INSTITUTO
UNIVERSITÁRIO
DE LISBOA

Machine learning: challenges and opportunities on credit risk

Patrícia Alexandra Guerreiro Costa

MSc in Mathematical Finance

Dissertation supervised by:
Professor Ph.D , Diana Aldea Mendes, Associate Professor,
ISCTE Business School

October, 2022

Departament of Finance

Departament of Mathematics

Machine learning: challenges and opportunities on credit risk

Patrícia Alexandra Guerreiro Costa

MSc in Mathematical Finance

Dissertation supervised by:
Professor Ph.D , Diana Aldea Mendes, Associate Professor,
ISCTE Business School

October, 2022

Acknowledgements

I would like to thank Professor Doctor Diana Aldea Mendes, for her knowledge on machine learning algorithms passed on to me, for her valuable and constructive feedback to this research.

To my parents and brother, for always supporting me and teaching me the value of life and education.

A very special thanks to my partner, Tiago, for his continuous moral support, for encouraging me to not give up and for making me feel positive throughout this journey.

Resumo

O constante desafio na antecipação do risco de incumprimento por parte dos tomadores de crédito, levou a que as instituições financeiras desenvolvessem técnicas e modelos de forma a melhorar a sua monitorização do risco de crédito, e antever o quão provável será para determinados clientes entrar em incumprimento, assim como o quão provável será para outros de cumprirem com as suas obrigações financeiras. Portanto, interessa averiguar como as instituições financeiras podem antecipar esta ocorrência beneficiando de algoritmos de *Machine Learning*.

A presente dissertação pretende demonstrar o poder dos algoritmos de *Machine Learning* na análise de risco de crédito, com foco no processo de construção dos modelos, treinando-os e testando os dados, e apresentar as oportunidades e os desafios de *Machine Learning* que ainda estão em aberto para desenvolver futuros estudos. Para esse propósito, apresentamos dois algoritmos de classificação de *Machine Learning*: as Árvores de Decisão e a Regressão Logística. Adicionalmente, também se apresenta os resultados numéricos obtidos entre várias comparações desses algoritmos que foram programados e corridos em *Python*, utilizando a aplicação *Jupyter Notebook*. Os dados da amostra inicial, constituída por 850 observações, contêm detalhes de crédito sobre os tomadores de empréstimos nos Estados Unidos da América, sendo os dados de livre acesso e uitilização. Para verificar a execução e a performance do modelo, entre Regressão Logística e Árvores de Decisão, usamos medidas como o AUC, precisão e F1-score.

Palavras-Chave: Risco de Crédito; *Data Mining*; *Machine Learning*; Regressão Linear e Logística; Árvores de Decisão; Floresta Aleatória.

Classificação JEL: C02; C60.

Abstract

The constant challenge in anticipating the risk of default by borrowers has led financial institutions to develop techniques and models to improve their credit risk monitoring, and to predict how likely it is for certain customers to default on a loan, as well as how likely it is for others to meet their financial obligations. Thus, it is interesting to investigate how financial institutions can anticipate this occurrence using Machine Learning algorithms.

This dissertation aims to demonstrate the power of Machine Learning algorithms in credit risk analysis, focusing on building the models, training them, and testing the data, and presenting the opportunities and challenges of Machine Learning that are still open to developing future studies. For this purpose, we present two Machine Learning classification algorithms: Decision Trees and Logistic Regression. In addition, numerical results obtained from various comparisons of these algorithms, which were programmed and ran in Python using the Jupyter Notebook application, are also presented. The initial sample data, consisting of 850 observations, contained credit details about borrowers in the United States of America, and is freely available data. To check the model execution and performance, between Logistic Regression and Decision Trees, we used measures such as AUC, precision and F1-score.

Keywords: Credit Risk; Data mining; Machine Learning; Linear and Logistic Regression; Decision Trees; Random Forest.

JEL Classification: C02; C60.

Contents

List of Figures

List of Tables

List of Acronyms

AI	Artificial Intelligence
AUC	Area Under Curve
BCBS	Basel Committee on Banking Supervision
CM	Correlation Matrix
DM	Data Mining
DT	Decision Tree
EAD	Exposure at Default
FPR	False Positive Rate
LGD	Loss at Given Default
ML	Machine Learning
PD	Probability of Default
RF	Random Forest
ROC	Receiver Operator Characteristic
SVM	Support Vector Machine
TPR	True Positive Rate
VIF	Variance Inflation Factor

CHAPTER 1

Introduction

Credit risk consists of the risk that money owed could not be repaid (Gestel & Baesens, 2009), and it has long been an important and widely studied topic in banking. Credit risk remains the most critical and challenging risk for many commercial banks to manage and evaluate. In the last years, advances in information technology have lowered the costs of acquiring, managing, and analysing data to build more robust and efficient techniques for credit risk management (Pacelli & Azzollini, 2011).

Credit scoring is a statistical method used to predict the probability that a loan applicant or existing borrower will default (Mester, 1997). Hand and Jacka (1998) stated that "the process (by financial institutions) of modeling creditworthiness is called credit scoring". Credit scoring aims to help lenders or financial institutions make decisions more quickly and objectively by quantifying the financial risk involved (Chye *et al.,* 2004).

The Bank Regulation aims to keep the solvency of the banks by avoiding excessive risk. The Basel Committee on Baking Supervision (BCBS) was responsible for the Accords on Capital Adequacy, commonly known as Basel I and Basel II (Goodhart, 2011). Basel I has introduced a ratio of bank capital to risk-weighted assets for credit risk only (Penikas, 2015). Basel II has announced risk-based capital requirements, reviewed the capital adequacy and assessment process, and established market discipline and sound banking practices (Kashyap & Stein, 2004). Risk parameters were created to assess the bank's credit risk exposure and to have a consistent and common criterion to compare the various risks to be managed. The Probability of Default (PD), the Loss of Given Default (LGD), Exposure at Default (EAD), and, Expected Loss (EL) are risk parameters used to estimate the capital that banks need to secure each loan. According to Basel II and under the IRB (internal ratings-based) approach, financial institutions can use their estimated risk parameters to calculate regulatory capital. It also considers the maturity (M) of exposures (Basel, 2001). With Basel III, the quality and quantity of capital increased, additional buffers were established, a leverage ratio was introduced, the liquidity was improved, and the liquidity coverage ratio (LCR) was designed (King & Tarbert, 2011).

Machine learning is one of the most critical developments in the digital transformation industry. While this advanced technology has been around for decades, it is now becoming

commercially viable. We're moving into an era where machine learning algorithms are essential tools to create value for businesses that want to understand the hidden value of their data (Hurwitz & Kirsch, 2018).

Machine learning (ML) teaches machines to optimize performance criteria using example data or past experience (Alpaydin, 2020). Its purpose is to learn from the data by the algorithm itself without being explicitly programmed. The kind of algorithm employed depends on the problem you wish to solve, the number of variables, the type of model that would suit it best, and so on (Mahesh, 2020).

The first section of this thesis will introduce the concepts of Bank Regulation, and the evolution of international banking regulations (Basel I, Basel II and Basel III) put forth by the Basel Committee on Bank Supervision (BCBS). The second section will exhibit an overview of credit risk concepts and parameters and describe the credit risk model selected for this work. The third section will focus on a branch of Data Mining, specifically Machine Learning, and we presented four ML models: Linear Regression, Logistic Regression, Decision Trees, and Random Forests. The fourth section will focus on data exploratory analysis and modeling, showing the data pre-processing and manipulation and the building of ML models (Logistic Regression and Decision Trees). Also, the numerical results will be interpreted, and the comparisons between the two ML models will be presented. The final section will conclude our work and present future research opportunities.

CHAPTER 2

Bank Regulation

The BIS – *Bank for International Settlements*, in 1974, created the BCBS- *Basel Committee on Banking Supervision* that aimed a creation of bank supervision standards, risk management and, good practices in banks at the international level, thus, strengthening the soundness and stability of the international banking system

2.1. Basel I

In 1988, the BCBS established a global principle for banks to measure capital adequacy, known as the Basel Capital Accord, published in Basel, Swiss. The Basel I requires financial institutions to maintain enough capital reserves so that their capital ratio exceeds 8%, with capital ratio defined as:

$$capital\ ratio = \frac{total\ capital}{total\ credit\ risk} \quad (1)$$

The 1988 Accord required a minimum ratio of capital to risk-weighted assets (RWA) of 8% and is estimated according to the following formula:

$$K \geq 0.08 \sum_{i=1}^{n} RW_i A_i \quad (2)$$

where K is the minimum capital requirement, RW_i is the risk weight assigned to asset i and A_i is the credit risk exposure of asset i .

Advantages and positive effects of the implementation of Basel I standards:

i. Substantial increases in capital adequacy ratios of internationally active banks;
ii. Relatively simple structure;
iii. Worldwide adoption;
iv. Increased competitive equality among internationally active banks;
v. Greater discipline in managing capital;
vi. A benchmark for assessment by market participants.

Weaknesses of Basel I standards

However, several researchers such as (Moosa, 2008) and (Mendes, 2013) developed studies that pointed out weaknesses of this regulation:

i. Capital adequacy only depends on credit risk, ignoring other risks (e.g., liquidity, market and, operational);
ii. Did not consider innovations in the financial markets or the correlation factors;
iii. The temporal structure is not considered in the risk weighting of assets;
iv. In credit risk assessment, there is no difference between debtors of different credit quality and ratings;
v. Emphasis is on book values and not market values;
vi. Inadequate assessment of risks and effects of the use of new financial instruments, as well as risk mitigation techniques.

2.2. Basel II

Published in 2004, Basel II focused on strengthening the capital requirements of banks by reinforcing three pillars:

1) Minimum capital requirements make a bank's capital more risk-sensitive;
2) Promoted enhanced risk management tactics among large banks;
3) Market discipline to motivate safe and sound baking practices.

The main difference between Basel II and Basel I is that Basel II incorporates the credit risk of assets held by financial institutions to determine regulatory capital ratios.

Basel II defined the capital ratio as:

$$capital\ ratio = \frac{total\ capital}{total\ credit\ risk + market\ risk + operational\ risk} \quad (3)$$

2.3. Basel III

In the context of the 2008 Global Financial Crisis on Banks, it was verified that many banks held insufficient capital levels, and some of this capital did not have sufficient loss absorbency.

In particular, the first element of strengthening the prudential rules is increasing the quantity and quality of the regulatory capital. The new regulatory framework of Basel 3 also seems to

confirm the sensibility of the Committee about the increasingly sophisticated models for calculating capital charges and managing credit risk.

Basel III appeared to "improve the banks' ability to handle shocks from financial stress and to strengthen their transparency and disclosure" (CFI).

The new regulatory framework of Basel III is a set of reform measures from the previous accords. Still, it introduces the necessary ones to increase banks' capital adequacy, particularly regarding liquidity risk. So, Basel III amended requirements concerning:

1) Capital resources – minimum ratio requirements were increased, and additional buffers were introduced;
2) Loss-absorbing capability – capital items have differing abilities to absorb losses, so eligibility criteria were tightened.

As a result, banks were forced to increase their capital levels, hold more common equity-type capital and replace debt instruments that did not meet the new eligibility criteria.

CHAPTER 3

Credit risk

Credit risk is when financial obligations are not paid when they fall due to the borrower or counterparty ("obligor") is either unable or unwilling to pay.

While there is no standard definition of default, the definition provided by Basel Committee on Banking Supervision (BCBS) captures the most salient features:

"*A default is considered to have occurred with regards to a particular obligor when either or both of the two following events have taken place:*

- *The bank considers that the obligor is unlikely to pay its credit obligation to the banking group in full, without recourse by the bank to actions such as realizing security (if held);*
- *The obligor is past due more than 90 days on any material credit obligation to the baking group. Overdrafts will be considered as being past due once the customer has breached an advised limit or been advised of a limit smaller than current outstanding*."

("International Convergence of Capital Measurement and Capital Standards: A Revised Framework", Basel Committee on Banking Supervision (BCBS), 2006)

A credit transaction includes the transfer of a liquid asset, frequently cash, from a lender to a borrower. The borrower promises to return the amount lent, called the principal, plus an additional amount, the interest, at some future date.

The risk of default arises from the risk that the debtor will be unable to repay the debt of a loan within a certain period, resulting in a default that may result in a total or partial loss of the amount financed.

3.1. Risk Parameters

We must consider the critical parameters required for credit risk modeling to quantify the credit risk and accurately estimate the risk taken.

Measuring credit risk is non-trivial, but the conceptual framework is intuitive. There are critical inputs to the measurement process (these factors are explicitly set out as the standard inputs in calculating risk-based capital under Basel III).

The factors that affect credit risk range from borrower-specific criteria, such as debt ratios, to market-wide considerations, such as economic growth. The idea is that liabilities can be objectively valued and predicted to help protect against financial loss. There are several significant variables to consider: the financial health of the borrower, the severity of the consequences of a default for the borrower and the creditor; the size of the credit extension; historical trends in default rates; and a variety of macroeconomic considerations. Three possible factors are consistently identified as having a stronger correlation to credit risk. (Ross, 2019)

i) *Probability of Default* (PD) is the probability that a borrower, over a specified period (typically one year), will be unable to meet the debt obligations.

ii) *Loss Given Default* (LGD) is the amount of money a bank or other financial institution loses when a borrower defaults on a loan. When a borrower defaults, the lender has a recoverable amount called *Recovery Rate* (RR= 1 – LGD).

iii) *Exposure at Default* (EAD) is the total value a bank is exposed to at the time of default of its borrower.

iv) *Expected Loss* (EL) is the potential losses financial institutions may incur if they lend to a company that defaults.

$$EL = PD \times EAD \times LGD \tag{4}$$

3.2. Credit Risk Models

3.2.1. Merton

Merton (1974) proposed a model based on the Black Scholes option pricing model (BSM) that estimated a company's default risk. The BSM model is used to determine the fair prices of stock options based on six variables: volatility, type, underlying stock price, strike price, time, and risk-free rate. It is based on the principle of hedging and focuses on eliminating risks associated with the volatility of underlying assets and stock options (Corporate Finance of Institute, 2021).

