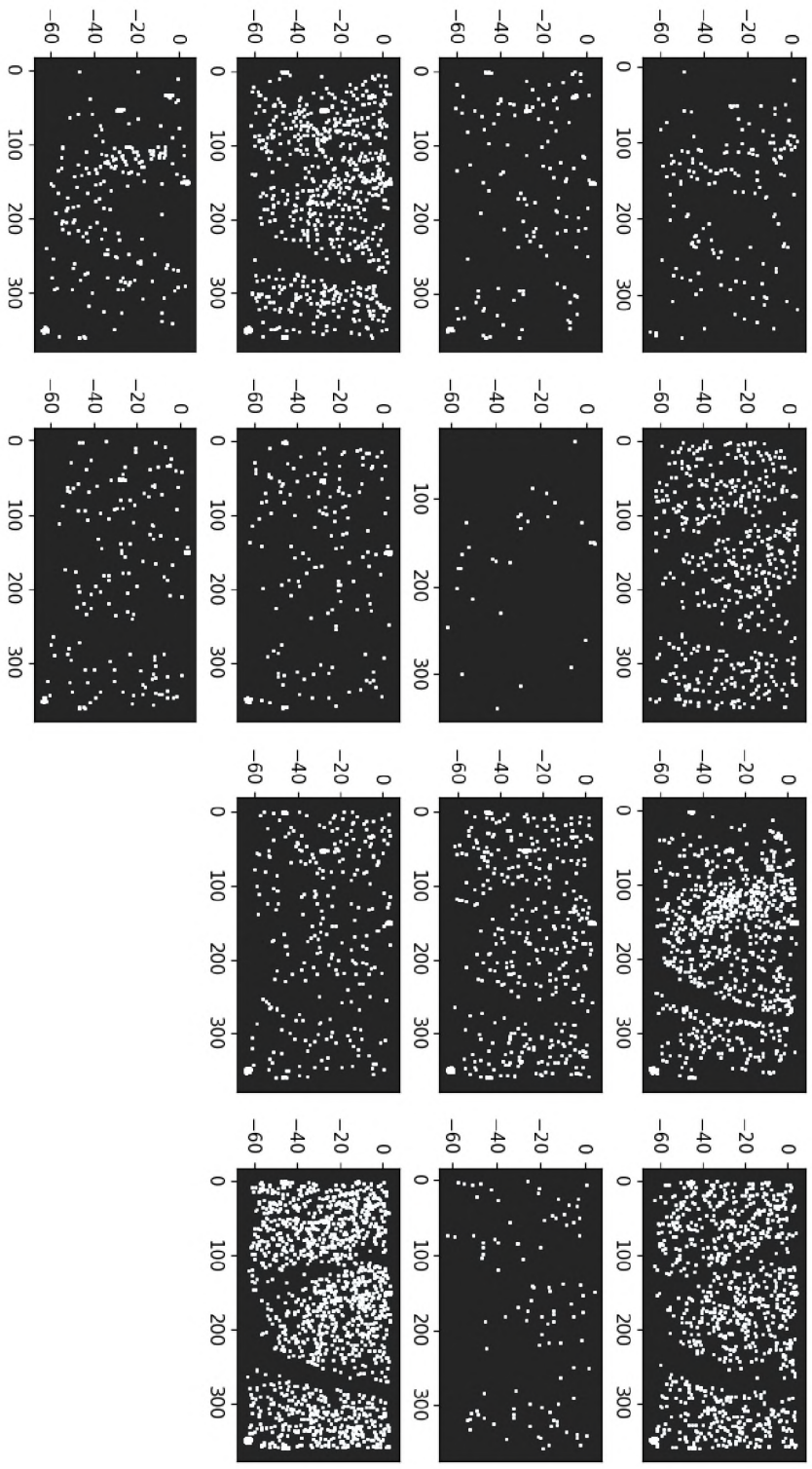

Figura 16 – Disposição das fontes nas coordenadas *right ascension* e *declination*.