The following formula gives the price of a call option C:

$$C(S_t, t) = N(d_1)S_t - Ke^{-r\Delta T}(d_2) \quad (5)$$

where:

$$d_1 = \frac{ln\left(\frac{V_t}{K}\right) + (r + \frac{\sigma^2}{2})\Delta T}{\sigma_v\sqrt{\Delta T}}$$

$$d_2 = d_1 - \sigma_v\sqrt{\Delta T}$$

The price of a put option P is given by:

$$P(S_t, t) = Ke^{-r\Delta T}S_t + C(S_t, t) \quad (6)$$

Where:

N: Cumulative distribution function of the standard normal distribution. It represents a standard normal distribution with mean equal to 0, and standard deviation equal to 1

S_t: Spot price of the underlying asset

K: Strike price

r: Risk-free rate

ΔT: Time to maturity (in years)

σ : Volatility of returns of the underlying asset

Volk (2014) defends that one of the Merton model's significant weaknesses is market prices for the asset value. To overcome this problem, Harmele *et al.* (2003) and Harmele *et al.* (2004) suggested a latent variable approach, and according to these authors, the default event is modeled as a random variable Y_{it}, which may be interpreted as the value of the firm's assets, where t denotes the actual time and a single period. The two possible states at time t for borrower i, i =1,...,N, "default" and "non-default" are modeled by the indicator variable Y_{it}^*:

$$Y_{it}^* = \begin{cases} 1 & \text{borrower } i \text{ defaults in } t \\ 0 & \text{else} \end{cases} \quad (7)$$

The default event is equivalent to the value of a firm's assets crossing a threshold it c_{it} at t:

$$Y_{it} < c_{it} \Leftrightarrow Y_{it}^* = 1 \quad (8)$$

The probability that a borrower i will default in time t, given survival until time $t-1$, is denoted by the threshold model:

$$\lambda_{it} = P(Y_{it} < c_{it} \mid Y_{it-1} \geq c_{it-1}) \tag{9}$$

Equivalently $T_i = t$ means that default happens at time t, the condition $T_i > t-1$ means that firm i did not default before t. The conditional default probability is given by:

$$\lambda_{it} = P(T_i = t \mid T_i > t-1) \tag{10}$$

Also called a discrete-time hazard rate (Harmele *et al.*, 2003).

The available data is crucial to the observability of the value of a firm's assets Y_{it}. If the asset value is observable, then the model is linear. Otherwise, a non-linear model is estimated, treating the asset value as a latent variable.

CHAPTER 4

Data mining

Data mining is known as "knowledge discovery in databases". It is the process of discovering patterns in datasets that are useful in decision-making and provide tools by which data can be automatically analysed. The accessibility and abundance of data today make data mining a matter of considerable importance and necessity (Fayyad *et al.*, 1996), (Bose & Mahapatra, 2001), (Rokach & Maimon, 2014). Figure 1 shows a general view of the data mining taxonomy.

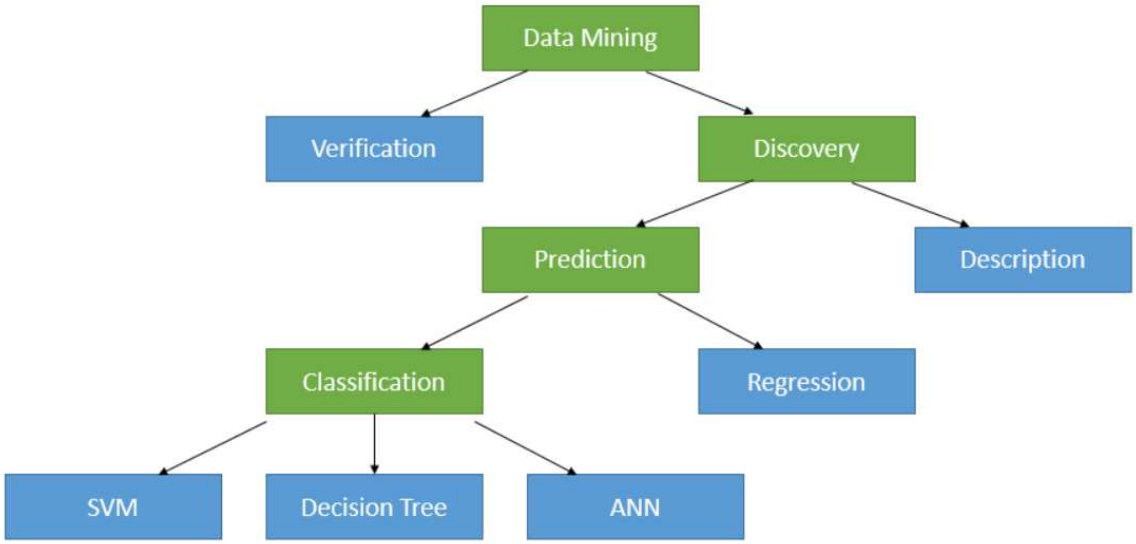


Figure 1 - Taxonomy of data mining methods based on Rokach and Maimon, 2014.

The learning algorithms are generally classified into two main classes in data mining: supervised and unsupervised learning. Supervised feature selection determines feature relevance by evaluating the feature's correlation with the class. Unsupervised feature selection exploits data variance and separability to evaluate feature relevance (He *et al.*, 2005; Dy & Brodley, 2004).

Supervised learning describes a scenario where the "experience" contains essential information (Shalev-Shwartz & Ben-David, 2014) and generates a function that maps inputs to desired outputs. (Ayodele, 2010). Thus, the data's structure is already known, and the objective is to allocate new data to the correct classes (in the context of classification problems).

Figure 2 illustrates a supervised learning scheme about how supervised learning detects the correct output in a large dataset.

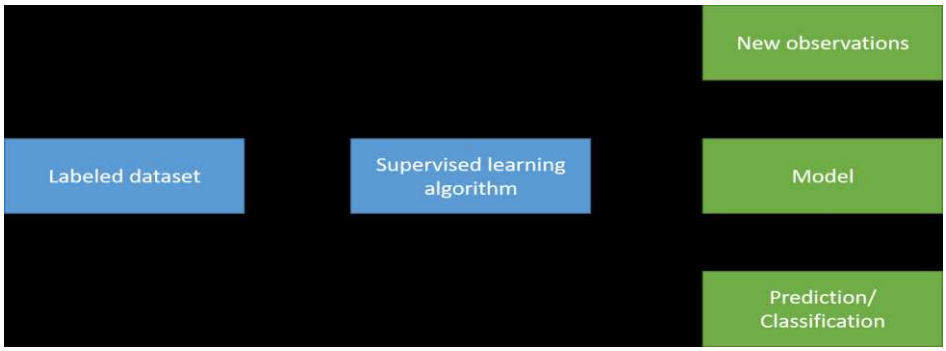


Figure 2 - Scheme of Supervised Learning (Sethi, 2020)

Although supervised learning can offer businesses advantages, such as deep data insights and improved automation, some challenges occur when building sustainable supervised learning models. The following are some of these challenges:

- Supervised learning models can require certain levels of expertise to structure accurately;
- Training supervised learning models can be very time-intensive;
- Datasets can have a higher likelihood of human error, resulting in algorithms learning incorrectly;
- Unlike unsupervised learning models, supervised learning cannot cluster or classify data on its own.

Unsupervised learning is a type of Machine Learning that works and learns within the data to discover structures and patterns observed randomly. The variety of tasks includes density estimation, dimensionality reduction, and clustering. (Bengio *et al.*, 2021)

Usually, it works with unlabeled data. (Sathya & Abraham, 2013)

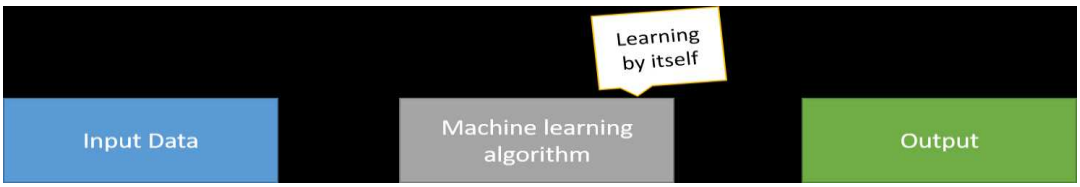


Figure 3 - Scheme of Unsupervised Learning (Sethi, 2020)

The first step of unsupervised learning is to learn a classifier with few data labeled ("Input Data" in Figure 3). Afterward, the classifier is used to predict unlabeled data, and the predictions with higher reliability are added to the training set, the classifier is retrained with the new

training set. This process (see Figure 3) is repeated until no new data can be added to the training set.

4.1. Machine Learning

Machine Learning (ML) is a branch of artificial intelligence and has been growing recently. It was developed from pattern recognition and the theory that computers can be programmed to do specific tasks by introducing data, making and improving predictions or behaviors based on data, and automating knowledge acquisition (Langley & Simon, 1995).

Donald Hebb created a model of brain cell interaction, where he presented theories on neuron excitement and communication between neurons in 1949.

Arthur Samuel, in 1952 developed a computer program for playing checkers that became better as more games were played. It was a self-learning program given that all combinations seen until the moment were recorded and remembered with the values of the reward function.

In 1967, the nearest neighbor algorithm was created, and it was the start of pattern recognition, which allowed to find similarities when comparing the known data with new data.

Since 1990 Machine Learning has been used in language learning fields, and adaptive software systems, but it is more recently that ML has been applied to different areas in our day-to-day. Most of the significant advancements in technology, such as self-driving vehicles, individually adjusted prices on e-commerce shops, GPS applications on smartphones, movie recommendations, and suggested products to buy, are increasingly using ML algorithms. High-stakes decisions in loan applications are not an exception. (Mehrabi *et al.*, 2021).

Today, ML models are trendy in various specialized environments. Businesses are looking for Machine Learning techniques to help them anticipate the future and create competitive differentiation. AI, especially Machine Learning, has a significant role in credit risk assessment in the banking industry and can provide unprecedented levels of automation. For instance, ML algorithms can do repetitive tasks, allowing managers time for sophisticated challenges. Any financial institution should be up to date to know the clients' needs, what products they look for, and in which conditions they could become a good or a wrong customer by having an accurate digital footprint. At the same time, the banking industry is becoming more competitive. The banks that already are using ML have more success, since Machine Learning is very efficient in assessing customers' needs, increases customer retention, and saves money on acquiring new ones. Also, ML provides many advantages for the banks, such as improving decision-making, better risk management and more precise risk assessment, detection and

prevention of fraud, personalized customer service, internal operational solutions, chatbots, marketing, and lending solutions.

As the data is continuously changing and constantly growing, ML can adapt to new data and models. So, previous computations and trials can produce reliable and repeatable decisions and results. It's possible to create models quickly that can analyse complex and bigger data, simultaneously delivering faster and more accurate results, gaining advantages over competitors and having the opportunity to predict the future.

Nowadays, according to Cielen *et al.* (2016), there are many applications for machine learning algorithms, such as:

- Finding oil fields, gold mines, or archeological sites based on existing sites (classification and regression);
- Finding place names or persons in text (classification);
- Identifying people based on pictures or voice recordings (classification),
- Recognizing birds based on their whistle (classification);
- Identifying profitable customers (regression and classification);
- Proactively identifying car parts that are likely to fail (regression);
- Identifying tumors and diseases (classification);
- Predicting the amount of money a person will spend on product X (regression);
- Predicting the number of eruptions of a volcano in a period (regression);
- Predicting your company's yearly revenue (regression);
- Predicting which team will win the Champions League in soccer (classification).

When we talk about advantages, we also must talk about disadvantages. Specific risks and fears are associated with AI and ML, such as lower trust due to less human contact when chatting with a robot on a web page and less confidence when providing personal and sensitive information (Holzinger, 2018). Data privacy concerns have become extremely serious with the usage of ML algorithms, as they need a lot of personal information to produce accurate results (Qiu *et al.*, 2016). Also, there are ethical risks associated with the amount of data that companies need to collect, store, systematize, analyse and use for their benefit, like training and testing ML algorithms using this data (Char *et al.*, 2018). The idea that robots will replace humans in their jobs is not true, because it is not technology that determines employment patterns but the other way around. Even though, AI and ML can create countless new roles/positions in the banking industry, job cuts occur for repetitive tasks that ML algorithms can do (Fleming, 2019).

In this thesis, we are particularly interested in classification algorithms. Classification is one of the most common task of Machine Learning and is a problem of classification unknown instance in one of the pre-offered categories/classes. To do a classification of an object is needed to analyse its characteristics and find similarities with predetermined objects that are members of different classes, and then, with a certain accuracy, classify into one of the classes. The task is to build a model that will classificate new objects, based on the characteristics of objects already classified and known in advance. (Novaković *et al.,* 2017)

In what follows, we introduce the basic concepts related to these models' design, properties, and performance.

4.1.1. Machine Learning models

To analyse the ML algorithms and compare their performance, we need to evaluate them based on three main criteria:

i) Speed: The time required by the algorithm to do the classification;

ii) Accuracy: The correct predictions that the algorithm makes are divided by the total of forecasts;

iii) Noise tolerance: how the ML algorithms can work with noise as irrelevant features.

4.1.1.1.Linear Regression and Logistic Regression

Linear and logistic regression are Machine Learning algorithms that are part of supervised learning models.

The goal of regression is to predict the value of one or more continuous target variables y, given the value of a D-dimensional vector x of *input* variables (Bishop, 2006).

Multiple linear regression, in matrix form, can be defined as:

$$y = w^T X + \varepsilon \tag{11}$$

where y is a vector of response variables, X is a matrix of explanatory variables with an intercept constant, w is a vector of parameters and ε is a vector of error terms, usually Gaussian white noise.

Rather than modeling this response y directly, logistic regression models the probability that y belongs to a particular category (Gareth *et al.*, 2013).

Logistic regression is a supervised method for two-class classification problems (Hosmer & Lemeshow, 2013). For example, we can consider 1 for the default class and 0 otherwise. After the model receives the input parameters, it will calculate the probability that each observation will belong to one of the two classes.

The sigmoid function, used to define the logistic regression, is given by the following equation (Bishop, 2006):

$$\sigma(a) = \frac{1}{1+e^{-a}} \tag{12}$$

According to Harrington (2012), the probability that the applicant has defaulted can be written as logistic sigmoid acting on a linear function of the explanatory variables, that is:

$$p(x) = \sigma(w^T x) = \frac{1}{1+e^{-w^T x}} \tag{13}$$

where x is a vector containing the explanatory variables and an intercept constant, and w is the coefficient vector describing the variables' influence on the class probabilities.

Concluding, the formula for non-default loans will be 1- p.

4.1.1.2. Decision Trees

A decision tree is a classification algorithm based on a tree structure, where each internal node represents a test on some attribute of the instance. Each possible value of that attribute of the instance corresponds to a branch of the tree, and each final leaf node predicts the value of the class variable (Breiman *et al.*, 1984).

Decision trees represent a disjunction of conjunctions of constraints on the attribute values of instances. Each path from the tree root to a leaf, corresponds to a conjunction of attribute tests, and the tree itself to a disjunction of these conjunctions. It is a method for approximating discrete-valued functions robust to noisy data and capable of learning disjunctive expressions. These learning methods are among the most popular inductive inference algorithms and have been successfully applied to a broad range of tasks, from learning to diagnose medical cases to learning to assess the credit risk of loan applicants (Mitchell, 1997).

A decision tree is illustrated in Figure 4. The highest decision node is the root node, and the feature that sorts the data is chosen through the root node. This approach is applied for each sub-category of the training data, and it ends when all information is allocated to specific

classes. To build a decision tree, we choose the attribute with the most minor Gini index for the root node.

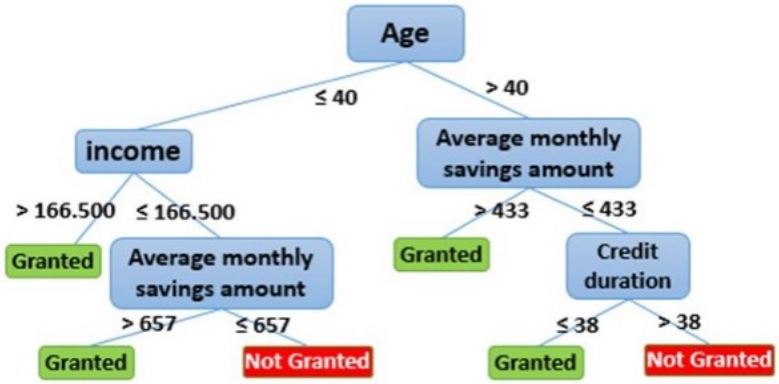


Figure 4 - Source: Machine learning for Banking: Loan approval use case | by Youssef Fenjiro | Medium

From the classification and regression trees (CART) algorithms, the C4.5 algorithm by Quinlan (2014) is the most used decision tree approach, an extension of the ID3 algorithm. C4.5 and its predecessor, ID3, are using formulas based on information theory to evaluate the "goodness" of a test; in particular, they choose the test that extracts the maximum amount of information from a set of cases, given the constraint that only one attribute will be tested. (Quinlan, 2014).

The Information Gain and Gini index are splitting measures that help to decide which attribute should be the root node or which features will be internal or leaf nodes. Information Gain is based on the entropy concept and identifies the best split in the decision trees algorithm. It is calculated by subtracting the weighted entropies of each class from the original entropy, and it is given by the following formula (Mahjoobi *et al.*, 2008):

$$E(S) = \sum_{i=1}^{c} -p_i log_2 p_i \tag{14}$$

Another definition for gain, used by the CART algorithm, was given by Breiman *et al.* (1984), was denoted by Gini index, and is expressed as follows:

$$G = 1 - \sum_{i=1}^{n} (p_i)^2 \tag{15}$$

where p_i is the proportion of data points (in a particular leaf node) assigned to class i.

Advantages and Challenges

The main advantage of decision tree algorithms is that some decision rules can be produced in a way that is easy to understand by humans and can be easily interpreted even by those who are not experts in the area.

Trees can easily handle qualitative predictors without creating dummy variables (Gareth, *et al.*, 2013).

The C4.5 algorithm, compared with other ML algorithms, has a very good combination of classification speed and error rate (Tjen-Siem *et al.*, 2000). Also, the C4.5 algorithm can handle the issue of incomplete data, and it can work with discrete and continuous data.

The main disadvantages of DT are that they require large amounts of memory to store the entire tree for deriving the rules (Cios *et al.*, 2007), and trees generally do not have the same level of predictive accuracy as some of the other regression and classification approaches (Gareth *et al.*, 2013).

Breiman *et al.* (1984) proposed two techniques to prevent overfitting training data: stopping the training algorithm before the perfect fit is reached and pruning the decision tree by removing sections from the tree that contain little information. This way, the complexity of decision trees is reduced.

4.1.1.3. Random Forest

The random forest (RF) scheme was proposed by Breiman (2001). According to him, random forests are a combination of tree predictors. Each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest.

Random forest is a tree-based supervised learning model. It is an ensemble learning method that can be used for regression and classification problems. It works well with categorical and continuous variables and can handle missing values and outliers (Breiman, 2001).

RF is a combination of hundreds of decision trees, and each decision tree is trained on a different sample of observations. Then, the final prediction of RF is made by an average of predictions of each tree. Thus, the results have a more precise correction of the overfitting issue and enable the model to give a more accurate prediction. Moreover, the RF algorithm is very stable, even if new data is introduced, the overall algorithm is not affected as the new data may impact one tree and not all the trees.

The disadvantages of RF are the longer training period because it requires a lot of time to be trained compared with DT and the complexity, which results from the creation of hundreds

of decision trees, so, usually, an RF algorithm needs much more computational power and resources than DT (Yang *et al.,* 2019).

CHAPTER 5

Data and Results

The present work aims to compare two Machine Learning models (Logistic Regression and Decision Tree) applied to our data on the credit scoring area and determine which one has better performance and accurate results to reduce the risk of decision-making. Thus, it will be possible to predict the future payment behavior of existing debtors to identify the wrong customers to provide more attention and assistance, reducing the likelihood of these customers defaulting. Also, it will be possible to forecast the future behavior of a new credit applicant by predicting loan default chances or poor repayment behaviors at the time the credit is granted. This model is called Probability of Default (PD).

The data contains the credit details of USA credit borrowers, has 850 observations, and the dataset was programmed and launched in Python. It is non-simulated data of cross-sectional type concerning a specific time period.

Variables:

- Age: Age of costumer;
- Ed: Education level of customer;
- Employ: Tenure with current employer (in years);
- Address: Number of years in the same address;
- Income: Customer income;
- Debtinc: Debt to income ratio;
- Creddebt: Credit to Debt ratio;
- Othdebt: Other debts;
- Default: Customer defaulted in the past (1=Defaulted, 0=Never defaulted).

The dependent variable is Default, and the independent variables are: Age, Ed, Employ, Adress, Income, Debtinc, Creddebt and Othdebt.

5.1. Empirical analysis

The first step of the process is to manipulate and prepare the data. We included all the variables in the study since they are all relevant. From the first checks, we can determine if there are null

values on each variable, and in this case, we found 150 observations with the default variable in blank. Missing values can occur for various reasons, for example, the customer decides not to disclose their information for personal reasons, or the data can be non-applicable for some clients. Considering these observations are irrelevant to the case study's target, the most straightforward option is to omit the observations or variables with missing values. So, for that reason, we will consider only 700 observations with all variables filled. Thus, we can conclude through Figure 5 that 560 customers never defaulted, and the remaining 140 customers already defaulted at least once.

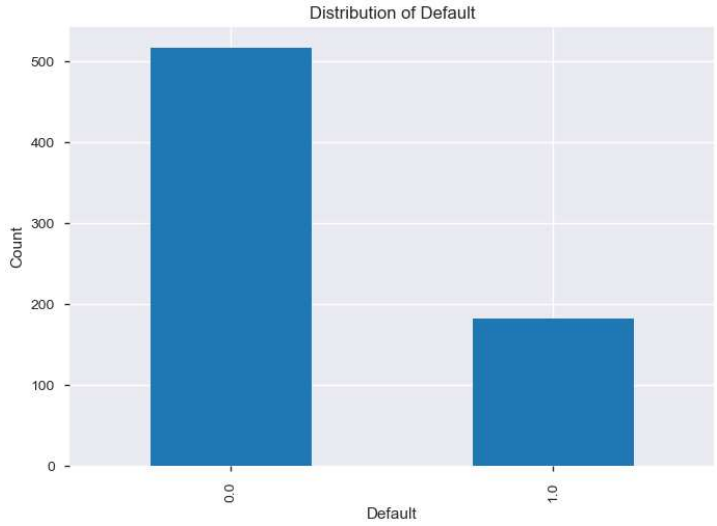


Figure 5- Distribution of Default Graph

Table 1 shows a statistical summary with the number of observations of each variable, the respective mean, the standard deviation, the minimum and maximum value and, the three quartiles. Also, we can see that are few outliers, it is the case for variable *income,* where the minimum value is 14 and the maximum value is 446.

	count	mean	std	min	25%	50%	75%	max
age	700.0	34.860000	7.997342	20.000000	29.000000	34.000000	40.000000	56.00000
ed	700.0	1.722857	0.928206	1.000000	1.000000	1.000000	2.000000	5.00000
employ	700.0	8.388571	6.658039	0.000000	3.000000	7.000000	12.000000	31.00000
address	700.0	8.278571	6.824877	0.000000	3.000000	7.000000	12.000000	34.00000
income	700.0	45.601429	36.814226	14.000000	24.000000	34.000000	55.000000	446.00000
debtinc	700.0	10.260571	6.827234	0.400000	5.000000	8.600000	14.125000	41.30000
creddebt	700.0	1.553553	2.117197	0.011696	0.369059	0.854869	1.901955	20.56131
othdebt	700.0	3.058209	3.287555	0.045584	1.044178	1.987567	3.923065	27.03360
default	700.0	0.261429	0.439727	0.000000	0.000000	0.000000	1.000000	1.00000

Table 1 - Statistical summary

The correlation matrix shows Pearson's correlation values that measure the degree of the linear relationship between each pair of variables. Its value lies between -1 and +1, -1 indicating total negative linear correlation, 0 indicating no linear correlation and, 1 indicating positive linear correlation. Furthermore, Pearson correlation, *r*, is invariant under separate changes in location and scale of the two variables, implying that for a linear function, the angle to the x-axis does not affect *r*. To calculate *r* for two variables *X* and *Y*, one divides the covariance of *X* and *Y* by the product of their standard deviations.

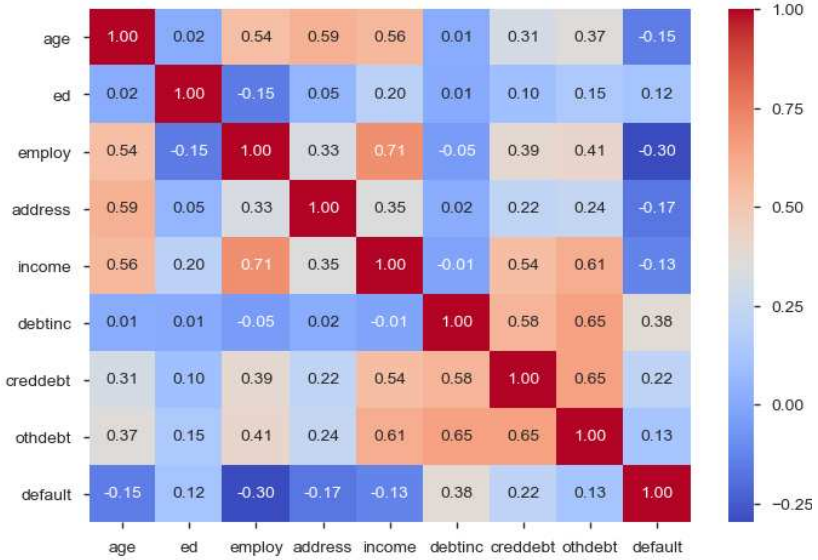


Figure 6 - Correlation seaborn heatmap

Analysing Figure 6, we verify a strong correlation between *income* and *employ* with a coefficient of 0.708981. This high correlation is expected in normal circumstances because if people are not employed, they will not have an income, on the other hand, the longer the customer has been employed, the higher the income is expected. Moreover, *othdebt* and *debtinc* have a high correlation with a coefficient of 0.645411*, othdebt* and *creddebt* with a coefficient of 0.645723. Those high and positive correlation values indicate that the variables measure the same skill or characteristic, i.e., the tendency to incur in debt (or to ask a new credit line to pay debt), taking into account other debts already incurred. Nevertheless, they can measure different characteristics if the variables are not highly correlated. Variables with a correlation greater than 0.7 are considered to be highly correlated.