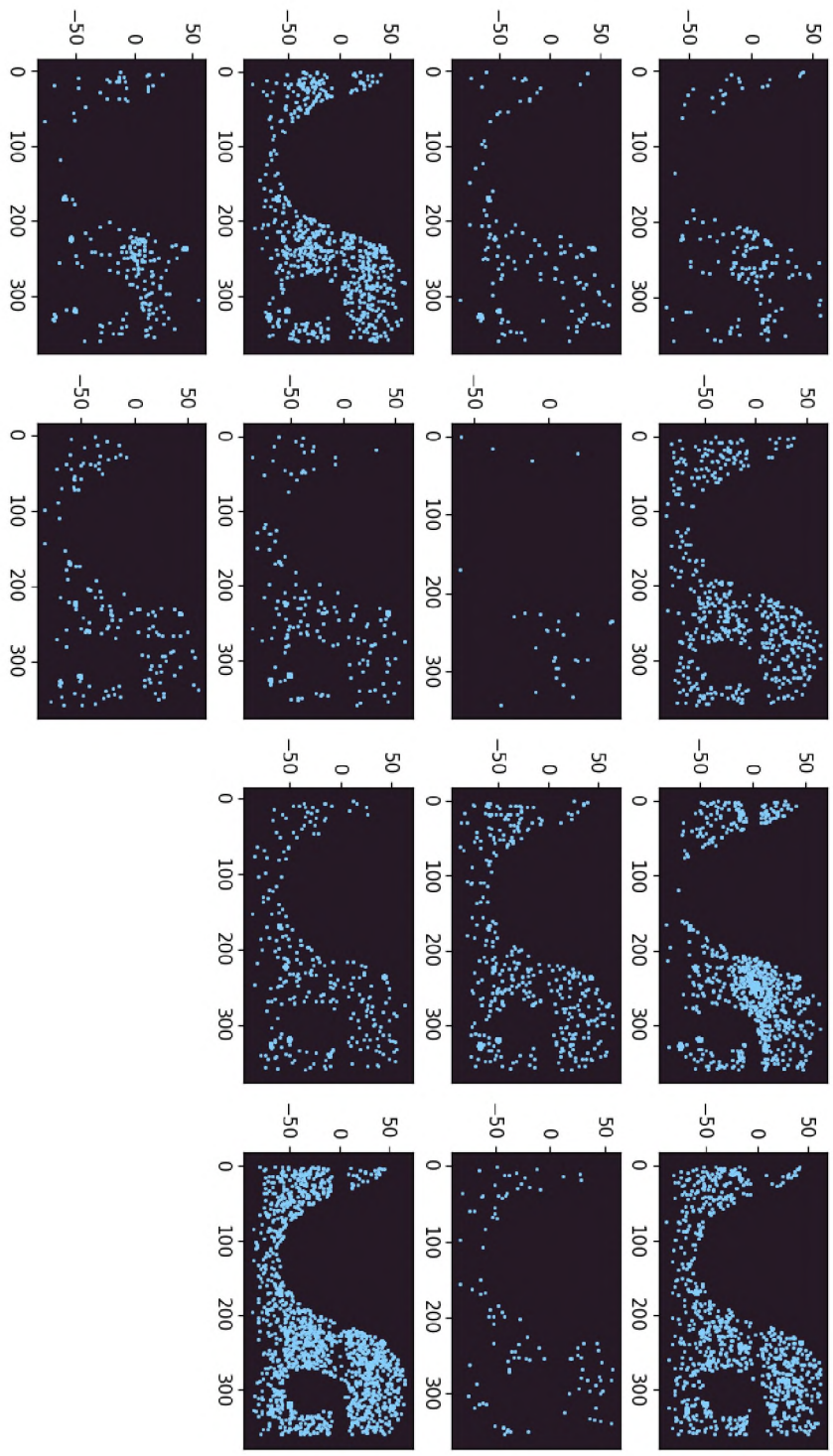

Figura 17 – Disposição das fontes nas coordenadas de latitude e longitude galácticas.

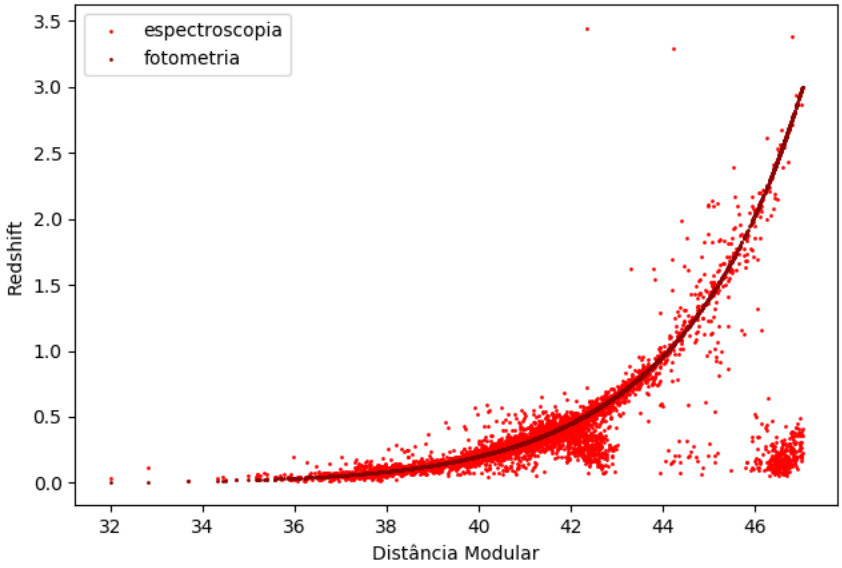

Figura 18 – Avermelhamento dos objetos de treino e suas distâncias modulares.

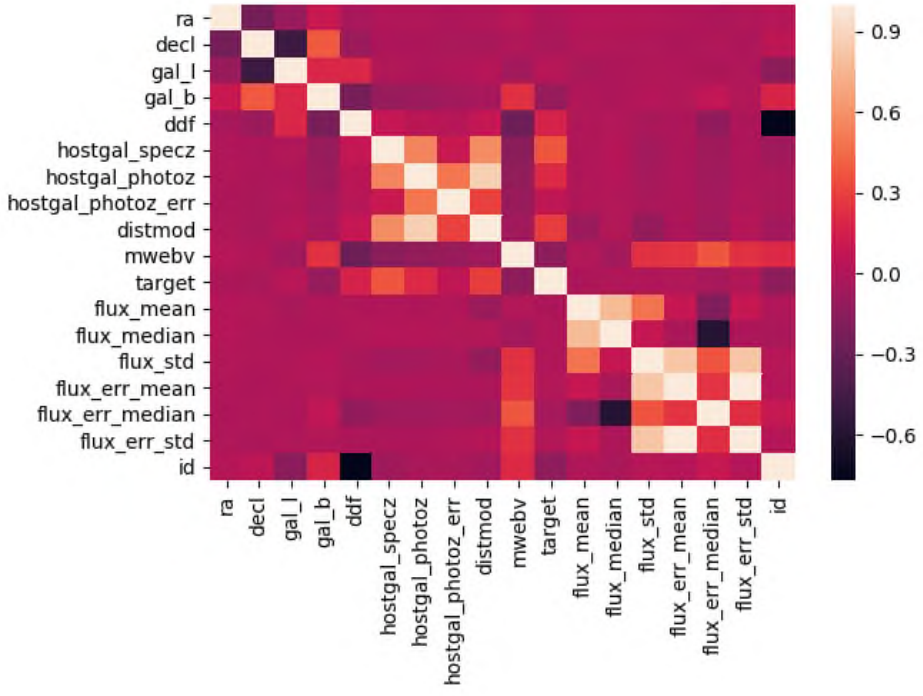

Figura 19 – Correlação entre as características extraídas do conjunto de treino.