Evaluating the Boxplots in Appendix for different variables, we can conclude that with the given dataset, people with lower age tend to default more than people with higher age. This is because younger people are employed less time than older people, and those considered senior

will have a higher income than the junior in their jobs. Thus, people who are recently employed tend to default more than seniors. Also, people with higher *creddebt* tend to default more often, which is expected, as well as those with higher *debtinc* ratio.

Data Exploratory analysis

Linear regression is used to quantify the relationship between a predictor variable and a response variable. The objective is to know if there is significance of regression coefficients, i.e., to determine if the independent variables are statistically significant for the model, for this check we performed a T-test.

The null hypothesis, H_0, states that the independent variables are not relevant compared to the dependent variable (Default). If P-value is high for some of the independent variables, it means they are not significant for the model, and then we can exclude them based on P-value.

	Variable Name	T-Statistic	P-Value
0	age	-3.67181	0.000259235
1	ed	3.04981	0.00237651
2	employ	-7.7948	2.3472e-14
3	address	-4.40472	1.22551e-05
4	income	-1.87974	0.0605604
5	debtinc	11.1754	8.65746e-27
6	creddebt	6.66875	5.24867e-11
7	othdebt	3.89121	0.000109296

Table 2 - T-test for training data

The results presented in Table 2, considering a P-value of 0.05, show that null hypothesis is rejected and we can conclude that there is a statistically significant relationship between the dependent variable and the independent variables. Although, *income* variable is slightly insignificant cannot be ignored based on T-test ((P-value=0.0605). Thus, all the other variables (*age*, *ed, employ, address*, *debtinc*, *creddebt, othdebt*) are statically significant for the model.

Multi Collinearity check

VIF (Variance Inflation Factor) measures the amount of multicollinearity in multiple regression variables. The smallest possible value for VIF is 1, which indicates the complete absence of collinearity. In practice, there is a small amount of collinearity among the predictors.

As a rule of thumb, a VIF value that exceeds 5 indicates a problematic amount of collinearity (Gareth *et al.*, 2013).

From the VIF check (Table 3), we found out that the correlation between the variables is within the acceptable limit, however, *age* and *ed* appear to be not significant for modeling as they have a low VIF and a high P-value.

	VIF Factor	Features
1	1.549126	address
2	2.068488	age
3	2.923224	creddebt
4	5.029469	debtinc
5	1.292434	ed
6	2.623756	employ
7	5.888177	income
8	5.322683	othdebt

Table 3 - VIF for each feature

Implementing the Logistic Regression model

The second step is implementing the two models (Logistic Regression and Decision Trees) and understanding which one performs better.

ML algorithms require quality training data and testing data to make precise predictions. The data that feeds the algorithm is called training data and builds up the ML algorithm. Hence, the model repetitively assesses the data to learn about the data's behavior and then, makes adjustments to meet the objective. During the training, validation data manage new data into the model that has not been evaluated before and provide the first test against unknown data, allowing us to understand if the model is making good predictions based on the new data. After the model is built, testing data will validate if the model can make accurate predictions. So, test data provides a final and real check of an unseen dataset and confirms that the ML algorithm was trained efficiently.

As different models have singular features, we need to analyse various metrics (such as confusion matrix, Area Under Curve (AUC), F1 score, and accuracy) that assess our ML algorithm.

Data sampling has been done based on 80-20% rule, therefore, out of the 700 observations in our dataset, we considered the proportion of 560 observations for training data to develop our model, and 140 observations for test data, which will be helpful for data validation.

The following results are about the implementation of **Logistic Regression**. We take into account all the independent variables as relevant for our model.

Confusion Matrix

A confusion matrix, also known as an error matrix, describes the performance of the ML classification algorithm, where we can see the relationship between true positives (the cases in which we predicted a customer would "Not Default" and the actual output was also "Not Default"), false positives (the cases in which we predicted "Not Default" and the actual output was "Default"), true negatives (the cases in which we predicted "Default" and the actual output was "Default") and false negatives (the cases in which we predicted "Default" and the actual output was "Not Default"). Table 4 illustrates the contingency table of binary classification.

		Predicted	
		Negative	Positive
Actual	Negative	True Negative	False Positive
	Positive	False Negative	True Positive

Table 4 - Made by authors

The Confusion matrix associated with the test set is illustrated in Figure 7 and shows the number of correctly classified cases (on the diagonal of the matrix) and those that were misclassified as the other category. In this case, the precision score of the model is 66.7% and the accuracy of the model is 80.7%, which is quite expectable, as the dependent variable is non-balanced, in other words, we have more clients that never defaulted than the opposite. To evaluate the model's performance, we can use a test set where we can have a balanced number between clients that never defaulted and those that went into default at least once.

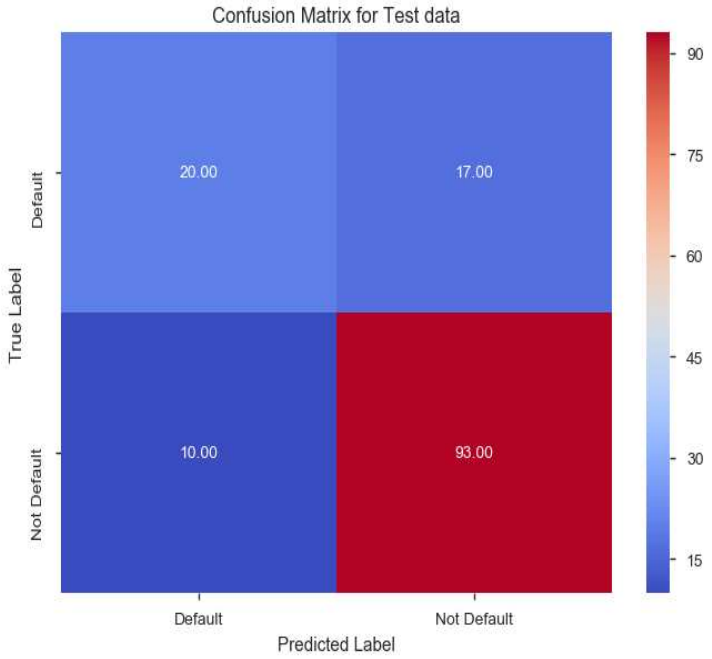


Figure 7 - Confusion matrix for Test Data

Before defining AUC, we need to understand some concepts applied to analyse the AUC. According to Fawcett (2004), from the confusion matrix, several common metrics can be calculated from it.

True positive Rate (Sensitivity)

$$True\ positive\ Rate = \frac{true\ positives}{true\ positives + false\ negatives} \tag{16}$$

True Negative Rate (Specificity)

$$True\ negative\ Rate = \frac{true\ negatives}{true\ negatives + false\ positives} \tag{17}$$

False Positive Rate (FPR)

$$False\ positive\ Rate = \frac{false\ positives}{true\ negatives + false\ positives} \tag{18}$$

Area Under Curve

Area Under Curve is the area under the curve resulting from plotting the False Positive Rate vs True Positive Rate at different points in the range of [0,1]. There is no realistic classifier with

an AUC lower than 0.5 (Bradley, 1997). The closer this area is to 1, the better is the performance of the model. The AUC performs a global assessment, including the whole score distribution, and the AUC test is based on ROC (Receiver Operator Characteristic) curves. The ROC curve is a two-dimensional graph in which the True Positive Rate (TPR) represents the y-axis and the False Positive Rate (FPR) is the x-axis (Tharwat, 2021).

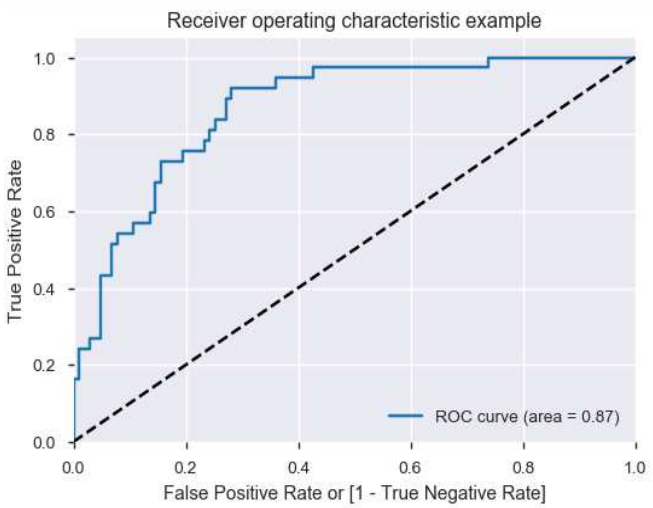


Figure 8 - ROC curve of a classifier shown in slid-line and random guesses show in dash-line

ROC curve is a curve of probability, and through its analysis, we can see four scenarios in a binary classification problem. We can conclude that the true positive rate is 86.8%, the false positive rate is 13.2%, the false negative rate is 9.7%, and the true negative rate is 54.05%. The AUC score is 0.86775, which means it has a good separability measure and is considered acceptable.

F1 score

Precision (equation 10) evaluates how precise the model is and, out of the predicted positives, how many of them are positive. In this case, we can translate the precision to how well the model can predict a customer as default when it is going to default. Recall (equation 11) calculates how many actual positives there are in the sample of the total positives. F1 score (equation 12) is the harmonic mean between the model's precision and recall, and it gives a better measure of incorrectly classified cases than accuracy. Accuracy is the number of correct cases identified on the total sample (equation 13) and is the most used quality assessment method.

$$Precision = \frac{true\ positives}{true\ positives + false\ positives} \quad (19)$$

$$Recall = \frac{true\ positives}{true\ positives + false\ negatives} \quad (20)$$

$$F1\ score = \frac{2 \times precision \times recall}{precision + recall} \quad (21)$$

$$Accuracy = \frac{true\ positives + true\ negatives}{total\ sample} \quad (22)$$

As shown in Table 5, the Logistic Regression model predicts that 103 customers out of 140 of our test data already defaulted once will not default again. This is a very good prediction since we have high values for precision (0.85), recall (0.90), and f1-score (0.87). Besides that, the LR model predicts that 37 customers out of those 140 who already defaulted, will default at least one more time. The values of precision, recall, and f1-score are not so high to determine the number of customers that will default again, but overall, the measures show a good result.

	precision	recall	f1-score	support
0.0	0.85	0.90	0.87	103
1.0	0.67	0.54	0.60	37
micro avg	0.81	0.81	0.81	140
macro avg	0.76	0.72	0.74	140
weighted avg	0.80	0.81	0.80	140

Table 5 - Classification report of Logistic Regression model for Test Data

Since the prediction of a Logistic Regression model is a probability and has the purpose of using it as a classifier, we will have to choose a cut-off value (or threshold value), where scores below this value will be classified as negative, and those above as positive. So, the cut-off should be selected to maximize the profit based on the model predictions of risk. The cut-off value is 0.225256 (Table 6) and the model will make predictions with the new cut-off probability.

	cutoff	falsepositiverate	sensitivity	specificity	total
63	0.225256	0.281553	0.918919	0.718447	1.637366

Table 6 - Classification report with cut-off value of Logistic regression

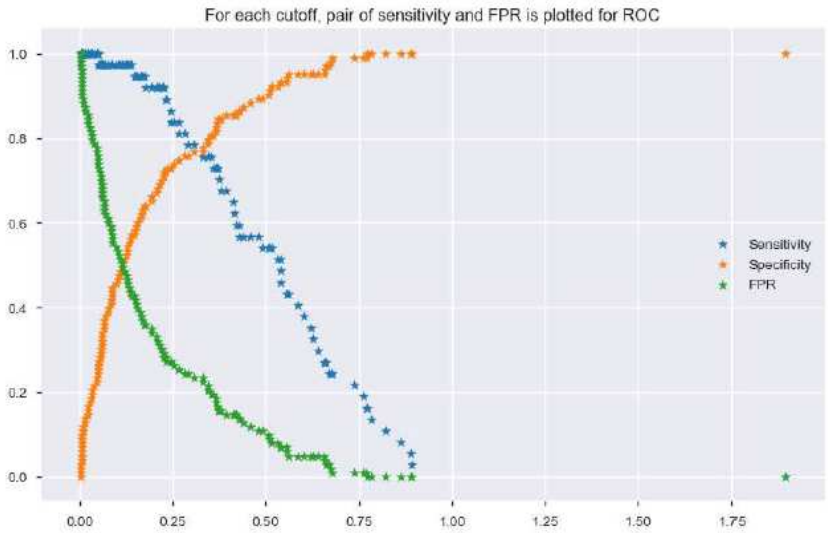


Figure 9 - ROC curve with cut-off value

	precision	recall	f1-score	support
0.0	0.95	0.72	0.82	103
1.0	0.53	0.89	0.67	37
micro avg	0.76	0.76	0.76	140
macro avg	0.74	0.81	0.74	140
weighted avg	0.84	0.76	0.78	140

Table 7 - Classification report of Logistic regression model with cut-off probability for Test Data

Comparing Table 5 with Table 7, we can conclude that the values of Table 7 are higher for the customers who won't default and lower for the ones who will default. In terms of recall, Table 7 shows a recall of 0.89 which means that the model produces lower cases of false negatives and shows a high percentage of true positive cases.

The overall accuracy of the LR model is 76.4% which is acceptable, and it means that the model was trained with success and provides accurate predictions. The greater the value, the better is the performance of our model.

Implementing the Decision Tree model

We move forward to **Decision Tree model** implementation, and the first step is to start building the model using the train data. A decision tree is a greedy algorithm that searches the entire space of possible decision trees, so we need to find optimum parameter(s) or criteria for stopping the decision tree at some point. Hyperparameters are used to prune the decision tree.

Cross Validation methods are defined to find the best combination of hyperparameters for the classification model by training and evaluating the model for each combination of these parameters. This way, a declaration of a hyperparameter is necessary to fine-tune the Decision Tree Classifier. We did a 5-fold cross-validation methodology, and for the best parameters found, the best score of the decision tree model is 0.78392.

The following step is to decide the root node through the Gini index. The feature with the lowest Gini index will be the root node.

To evaluate the model's performance, we will use the test data to see if the DT model was well built, so from now on, the results presented are considered for the test data.

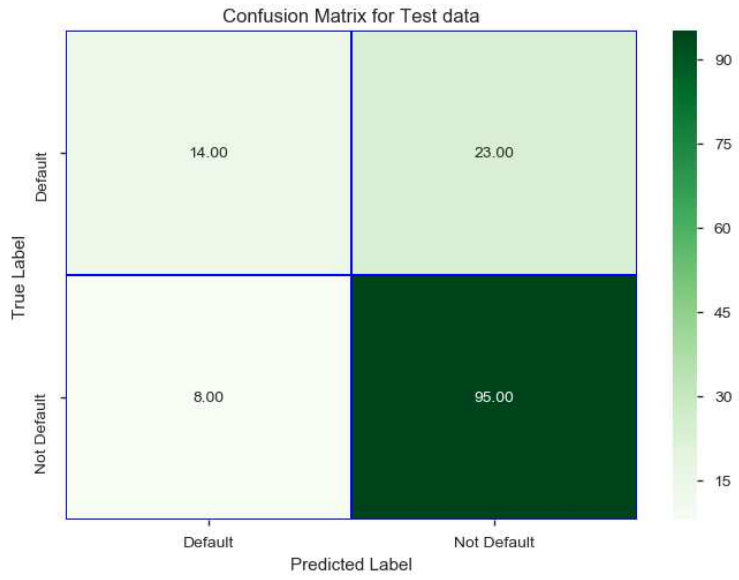


Figure 10 - Confusion Matrix for Test Data

The Confusion matrix for the test dataset observed in Figure 10 shows the number of correctly classified cases and those that were misclassified as the other category. Comparing Figure 10 with Figure 7 (Confusion Matrix for Test Data in LR model), we can conclude that

the LR model presents better performance. The DT model presented 109 cases correctly classified and LR model 113 cases.

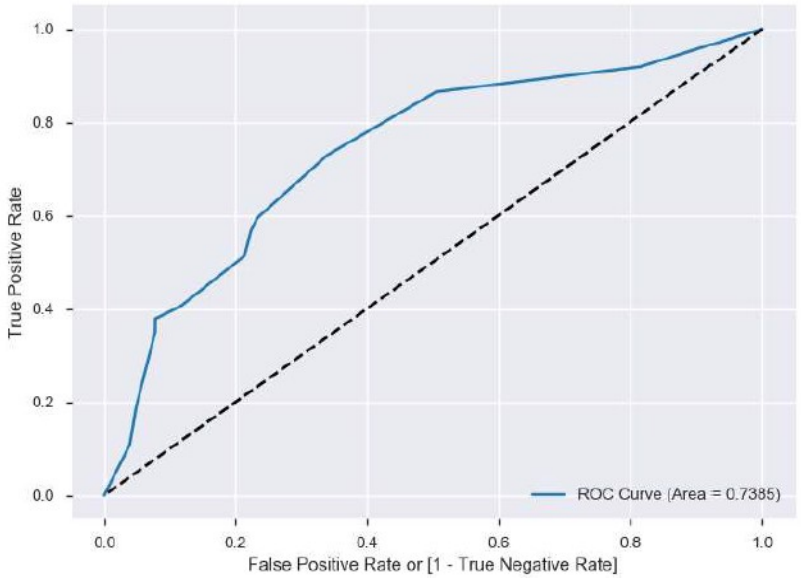


Figure 11 - ROC curve of Decision Tree in slid-line and random guesses show in dash-line

In this case, the overall model could correctly predict whether the customer's credit standing was good or bad with 77.9% accuracy. Our main goal is to reduce the proportion of bad credits. The percent of correct predictions for the bad category and the AUC score is 73.85%.

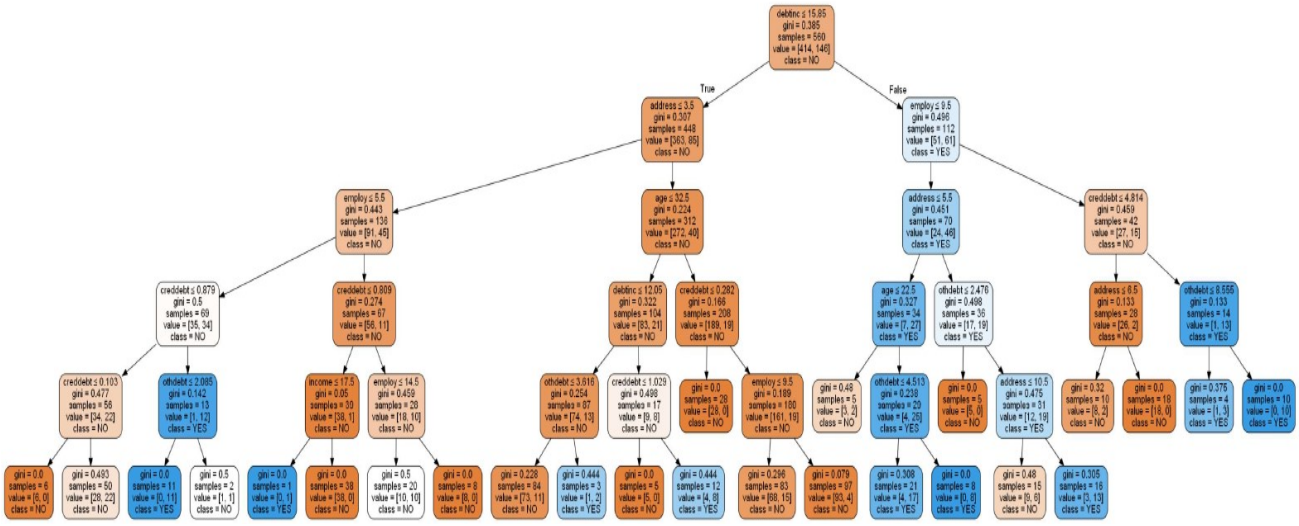


Figure 12 - Decision Tree graph

Each box in the tree in Figure 12 represents a node, the root node is the one on the top and the leaf nodes are the ones at the end of branches and play a special role when the tree is used for prediction. Each node contains information about the number of instances at that node and the distribution of dependent variable values (credit risk). The instances at the root node are the instances in the training set. (Nisbet *et al.*, 2009). In page 61 of Appendix, Figure 12 is resized for a clearer view.

Model selection and business insights

Based on the F1-score (harmonic mean of precision and recall), a LR model with an F1-score (for positive labels – default customers) of 0.67 (Table 7) gives better results than the decision tree with an F1-score of 0.47 (Table 8). Thus, we will use the Logistic Regression model to predict the credit worthiness of the customers.

	precision	recall	f1-score	support
0.0	0.81	0.92	0.86	103
1.0	0.64	0.38	0.47	37
micro avg	0.78	0.78	0.78	140
macro avg	0.72	0.65	0.67	140
weighted avg	0.76	0.78	0.76	140

Table 8 - Classification report of Decision Tree model for Test Data

Thereby, we will predict the probability of default for the remaining 150 customers, with "unknown" data non-considered in the first analysis of this work, using the LR model with a cut-off of 0.225.

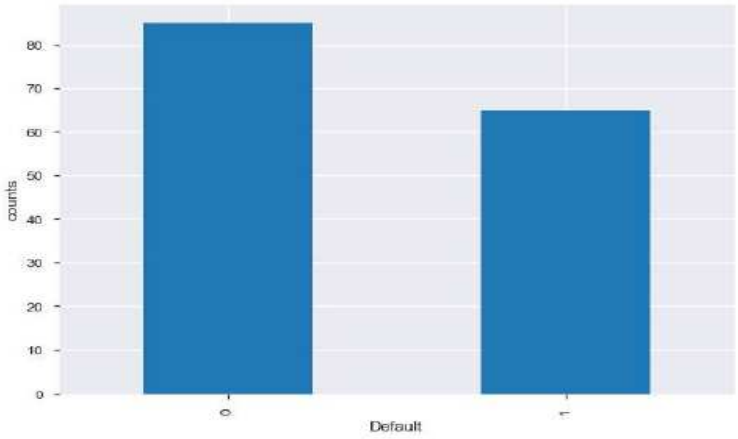


Figure 13 - Model prediction with Logistic Regression for test data

Out of 150 customers, the model predicted that 85 customers would not default on the bank loan, and the remaining 65 customers would most likely default on the loan, as shown in Figure 13.

Model performance validation

To check the model performance, we will use decile analysis for these validations (KS statistics, and Lift and Gain Chart). The Kolmogorov-Smirnov (KS) statistics are used to validate predictive models, compare the maximum difference between the distribution of cumulative events (default) and cumulative non-events (non-default) and check whether the model is capable of making the distinction between events and non-events.

In this case (Table 9), for the training sample, KS score is 50.59% and is at the fourth decile. Ideally, it should be in the first three deciles, and the score between 40 and 70%, since above 70% the data might be overfitted.

Training Sample									
Decile	Defaulters	Non Defaulters	Total	Default RATE	Default PERCENTAGE	CUMU. Default PERCENT	Non Default PERCENT	CUMU. Non Default PERCENT	KS
1	44	12	56	78.57%	30.14%	30.14%	2.90%	2.90%	27.24%
2	31	25	56	55.36%	21.23%	51.37%	6.04%	8.94%	42.43%
3	18	38	56	32.14%	12.33%	63.70%	9.18%	18.12%	45.58%
4	20	36	56	35.71%	13.70%	77.40%	8.70%	26.81%	50.59%
5	12	44	56	21.43%	8.22%	85.62%	10.63%	37.44%	48.18%
6	8	48	56	14.29%	5.48%	91.10%	11.59%	49.03%	42.06%
7	7	49	56	12.50%	4.79%	95.89%	11.84%	60.87%	35.02%
8	4	52	56	7.14%	2.74%	98.63%	12.56%	73.43%	25.20%
9	2	54	56	3.57%	1.37%	100.00%	13.04%	86.47%	13.53%
10	0	56	56	0.00%	0.00%	100.00%	13.53%	100.00%	0.00%
	146	***414***	***560***	26%				KS	50.59%

Table 9 - KS statistic for training data

For test data (Table 10), the KS score is 56.94% and is at the fifth decile. And so, it is a significant value, which makes the predictive model quite acceptable.

Test Sample									
Decile	Defaulters	Non Defaulters	Total	Default RATE	Default PERCENTAGE	CUMU. Default PERCENT	Non Default PERCENT	CUMU. Non Default PERCENT	KS
1	10	4	14	71.43%	27.03%	27.03%	3.88%	3.88%	23.14%
2	10	4	14	71.43%	27.03%	54.05%	3.88%	7.77%	46.29%
3	6	8	14	42.86%	16.22%	70.27%	7.77%	15.53%	54.74%
4	4	10	14	28.57%	10.81%	81.08%	9.71%	25.24%	55.84%
5	4	10	14	28.57%	10.81%	91.89%	9.71%	34.95%	56.94%
6	2	12	14	14.29%	5.41%	97.30%	11.65%	46.60%	50.70%
7	0	14	14	0.00%	0.00%	97.30%	13.59%	60.19%	37.10%
8	0	14	14	0.00%	0.00%	97.30%	13.59%	73.79%	23.51%
9	1	13	14	7.14%	2.70%	100.00%	12.62%	86.41%	13.59%
10	0	14	14	0.00%	0.00%	100.00%	13.59%	100.00%	0.00%
	37	***103***	***140***					KS	56.94%

Table 10 - KS statistic for test data

A lift chart (Figure 14) is derived from the cumulative gains chart and measures how much better we can expect to do with the predictive model than without a model. Lift is the ratio of the gain percentage relative to the expected random result. The dashed-line is the "baseline" curve, i.e., if we select 10% of the cases from the scored dataset, we expect to "gain" approximately 10% of all the cases that will default, which means that there is no gain compared with random. The "Dev Sample" and "Val Sample" curves are modeled with test and training data, respectively, and they represent the expected response using the predictive model.

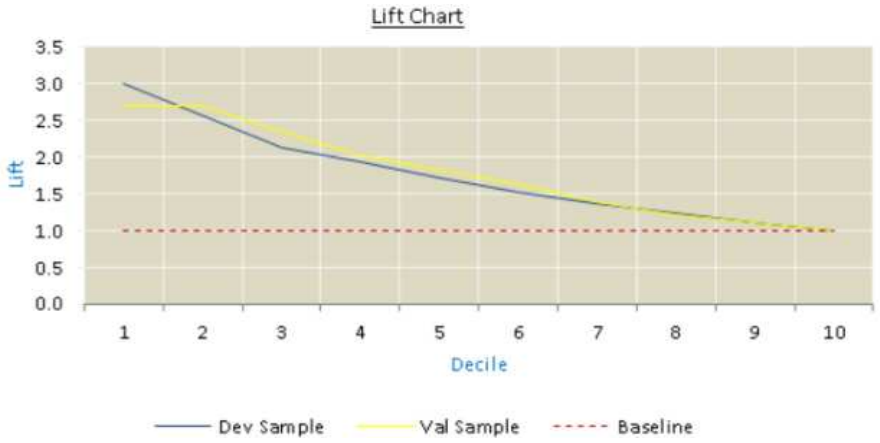


Figure 14 - Lift Chart

In the current model, the cumulative lift for the top two deciles is 2.7, which means that by selecting 20% of the records based on the model, hence we can expect 2.7 times the total number of defaulters to be found than randomly selecting 20% of the data without a model.