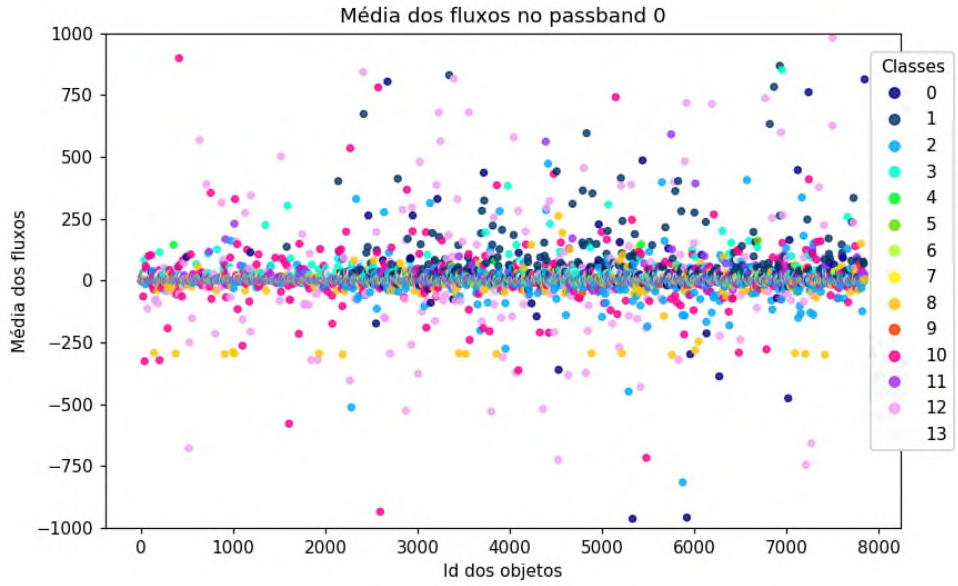

Figura 20 – Média dos fluxos observados no *passband u* de cada objeto com coloração baseada em sua classe.

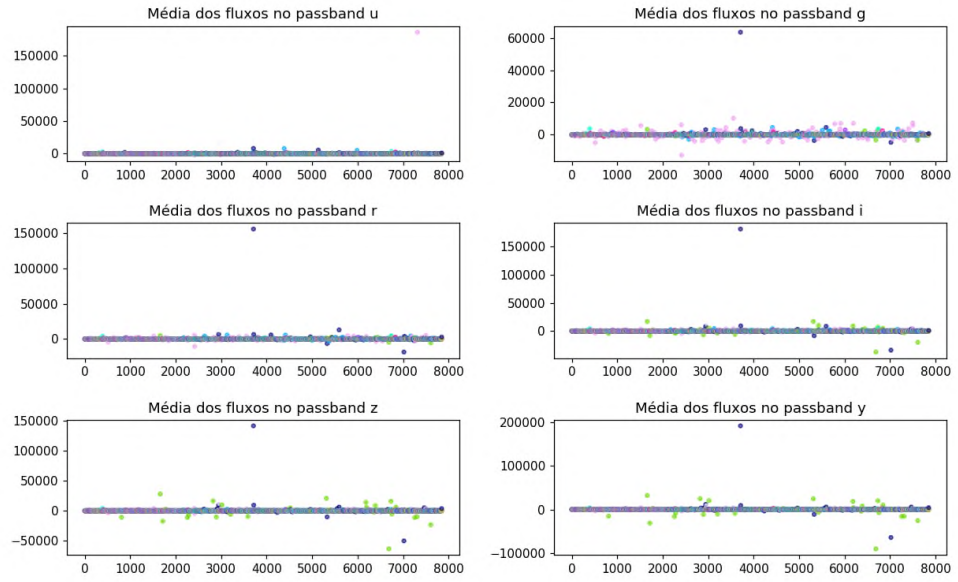

Figura 21 – Média dos fluxos observados de cada objeto em cada *passband* com coloração baseada em sua classe.

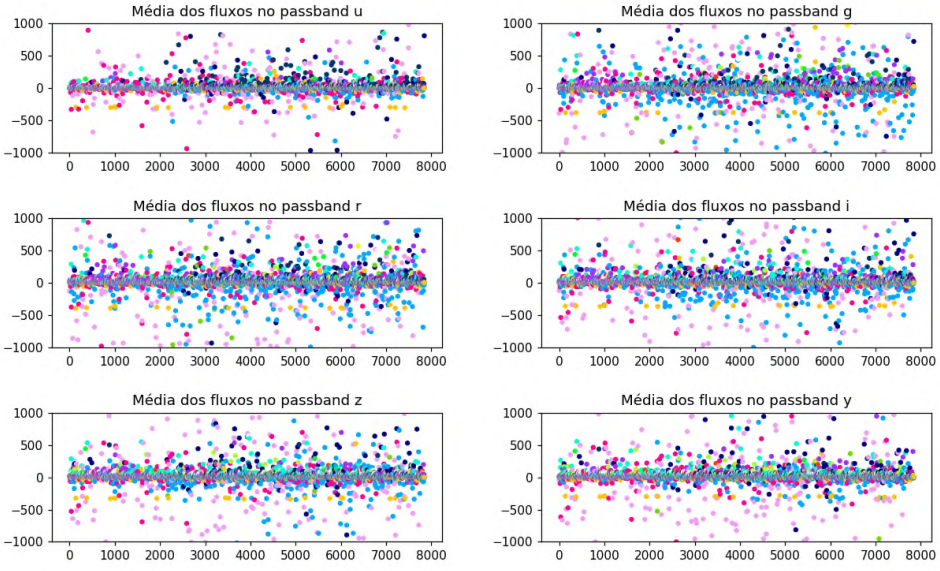

Figura 22 – Média dos fluxos observados de cada objeto em cada *passband* com coloração baseada em sua classe e eixo Y limitado.

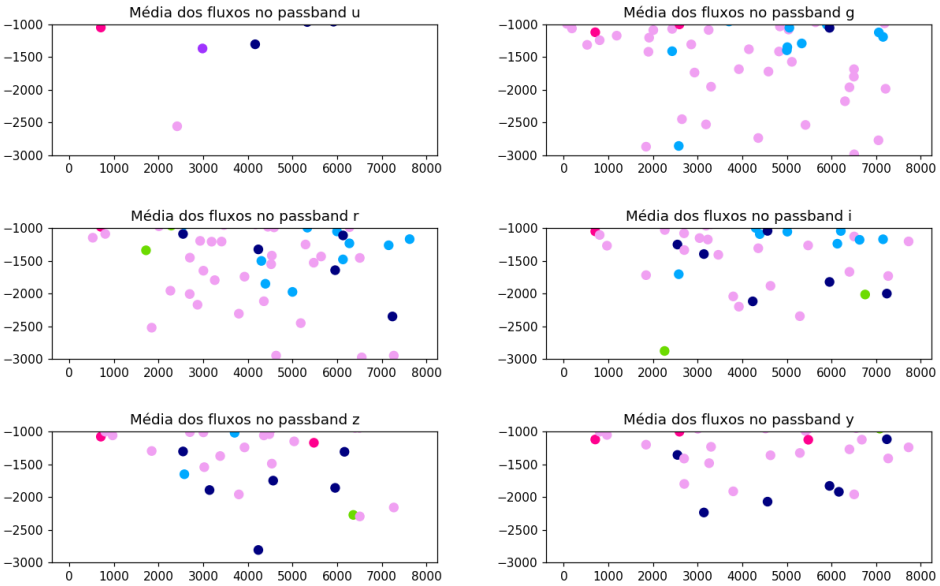

Figura 23 – Média dos fluxos observados na parte inferior da média.

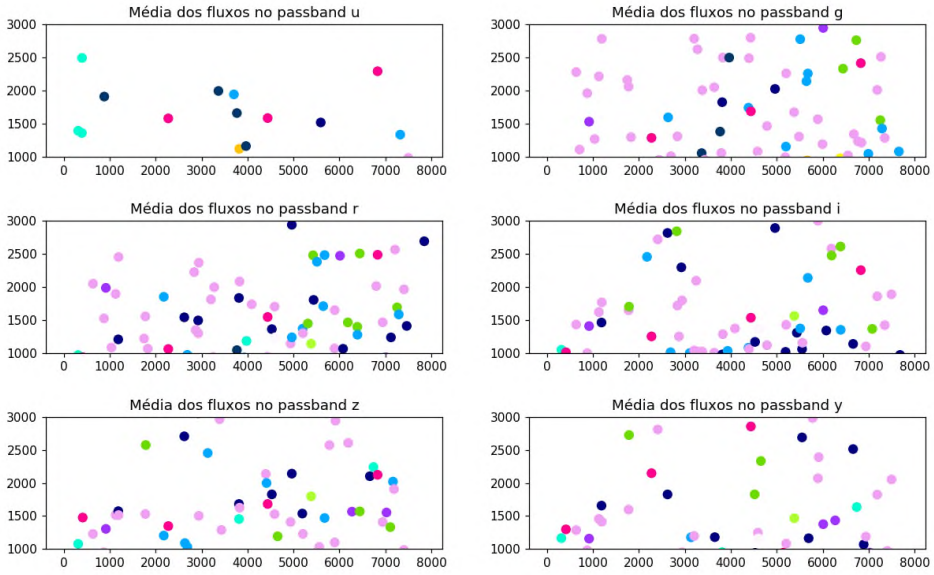

Figura 24 – Média dos fluxos observados na parte superior da média.

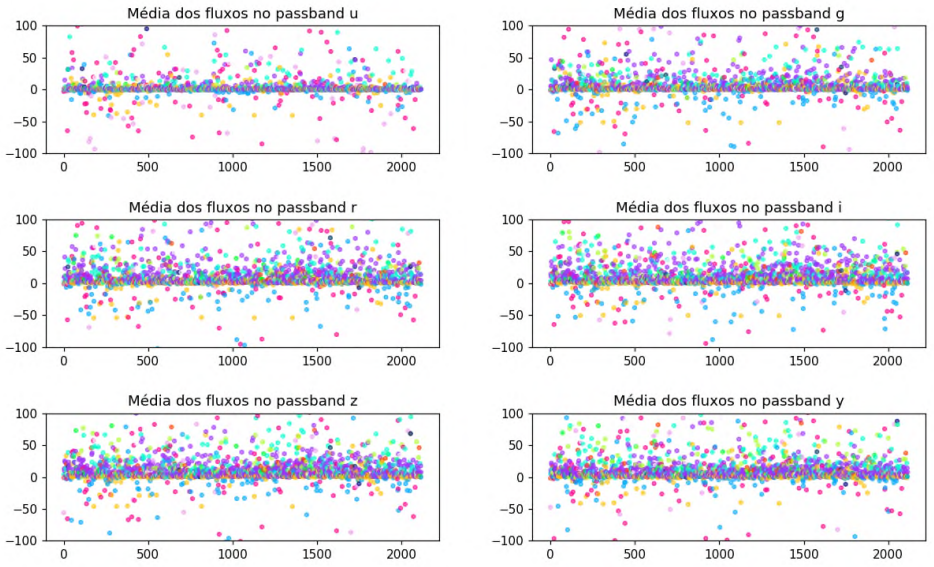

Figura 25 – Média dos fluxos observados limitados de −100 a 100 na área de DDF.

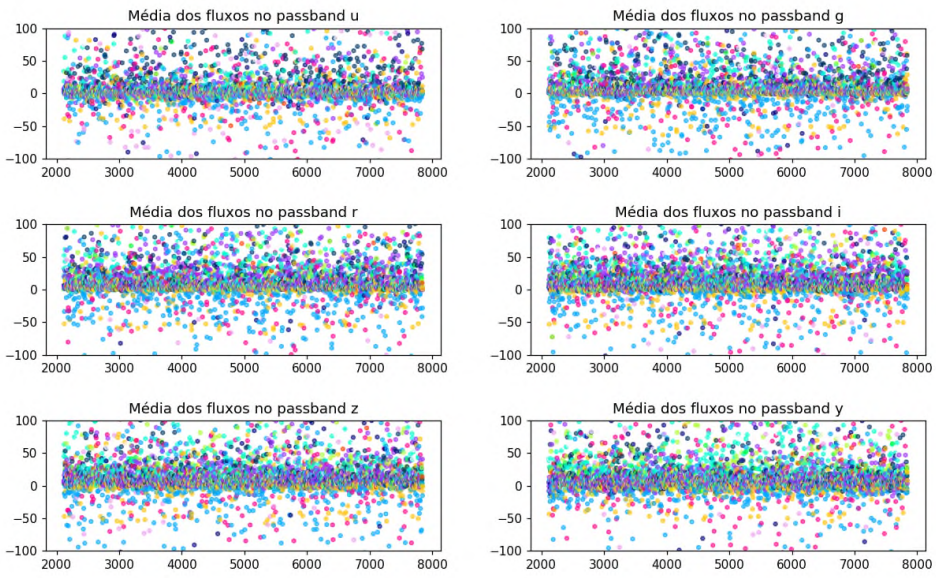

Figura 26 – Média dos fluxos observados limitados de −100 a 100 na área de WDF.