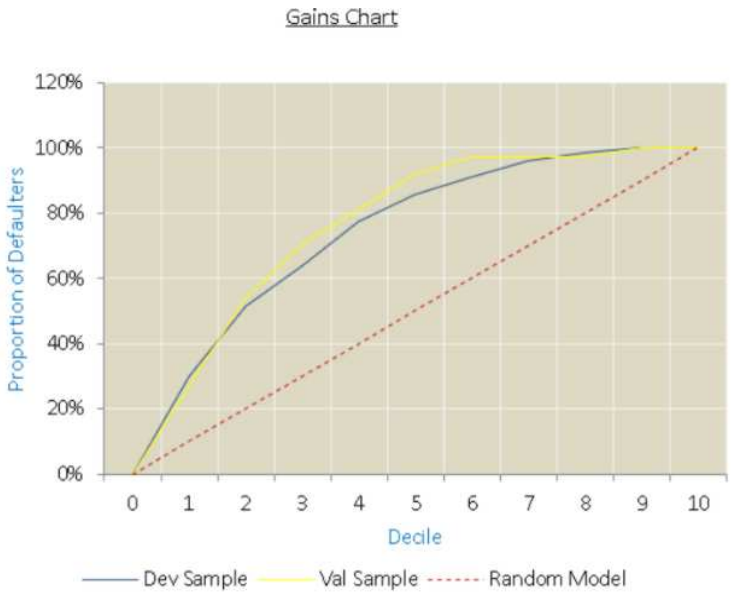


Figure 15 - Gains chart

The gain chart (Figure 15) tells the percentage of targets (events) covered at a given decile level. In the current case, we can say that we can identify 90% of the defaulters who are likely to default on the loan by analysing just 50% of the total customers.

The bigger the distance between "Random Model" and "Dev Sample", the bigger the company gains from using the predictive model to find the defaulters, rather than merely making a random selection.

To summarize, gain and lift charts help to understand how good the predictive model is and how the response rate of a targeted selection is compared to a random selection. Also, the gain and lift chart can determine the point at which the model's predictions become less useful.

The intended gain depends on the cost of Type I and Type II errors and the business objective. If the bank's priority is the bad debt, then the Type I error must be lowered, i.e., avoid classifying a defaulter as a non-defaulter. On the cumulative gains chart, this could match rejecting loans to applicants of 40% of the total customers, detaining approximately 80% of the possible defaulters; however, the applicant pool is practically halved. If the bank's primary concern is to increase the customer base, it must avert to classifying a non-defaulter as a defaulter (Type II error). In Figure 15, this is associated with rejecting the top 10%, which captures 30% of the defaulters and leaves almost the applicant pool intact.

CHAPTER 6

Conclusion

A lender usually faces two types of decisions: to provide credit to a new customer, and how to deal with existing customers, especially with increasing their credit limits. In both cases, whatever the algorithms used, it is critical to have a large sample of previous customers with their application details, behavioral patterns, and subsequent credit history available. Most of the methods are then used on this sample to identify the connection between the characteristics of the consumers (as annual income, age, number of years in employment with their current employer, etc.) and their financial history.

This study has been both data-focused and method-focused. Data-focused in the sense that prediction models were based only on loan data and method-focused in applying the Machine Learning models to predict if the customers will go into default or non-default on loans application.

Concerning the dataset and empirical analysis, we started by applying pre-processing data techniques to remove missing values, outliers, or unknown data. Likewise, we performed univariate and bivariate analysis to comprehend the relationship between the variables (correlation, VIF and T-test), check if the data was biased, and focus on understanding the importance of each variable in the model. After this data preparation, we moved forward to model building and performance check of both ML models. Additionally, we have shown that it is essential to consider different Machine Learning models for the data sample we are analysing and the predictions needed regarding to the business problem. There is an apparent confirmation looking for the results and exploring the different metrics like AUC, precision, recall, F1 score, and, accuracy, which we obtained with the Logistic Regression model and Decision Trees model. We have seen that Logistic Regression, compared with the Decision Tree, had a better performance with an accuracy rate of 76.4% with missing observations omitted. This way, the LR model was chosen to predict the probability of default for the sample data, even for the customers who had the default variable in blank. In general, LR model obtained a satisfactory performance. Although it is difficult to say if there is a better Machine Learning model than others, this depends on the data you use and the purpose of the business problem in study.

It is also important to emphasize that this work can be largely improved. Other Data Mining algorithms such as: Random Forest, K-Nearest Neighbor, Support Vector Machine, and Artificial Neural Network can be implemented to this dataset (or to the new added variables datasets) in future research as we only compared results and performance between Logistic Regression and Decision Tree model.

Furthermore, the present work solely contemplated the dataset variables, and thereby new features can be included with more details about each customer's economic, social and financial status. Thus, in the field of economic features, we could add information about *unemployment rate*, *interest rate*, *inflation rate*, *effort rate*, *rent ratio*, and *heritage*. Social features that can improve the model would be *number of family members*, *birth rate*, *divorce rate*, *marriage rate*, *health levels* and *lifestyle*. We know there are numerous credit applications and various types of credit, such as: credit cards, mortgages, home equity loans, mail catalog orders, auto loans, and a wide variety of personal loan products. This information would allow us to understand better which type of credit the client is most likely to default as diverse credits have different thresholds and allowed amounts to the borrower. So, considering financial features, we could add to the model "consumer leverage ratio", "mortgage percentage of income", "auto loans percentage rate", "personal loan percentage rate", and "credit card percentage rate". On the other hand, we could include additional measures related to credit scoring, like *Weight of Evidence (WOE)*- the predictive power of independent variables concerning dependent variables - and *Information Value (IV)* - which selects essential variables in a predictive model. These measures would reduce missing values or outliers and select variables, allowing for strong grouped characteristics by representing independent information types.

Moreover, if a credit scorecard model is built only considering the applications accepted and not considering the applications rejected in the past, or if the default status is unknown, then the data can be biased. Thus, we could use *Reject Inference* methods, which are used to improve the quality of credit scorecards by incorporating data from rejected loan applications.

Extending the work on this topic, the *Survival Analysis* refers to time-to-event data when modeling a particular event of interest is the main objective of the problem, in our case, when the default will occur. This is an exciting area to explore, as the banking industry will be able to forecast the probability of default, using only limited event occurrence information at the early stage (Wang *et al.*, 2019).

References

Alpaydin, E. (2020). *Introduction to machine learning*. MIT press, 3-18.

Ayodele, T.O. (2010). Types of Machine Learning Algorithms. *New Advances in Machine Learning, 3*, 19-24.

Bank for International Settlements. A brief history of the Basel Committee. Retrieved February 17, 2020, from https://www.bis.org/bcbs/history.htm.

BBVA. Credit risk: Methodologies for credit risk quantification. Retrieved March 9, 2020, from https://shareholdersandinvestors.bbva.com/microsites/FinancialReport2011/en/Riskmanagement/CreditriskMethodologiesforcreditriskquantification.html.

BCBS. (2005). Studies on the validation of Internal Rating Systems. Working Paper No. 14, BIS.

BCBS. (2006). International convergence of capital measurements and capital standards: A revised framework comprehensive version.

Bengio, Y., Lodi, A., & Prouvost, A. (2021). Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, *290*(2), 405-421.

Bishop, C. M. (2006). *Pattern recognition and machine learning*, New York: springer, 137-281.

Bose, I., & Mahapatra, R.K. (2001). Business data mining – a machine learning perspective. *Information & Management*, *39*(3), 211-225.

Bradley, A. P. (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern recognition, 30*(7), 1145-1159.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (2017). *Classification and regression trees*. Routledge, 10-60.

Breiman, L. (2001). Random Forests. *Machine Learning, 45*(1), 5-32.

Char, D. S., Shah, N. H., & Magnus, D. (2018). Implementing machine learning in health care—addressing ethical challenges. *The New England journal of medicine, 378*(11), 981.

Chye, K. H., Chin, T. W., & Peng, G. C. (2004). Credit scoring using data mining techniques. *Singapore Management Review, 26*(2), 25-48.

Cielen, D., Meysman, A.D.B. and Mohamed, A. (2016). *Introducing Data Science: Big Data, Machine Learning, And More, Using Python Tools,* 58-59.

Cios, K.J., Pedrycz, W., Swiniarski, R.W., & Kurgan, L.A. (2007). *Data mining: a knowledge discovery approach*, 391–393.

Fawcett, T. (2004). *ROC graphs: Notes and practical considerations for researchers*. Machine learning, 31(1), 1-38.

Fayyad, U, Piatetsky-Shapiro, G., & Smyth, P. (1996). From data mining to knowledge discovery databases. *AI Magazine , 17*(3), 37-37.

Fleming, P. (2019). Robots and organization studies: Why robots might not want to steal your job. *Organization Studies, 40(1),* 23-38.

Gestel, T.V., & Baesens, B. (2009). *Credit Risk Management: Basic Concepts: Financial Risk Components, Rating Analysis, Models, Economic and Regulatory Capital*. Oxford University Press, 1-59.

Goodhart, C. (2011). *The Basel Committee on Banking Supervision: A history of the early years 1974–1997*. Cambridge University Press, 1-10.

Harrington, P. (2012). *Machine Learning in Action.* Simon and Schuster, 83-128.

Hebb, D. (1949). *The Organization of Behavior.* John Wiley & Sons, 107-140.

Holzinger, A., Kieseberg, P., Weippl, E., & Tjoa, A. M. (2018). Current advances, trends and challenges of machine learning and knowledge extraction: from machine learning to explainable AI. *In International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, 1-8.

Hosmer Jr, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). *Applied logistic regression* (Vol. 398). John Wiley & Sons, 8-31.

Kashyap, A. K., & Stein, J. C. (2004). Cyclical implications of the Basel II capital standards. *Economic Perspectives-Federal Reserve Bank Of Chicago, 28(1)*, 18-33.

King, P., & Tarbert, H. (2011). Basel III: an overview. *Banking & Financial Services Policy Report, 30(5)*, 1-18.

Langley, P., & Simon, H. A. (1995). Applications of machine learning and rule induction. *Communications of the ACM*, *38*(11), 54-64.

Mahesh, B. (2020). Machine Learning Algorithms - A Review in *International Journal of Science and Research (IJSR), 9*(1).

Mahjoobi, J., & Etemad-Shahidi, A. (2008). An alternative approach for the prediction of significant wave heights based on classification and regression trees. *Applied Ocean Research, 30*(3), 172-177.

Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., & Galstyan, A. (2021). A survey on bias and fairness in machine learning. *ACM Computing Surveys (CSUR), 54*(6), 1-35.

Mendes, A. S. R. (2013). *Transição de Basileia II Para o Basileia III "Qual o Enfoque Que é Dado aos Riscos nos Acordos de Basileia?"* (Doctoral dissertation, Universidade de Lisboa (Portugal)).

Merton, R. (1974). On the pricing of corporate debt: The risk structure of interest rates, *The Journal of Finance 29 (2),* 449-470.

Mester, L. (1997). What's the Point of Credit Scoring?. *Federal Reserve Bank of Philadelphia Business Review*, 3-16.

Moosa, I. (2008). *Quantification of Operational Risk Under Basel II: The Good, The Bad, and The Ugly*. Springer, 1-65.

Nisbet, R., Elder, J., & Miner, G. D. (2009). *Handbook of statistical analysis and data mining applications*. Academic press, 459-470.

Novaković, J. D., Veljović, A., Ilić, S. S., Papić, Ž., & Milica, T. (2017). *Evaluation of classification models in machine learning*. Theory and Applications of Mathematics & Computer Science, 7(1), 39-46.

Penikas, H. (2015). History of banking regulation as developed by the Basel Committee on Banking Supervision 1974-2014. *Estabilidad financiera. Nº 28 (mayo 2015)*, 9-47.

Qiu, J., Wu, Q., Ding, G., Xu, Y., & Feng, S. (2016). A survey of machine learning for big data processing. *EURASIP Journal on Advances in Signal Processing, 2016(1),* 1-16.

Quinlan, J. R. (2014). *C4. 5: programs for machine learning.* Elsevier, 17-80.

Rokach, L. & Maimon, O. (2014). *Data mining with decision trees: theory and applications*, (vol. 81). World Scientific.

Ross, S. (2019). *What Factors are Taken Into Account to Quantify Credit Risk?* Retrieved August 18, 2021, from https://www.investopedia.com/ask/answers/022415/what-factors-are-taken-account-quantify-credit-risk.asp

Sathya, R. & Abraham, A. (2013). Comparison of Supervised and Unsupervised Learning Algorithms for Pattern Classification in *International Journal of Advanced Research in Artificial Intelligence*, *2*(2).

Sethi, A. (2020). *Supervised Learning vs. Unsupervised Learning – A Quick Guide for Beginners*. Retrieved August 18, 2021, from https://www.analyticsvidhya.com/blog/2020/04/supervised-learning-unsupervised-learning/

Shalev-Shwartz, S. & Ben-David, S. (2014). *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press. 1-16.

Tharwat, A. (2021). Classification assessment methods. *Applied Computing and Informatics*. Vol. 17 No. 1. 168-192.

Wang, P., Li, Y., & Reddy, C. K. (2019). Machine learning for survival analysis: A survey. *ACM Computing Surveys (CSUR), 51(6),* 1-36.