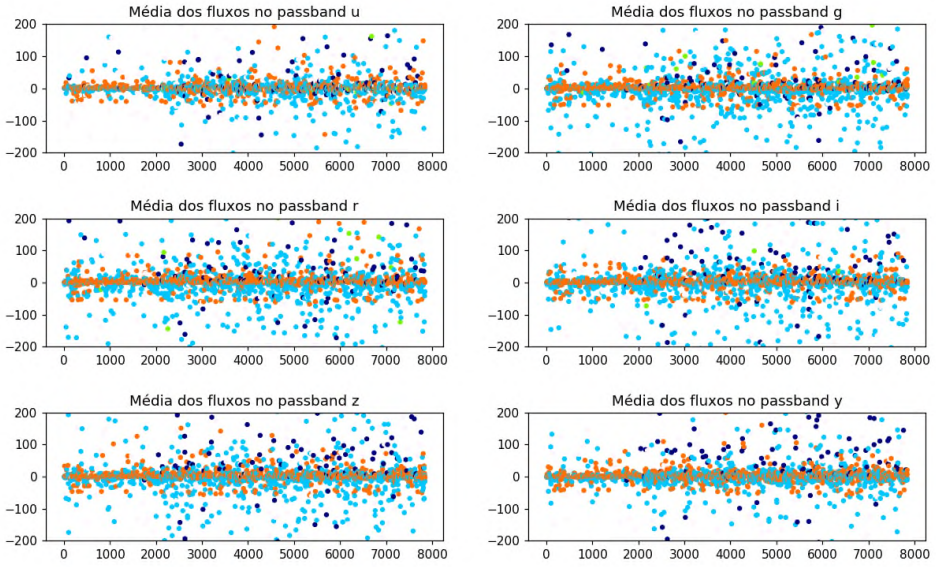

Figura 27 – Média dos fluxos observados, limitados de −200 a 200, das classes galácticas.

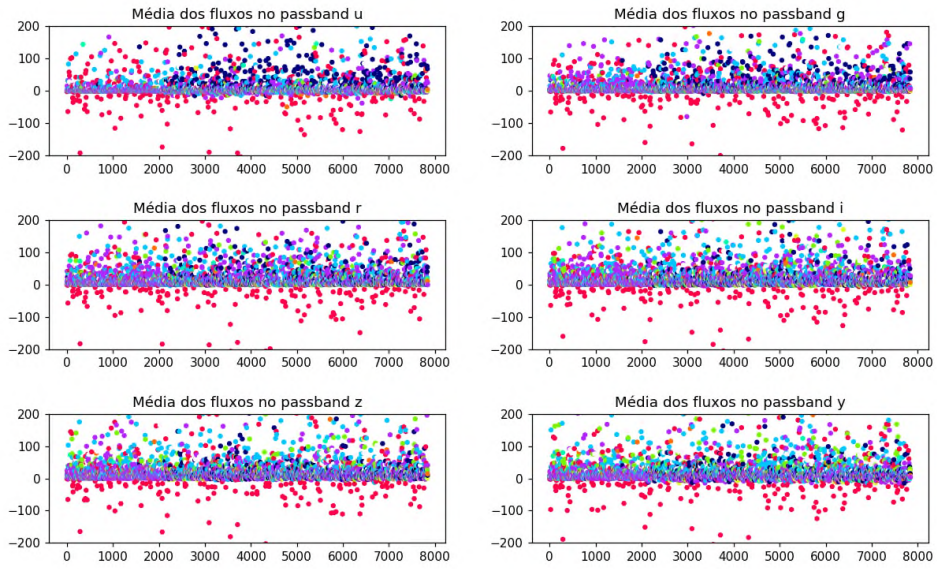

Figura 28 – Média dos fluxos observados, limitados de −200 a 200, das classes extragalácticas.

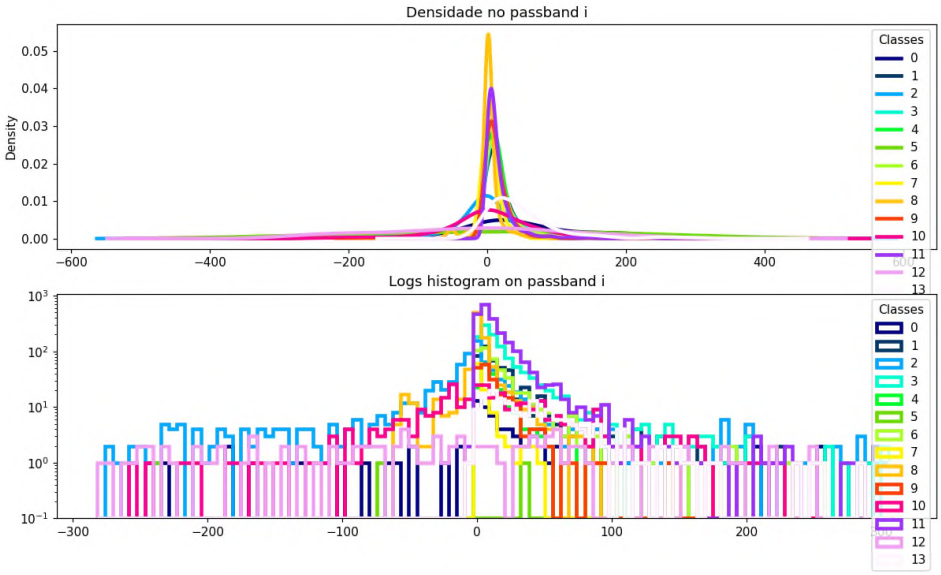

Figura 29 – Densidade e histograma das médias de cada classe diferenciadas pela cor.

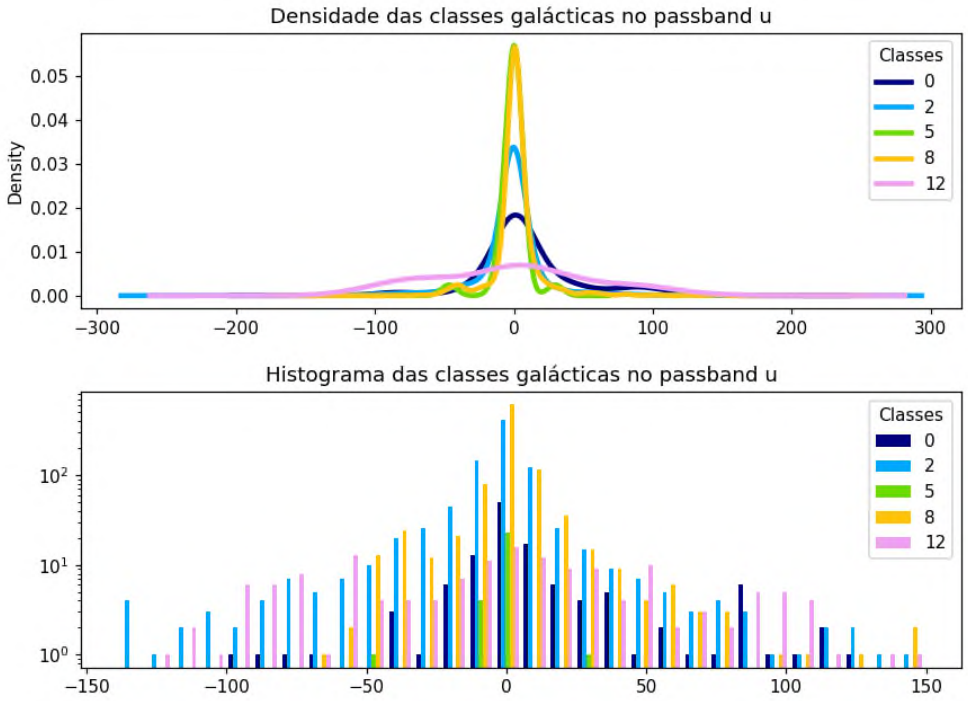

Figura 30 – Densidade e gráfico em barras das médias das classes galácticas diferenciadas pela cor.

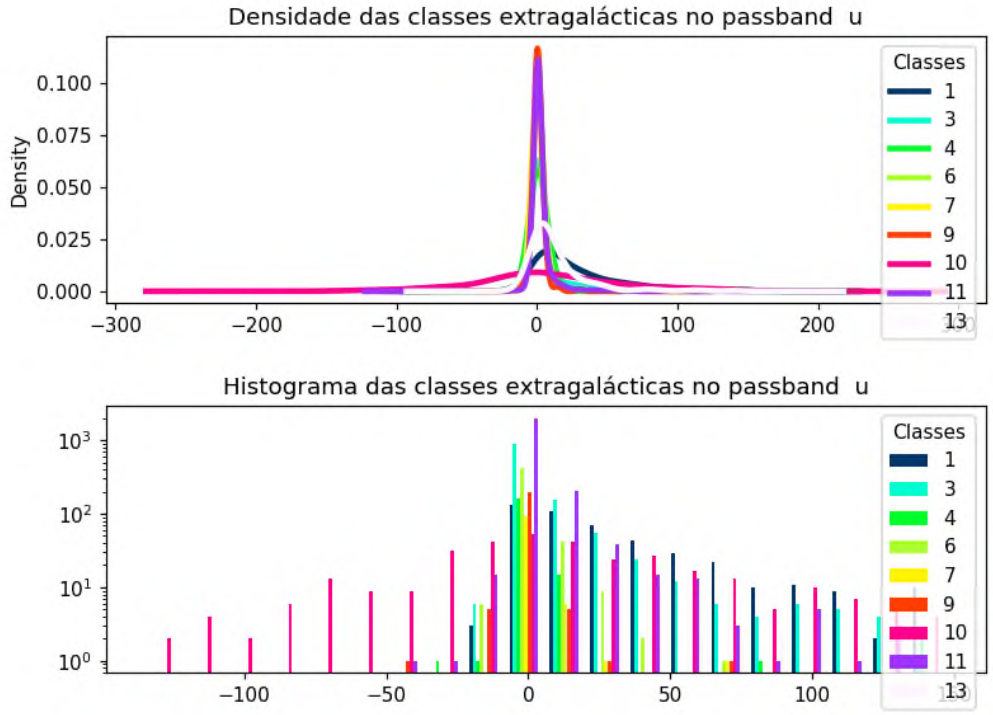

Figura 31 – Densidade e gráfico em barras das médias das classes extragalácticas diferenciadas pela cor.

5 Conclusão

Respondendo a pergunta proposta pela competição PLAsTiCC (LSSTC, 2019c), "Podemos classificar fontes astronômicas utilizando curvas de luz simuladas designadas a imitar os dados do LSST?", é possível classificar os objetos presentes nas observações simuladas, pois, os resultados obtidos foram melhores que o resultado de classificar todos os objetos pertencendo a cada de classe de maneira igual, com a mesma probabilidade. A pontuação das predições do classificador construído nesse trabalho de conclusão foi de 1,651, obtida através da própria competição hospedada pelo *site* Kaggle, enquanto predizer igualmente todas as classes para cada objeto resulta em uma pontuação de 2,80 como mostrado no *ranking* da competição (https://www.kaggle.com/c/PLAsTiCC-2018/leaderboard).