Yang, M., Rosenhahn, B., & Murino, V. (Eds.). (2019). *Multimodal scene understanding: Algorithms, applications and deep learning.* Academic Press. 65-100.

Appendix

Import libraries

In [100]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats as stats
%matplotlib inline
from pandas_profiling import ProfileReport
```

In [101]:
```
import statsmodels.formula.api
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics

from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import cross_validate
```

Load datasets

In [102]:
```
data = pd.read_csv('C:/Users/costa/Downloads/Credit-Risk-Analytics-master/Credit-Risk-Analytics-master/bankloans.csv')
```

In [103]:
```
data.head()
```

Out[103]:

	age	ed	employ	address	income	debtinc	creddebt	othdebt	default
0	41	3	17	12	176	9.3	11.359392	5.008608	1.0
1	27	1	10	6	31	17.3	1.362202	4.000798	0.0
2	40	1	15	14	55	5.5	0.856075	2.168925	0.0
3	41	1	15	14	120	2.9	2.658720	0.821280	0.0
4	24	2	2	0	28	17.3	1.787436	3.056564	1.0

In [104]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 850 entries, 0 to 849
Data columns (total 9 columns):
age         850 non-null int64
ed          850 non-null int64
employ      850 non-null int64
address     850 non-null int64
income      850 non-null int64
debtinc     850 non-null float64
creddebt    850 non-null float64
othdebt     850 non-null float64
default     700 non-null float64
dtypes: float64(4), int64(5)
memory usage: 59.8 KB
```

In [105]:
```
data.isnull().sum()
```

Out[105]:
```
age           0
ed            0
employ        0
address       0
income        0
debtinc       0
creddebt      0
othdebt       0
default     150
dtype: int64
```

In [106]:
```
#splitting the data set into two sets
data_exist = data.loc[data.default.isnull()==0]
data_new = data.loc[data.default.isnull()==1]
```

In [107]:
```
data_exist.describe().T
```

Out[107]:

	count	mean	std	min	25%	50%	75%	max
age	700.0	34.860000	7.997342	20.000000	29.000000	34.000000	40.000000	56.00000
ed	700.0	1.722857	0.928206	1.000000	1.000000	1.000000	2.000000	5.00000
employ	700.0	8.388571	6.658039	0.000000	3.000000	7.000000	12.000000	31.00000
address	700.0	8.278571	6.824877	0.000000	3.000000	7.000000	12.000000	34.00000
income	700.0	45.601429	36.814226	14.000000	24.000000	34.000000	55.000000	446.00000
debtinc	700.0	10.260571	6.827234	0.400000	5.000000	8.600000	14.125000	41.30000
creddebt	700.0	1.553553	2.117197	0.011696	0.369059	0.854869	1.901955	20.56131
othdebt	700.0	3.058209	3.287555	0.045584	1.044178	1.987567	3.923065	27.03360
default	700.0	0.261429	0.439727	0.000000	0.000000	0.000000	1.000000	1.00000

In [108]:
```
ProfileReport(data)
```

Checking for outliers

In [109]:
```
def outlier_capping(x):
    x= x.clip(upper=x.quantile (0.95))
    x= x.clip(lower=x.quantile(0.01))
    return(x)
```

In [110]:
```
#outlier treatement
data_exist = data_exist.apply(lambda x: outlier_capping(x))
```

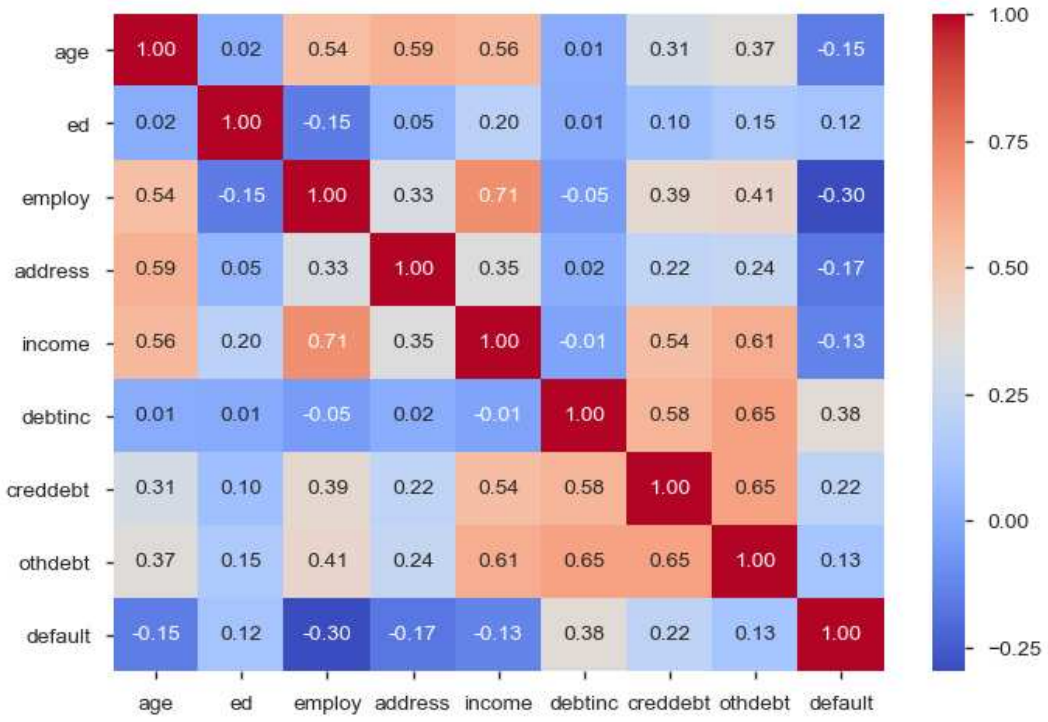
Correlation Matrix

In [111]:
```
data_exist.corr()
```

Out[111]:

	age	ed	employ	address	income	debtinc	creddebt	othdebt	default
age	1.000000	0.015431	0.538676	0.592088	0.559657	0.013812	0.313417	0.365274	-0.148476
ed	0.015431	1.000000	-0.154969	0.053037	0.203289	0.010386	0.099394	0.151947	0.118521
employ	0.538676	-0.154969	1.000000	0.326449	0.708981	-0.047798	0.392843	0.414215	-0.297359
address	0.592088	0.053037	0.326449	1.000000	0.352795	0.018753	0.222177	0.243753	-0.167796
income	0.559657	0.203289	0.708981	0.352795	1.000000	-0.013813	0.541225	0.609539	-0.125221
debtinc	0.013812	0.010386	-0.047798	0.018753	-0.013813	1.000000	0.575882	0.645411	0.381212
creddebt	0.313417	0.099394	0.392843	0.222177	0.541225	0.575882	1.000000	0.645723	0.217870
othdebt	0.365274	0.151947	0.414215	0.243753	0.609539	0.645411	0.645723	1.000000	0.125296
default	-0.148476	0.118521	-0.297359	-0.167796	-0.125221	0.381212	0.217870	0.125296	1.000000

In [112]:
```
#visualize the correlation using seaborn heatmap
sns.heatmap(data_exist.corr(),annot= True,fmt= '0.2f',cmap='coolwarm')
plt.show()
```



In [113]:
```
print(data_exist.shape)
print(data_new.shape)
```

```
(700, 9)
(150, 9)
```

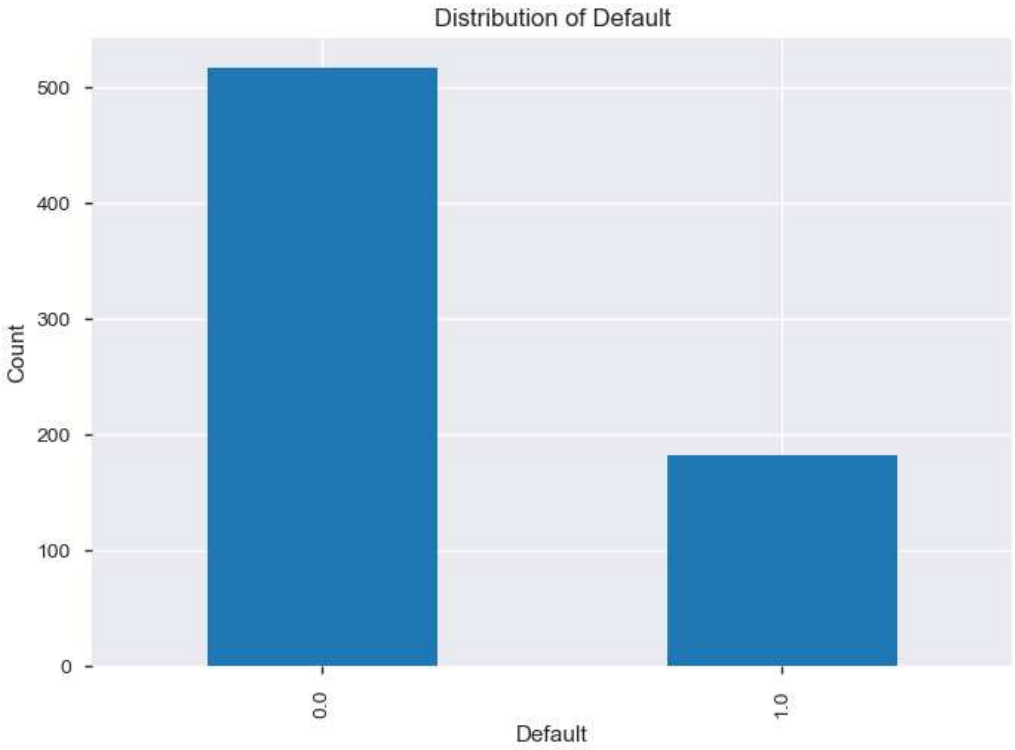
In [114]:
```
#Frequency Distribution of Default
print(data_exist.default.value_counts())
print(round(data_exist.default.value_counts()/data_exist.shape[0]* 100,2))
```

```
0.0    517
1.0    183
Name: default, dtype: int64

0.0    73.86
1.0    26.14
Name: default, dtype: float64
```

In [115]:
```
data_exist.default.value_counts().plot.bar()
plt.xlabel('Default')
plt.ylabel('Count')
plt.title('Distribution of Default')
```

Out[115]: Text(0.5,1,'Distribution of Default')



Data Exploratory analysis

- Univariate Analysis - Numeric(TTest)/Categorical(chisquare)

In [116]:
```
numeric_var_names = data_exist.columns
numeric_var_names=numeric_var_names.drop('default')
```

Uni-Variate and Bi-Variate Analysis

In [117]:
```
from matplotlib.backends.backend_pdf import PdfPages
```

In [118]:
```
### Bivariate boxplot
box_plot = PdfPages('BoxPlots with Default Split.pdf')

for num_variable in numeric_var_names:
    fig,axes = plt.subplots(figsize=(10,4))
    sns.boxplot(x='default',y=num_variable,data = data_exist)
    plt.title(str('Box Plot of ') + str(num_variable))
    box_plot.savefig(fig)
box_plot.close()
### Uivariate boxplot
box_plot= PdfPages('Boxplot with Total View.pdf')

for num_variable in numeric_var_names:
    fig,axes = plt.subplots(figsize=(10,4))
    sns.boxplot(y=num_variable,data= data_exist)
    plt.title(str('Boxplot of ')+ str(num_variable))
    box_plot.savefig(fig)
box_plot.close()
```