Para construir o modelo classificador elaborado por esse trabalho, alguns métodos de pré-processamento e classificação foram utilizados. Foi empregado uma limpeza de dados e uma redução de dimensionalidade para gerar as características aplicadas no treinamento do classificador que se utilizou do algoritmo LGBM. O *random sampling* foi aplicado para escolher as melhores características e para escolher o melhor modelo e seu parâmetros foi através da seleção de taxa de erro. Técnicas estas explicadas na Seção 2.2.3.

É possível notar como a falta de alguns métodos fazem muita diferença nas predições e em seu resultado. Algumas abordagens que poderiam colaborar com todo o modelo e que foram empregadas pelos competidores citados na Seção 2.5 são: crescimento da base de dados para treino de modo a diminuir as diferenças das distribuições entre o conjunto de treino e de teste; predizer a curva de luz dos objetos; extrair características das curvas de luz como, por exemplo, contar os picos da curva; utilizar uma função de validação no algoritmo LGBM; utilizar outros algoritmos de predição como, por exemplo, uma rede neural artificial; e utilizar outros métodos para classificar a classe 99 como, por exemplo, detecção de anomalias. Tais métodos podem ser um caminho para trabalhos futuros a partir do trabalho feito aqui.

Referências

APERS, S. D. M.; NARAYAN, G. *The Astronomical (complete) EDA - PLAsTiCC dataset*. 2019. PLAsTiCC kernel on Kaggle. Disponível em: <https://www.kaggle.com/danilodiogo/the-astronomical-complete-eda-plasticc-dataset>. Acesso em: apr. 2019. Citado na página 31.

BIANCO, F.; STREET, R. *Transients and Variable Stars LSST Science Collaboration*. 2019. Web page hosted on GitHub. Disponível em: <https://lsst-tvssc.github.io/>. Acesso em: 4 jul. 2019. Citado na página 11.

BOONE, K. *Naive Benchmark - Galactic vs Extragalactic*. 2019. PLAsTiCC kernel on Kaggle. Disponível em: <https://www.kaggle.com/kyleboone/naive-benchmark-galactic-vs-extragalactic/notebook>. Acesso em: apr. 2019. Citado na página 31.

BOONE, K. *Overview of 1st place solution*. 2019. PLAsTiCC discussion on Kaggle. Disponível em: <https://www.kaggle.com/c/PLAsTiCC-2018/discussion/75033>. Acesso em: 2 jun. 2019. Citado na página 24.

CANALLE, J. B. G.; MATSUURA, O. T. *Curso Astronáutica e Ciências do Espaço - Astronomia*. 1. ed. [S.l.]: AGÊNCIA ESPACIAL BRASILEIRA (AEB), 2007. Citado na página 16.

CDA-USP. *Coordenadas Celestes*. 2000. CDA - CDCC-USP. Disponível em: <http://www.cdcc.usp.br/cda/aprendendo-basico/esfera-celeste/coordenada-celeste/coordenada-estelar-gra.htm>. Acesso em: mar. 2019. Citado na página 15.

CESIUM, T. *Open-Source Machine Learning for Time Series Analysis*. 2019. Powered by Pelican and Zurb Foundation, Theme by Kenton Hamaluik. Disponível em: <http://cesium-ml.org/>. Acesso em: 1 jul. 2019. Citado 2 vezes nas páginas 26 e 27.

CHEN, S. *11th solution - very basic but may different methods*. 2019. PLAsTiCC discussion on Kaggle. Disponível em: <https://www.kaggle.com/c/PLAsTiCC-2018/discussion/75174>. Acesso em: 2 jun. 2019. Citado na página 24.

CHRIST, N. B. M.; NEUFFER, J. *tsfresh*. 2019. Hosted by Read the Docs. Disponível em: <https://tsfresh.readthedocs.io/en/latest/>. Acesso em: 1 jul. 2019. Citado 2 vezes nas páginas 26 e 27.

ERDEM, A. *4th Place Solution with Github Repo*. 2019. PLAsTiCC discussion on Kaggle. Disponível em: <https://www.kaggle.com/c/PLAsTiCC-2018/discussion/75011>. Acesso em: 2 jun. 2019. Citado na página 24.

GARRETA, A. *9th place solution*. 2019. PLAsTiCC discussion on Kaggle. Disponível em: <https://www.kaggle.com/c/PLAsTiCC-2018/discussion/75316>. Acesso em: 2 jun. 2019. Citado na página 24.

GRELLIER, O. *LB/CV scores*. 2019. PLAsTiCC discussion on Kaggle. Disponível em: <https://www.kaggle.com/c/PLAsTiCC-2018/discussion/69538>. Acesso em: jun. 2019. Citado 2 vezes nas páginas 19 e 29.

GRELLIER, O. *PLAsTiCC in a kernel meta and data*. 2019. PLAsTiCC discussion on Kaggle. Disponível em: <https://www.kaggle.com/ogrellier/plasticc-in-a-kernel-meta-and-data>. Acesso em: apr. 2019. Citado 4 vezes nas páginas 29, 33, 34 e 50.

HAN MICHELINE KAMBER, J. P. J. *Data Mining: Concepts and Techniques, third edition*. 3. ed. [S.l.]: Morgan Kaufmann Publishers, 2012. Citado 6 vezes nas páginas 16, 17, 18, 19, 30 e 39.

HUNTER, J. D. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, IEEE COMPUTER SOC, v. 9, n. 3, p. 90–95, 2007. Citado 2 vezes nas páginas 20 e 35.

INC., K. *How to use Kaggle*. 2019. Kaggle site. Disponível em: <https://www.kaggle.com/docs>. Acesso em: jul. 2019. Citado na página 20.

JR, T. A. et al. The photometric lsst astronomical time-series classification challenge (plasticc): Data set. *arXiv preprint arXiv:1810.00001*, v. 1, 2018. Citado 4 vezes nas páginas 11, 21, 22 e 31.

KEPPLER, S. O.; SARAIVA, M. de F. O. *Astronomia e Astrofísica*. 3. ed. [S.l.]: Departamento de Astronomia - Instituto de Física Universidade Federal do Rio Grande do Sul, 2014. Citado 3 vezes nas páginas 12, 14 e 16.

LIU, J. *Congrats and 8th place Rapids solution updated!* 2019. PLAsTiCC discussion on Kaggle. Disponível em: <https://www.kaggle.com/c/PLAsTiCC-2018/discussion/75012>. Acesso em: 2 jun. 2019. Citado na página 24.

LSSTC. *LSST Dark Energy Science Collaboration*. 2019. Web page hosted on GitHub. Disponível em: <https://lsstdesc.org/>. Acesso em: 4 jul. 2019. Citado na página 11.

LSSTC. *LSST Project Mission Statement*. 2019. Site for the LSST created by LSST Corporation. Disponível em: <www.lsst.org/about>. Acesso em: 20 apr. 2019. Citado na página 11.

LSSTC. *PLAsTiCC Astronomical Classification, Can you help make sense of the Universe?* 2019. Competition hosted by Kaggle. Disponível em: <https://www.kaggle.com/c/PLAsTiCC-2018>. Acesso em: mar. 2019. Citado 2 vezes nas páginas 21 e 64.

LSSTC. *PLAsTiCC data are unblinded*. 2019. Site for the PLAsTiCC created by LSST Corporation. Disponível em: <https://plasticc.org/data-release/>. Acesso em: 27 jun. 2019. Citado na página 35.

MELLO, H. *Dataset overview - Exploration and comments*. 2019. PLAsTiCC kernel on Kaggle. Disponível em: <https://www.kaggle.com/hrmello/dataset-overview-exploration-and-comments>. Acesso em: apr. 2019. Citado 3 vezes nas páginas 31, 40 e 45.

MICROSOFT. *LGBM Documentation*. 2019. Microsoft Open Source Code of Conduct. Disponível em: <https://lightgbm.readthedocs.io/en/latest/index.html>. Acesso em: jun. 2019. Citado na página 17.

MICROSOFT. *LGBM Parameters*. 2019. Microsoft Open Source Code of Conduct. Disponível em: <https://lightgbm.readthedocs.io/en/latest/Parameters.html>. Acesso em: jun. 2019. Citado 2 vezes nas páginas 18 e 34.

MICROSOFT. *LGBM Parameters Tunning*. 2019. Microsoft Open Source Code of Conduct. Disponível em: <https://lightgbm.readthedocs.io/en/latest/Parameters-Tuning.html>. Acesso em: jun. 2019. Citado na página 34.

NUMFOCUS. *Numpy*. 2019. NumFOCUS Sponsored-Projects. Disponível em: <https://www.numpy.org>. Acesso em: may. 2019. Citado 2 vezes nas páginas 19 e 33.

NUMFOCUS. *Python Data Analysis Library*. 2019. NumFOCUS Sponsored-Projects. Disponível em: <https://pandas.pydata.org>. Acesso em: may. 2019. Citado 2 vezes nas páginas 20 e 32.

O'BRIEN, M. *Train set vs Test set Differences EDA*. 2019. PLAsTiCC kernel on Kaggle. Disponível em: <https://www.kaggle.com/obrienmitch94/train-set-vs-test-set-differences-eda>. Acesso em: 7 jun. 2019. Citado 3 vezes nas páginas 25, 40 e 45.

PUGET, J. F. *Solution 5 tidbits (revised with code)*. 2019. PLAsTiCC discussion on Kaggle. Disponível em: <https://www.kaggle.com/c/PLAsTiCC-2018/discussion/75050>. Acesso em: 2 jun. 2019. Citado 2 vezes nas páginas 24 e 26.

PUGET, J. F. *Where is the gap?* 2019. PLAsTiCC kernel on Kaggle. Disponível em: <https://www.kaggle.com/cpmpml/where-is-the-gap>. Acesso em: apr. 2019. Citado 2 vezes nas páginas 31 e 41.

PUGET, J. F. *Write efficient code!* 2019. PLAsTiCC discussion on Kaggle. Disponível em: <https://www.kaggle.com/c/PLAsTiCC-2018/discussion/71398>. Acesso em: 8 jun. 2019. Citado na página 27.

SILVA, D. da. *The Astronomical (complete) EDA - PLAsTiCC dataset*. 2019. PLAsTiCC Kernel on Kaggle. Disponível em: <https://www.kaggle.com/danilodiogo/the-astronomical-complete-eda-plasticc-dataset>. Acesso em: apr. 2019. Citado na página 40.

SIONKOWSKI, G. *Use causality*. 2019. PLAsTiCC discussion on Kaggle. Disponível em: <https://www.kaggle.com/c/PLAsTiCC-2018/discussion/69696#410538>. Acesso em: jun. 2019. Citado na página 27.

TAN, M. S. P.-N.; KUMAR, V. *Introduction to Data Mining*. 1. ed. [S.l.]: Pearson, 2005. Citado na página 19.

TEAM, P. *The PLAsTiCC Astronomy "Starter Kit"*. 2019. Python nootbook on PLAsTiCC Kernels. Disponível em: <https://www.kaggle.com/michaelapers/the-plasticc-astronomy-starter-kit>. Acesso em: mar. 2019. Citado 3 vezes nas páginas 22, 31 e 40.

TROTTA, B. *14th place solution*. 2019. PLAsTiCC discussion on Kaggle. Disponível em: <https://www.kaggle.com/c/PLAsTiCC-2018/discussion/75054>. Acesso em: 2 jun. 2019. Citado 2 vezes nas páginas 24 e 29.

TROTTA, B. *14th place solution for the Kaggle Plasticc challenge*. 2019. Github Repository. Disponível em: <https://github.com/btrotta/kaggle-plasticc>. Acesso em: apr. 2019. Citado na página 29.

VANDERPLAS, J. *gatspy: General tools for Astronomical Time Series in Python*. 2019. Powered by Sphinx 1.3.5 Alabaster 0.7.6. Disponível em: <http://www.astroml.org/gatspy/>. Acesso em: 1 jul. 2019. Citado 2 vezes nas páginas 26 e 27.

ZAKI, M. J.; JR., W. M. *Data Mining and Analysis: Fundamental Concepts and Algorithms*. 1. ed. [S.l.]: Cambridge University Press, 2013. Citado 2 vezes nas páginas 16 e 17.