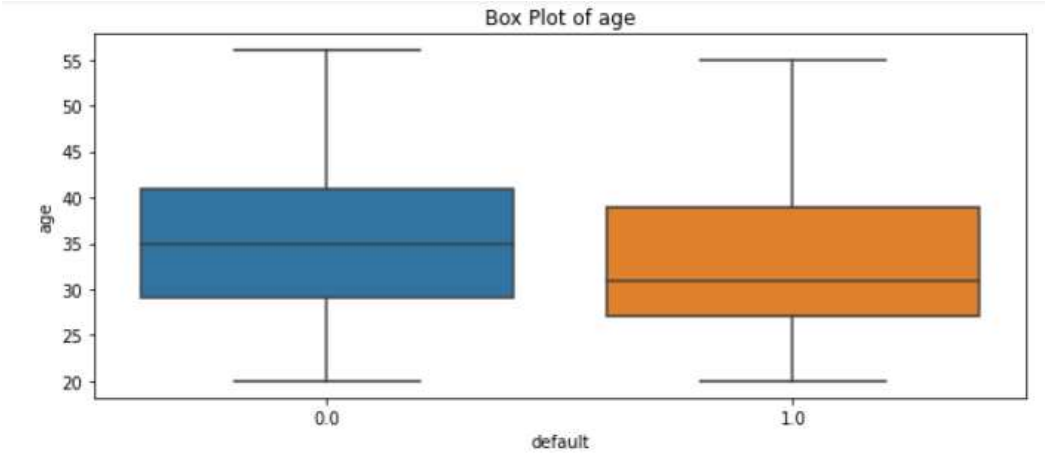


Figure 16 - Box Plot of age with default

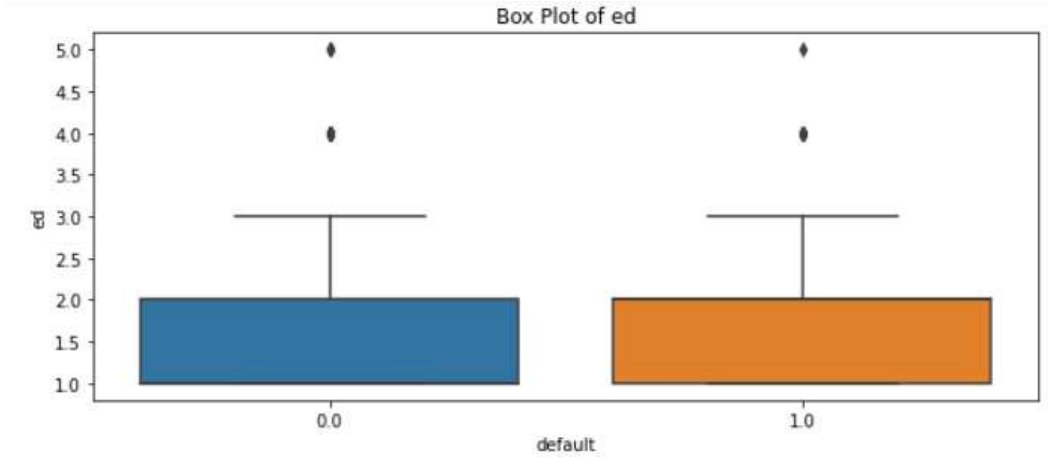


Figure 17 - Box Plot of education with default

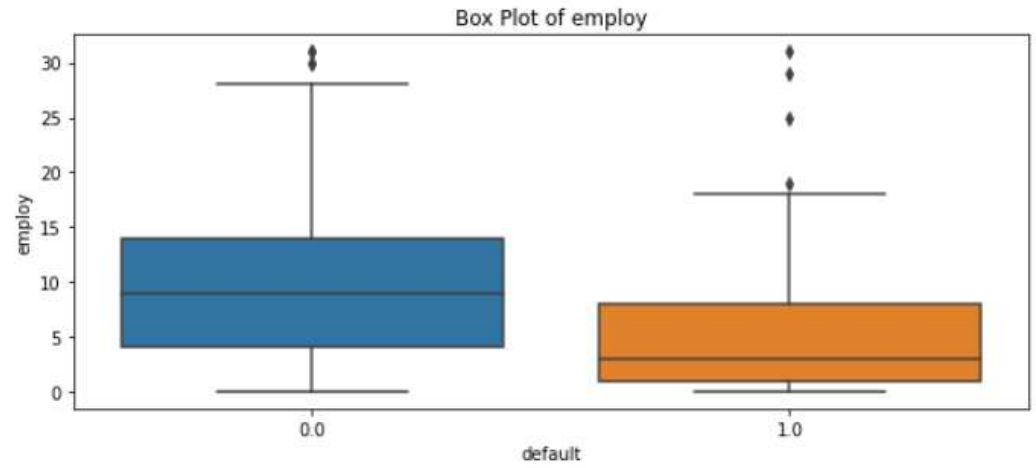


Figure 18 - Box Plot of employ with default

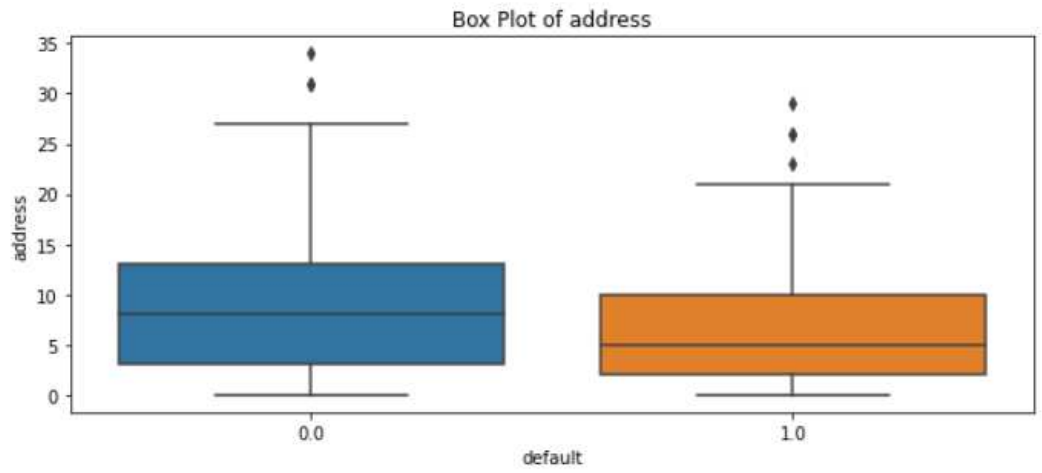


Figure 19 - Box Plot of address with default

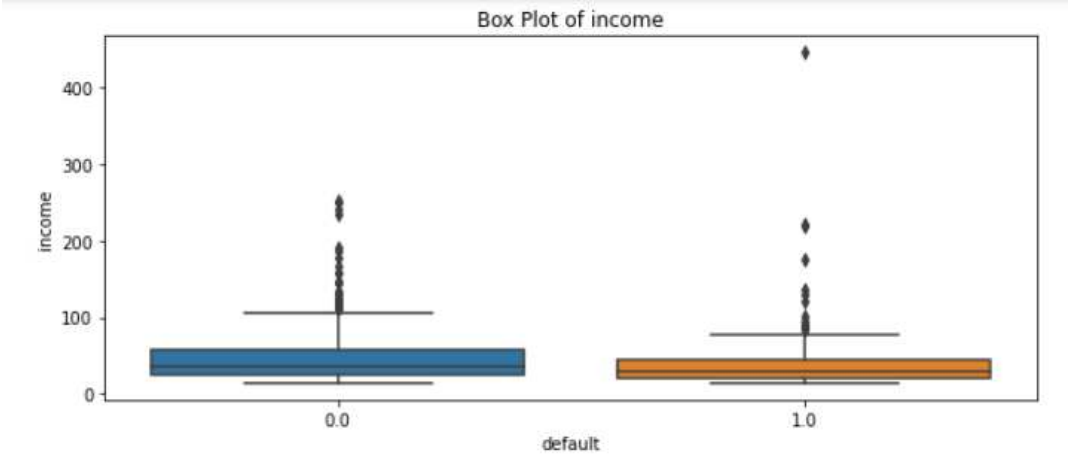


Figure 20 - Box Plot of income with default

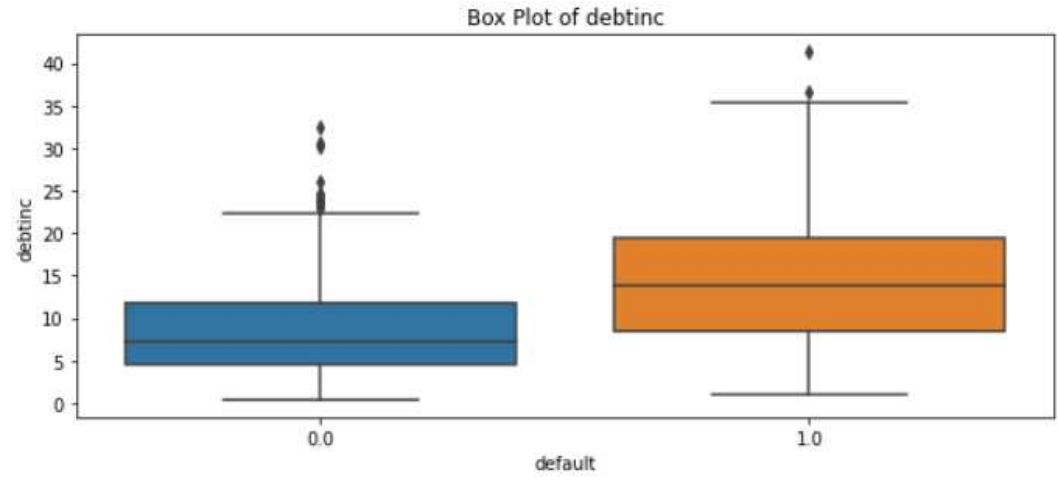


Figure 21 - Box Plot of debtinc with default

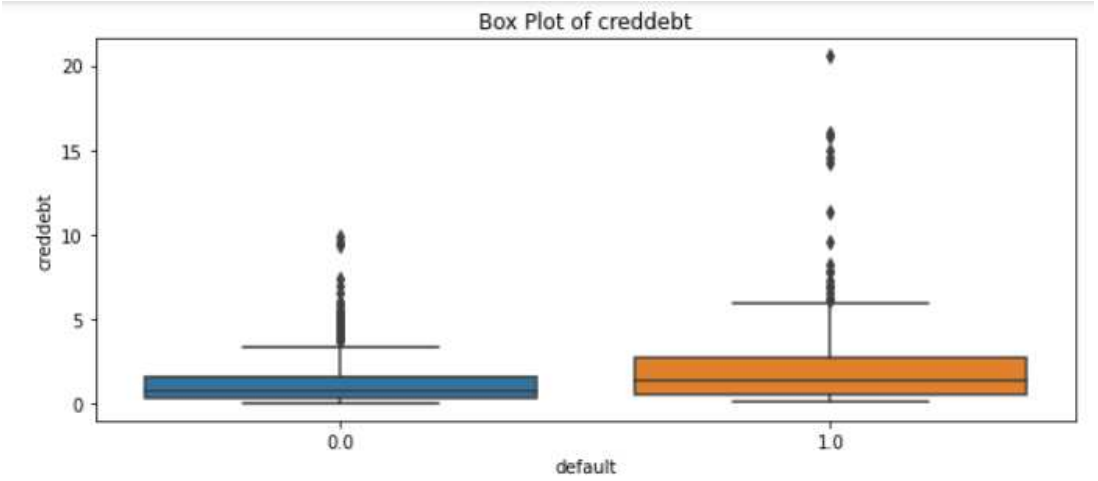


Figure 22 - Box Plot of creddebt with default

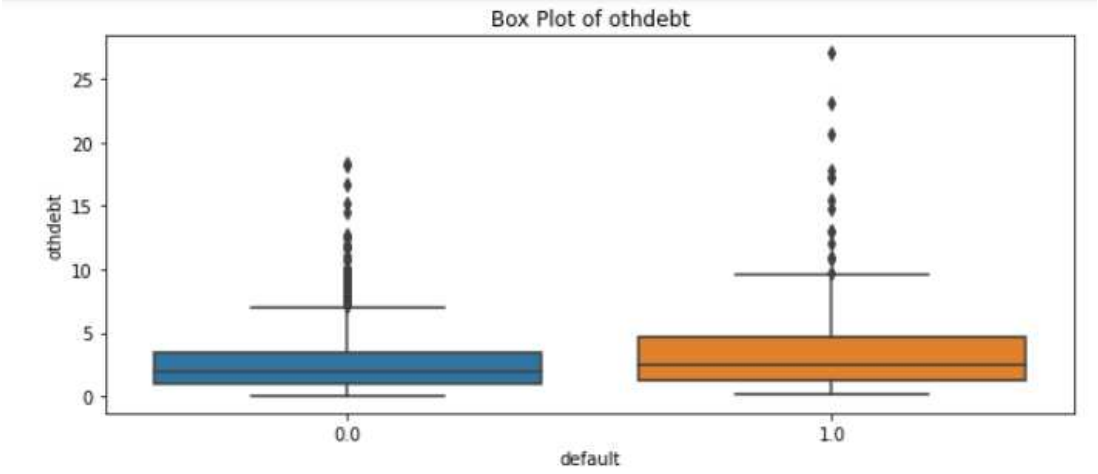


Figure 23 - Box Plot of othdebt with default

In [119]:
```
tstats_df = pd.DataFrame()
for num_variable in numeric_var_names:
    tstats = stats.ttest_ind(data_exist[data_exist['default']==1][num_variable],data_exist[data_exist['default']==0][num_variable
    temp = pd.DataFrame([num_variable, tstats[0], tstats[1]]).T
    temp.columns = ['Variable Name', 'T-Statistic', 'P-Value']
    tstats_df = pd.concat([tstats_df, temp], axis=0, ignore_index=True)

print(tstats_df)
```

```
  Variable Name T-Statistic      P-Value
0           age    -3.96667  8.04114e-05
1            ed     3.15351   0.00168236
2       employ    -8.22834  9.29726e-16
3       address    -4.49688  8.07382e-06
4        income    -3.33454  0.000899569
5       debtinc     10.8941  1.24191e-25
6      creddebt     5.89774  5.74188e-09
7       othdebt     3.33658  0.000893111
```

Multi Collinearity Check

In [120]:
```
from patsy import dmatrices
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

In [121]:
```
fetures = '+'.join(data_exist.columns.difference(['default']))
fetures
```

Out[121]:
```
'address+age+creddebt+debtinc+ed+employ+income+othdebt'
```

In [122]:
```
## vif(variance inflation factor)
x,y = dmatrices(formula_like='default ~' + fetures,data=data_exist,return_type = 'dataframe')
vif = pd.DataFrame()
vif['VIF Factor'] = [variance_inflation_factor(y.values,i)for i in range(y.shape[1])]
vif['Features'] = y.columns
vif
```

Out[122]:

	VIF Factor	Features
0	41.582718	Intercept
1	1.549126	address
2	2.068488	age
3	2.923224	creddebt
4	5.029469	debtinc
5	1.292434	ed
6	2.623756	employ
7	5.888177	income
8	5.322683	othdebt

From VIF check ,we found out that the correlation between the variables is within the acceptable limit.

Implementing the model

Logistic Regression

In [123]:
```
f_col =data_exist.columns.difference(['default'])
f_col
```

Out[123]:
```
Index([u'address', u'age', u'creddebt', u'debtinc', u'ed', u'employ',
       u'income', u'othdebt'],
      dtype='object')
```

In [124]:
```
X_train,X_test,y_train,y_test = train_test_split(data_exist[f_col],data_exist.default,test_size=0.2,stratify = data_exist.defaul
print(X_train.shape)
print(X_test.shape)
```

```
(560, 8)
(140, 8)
```

In [125]:
```
### Building Model
log_reg = LogisticRegression()
log_reg.fit(X_train,y_train)
```

Out[125]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=None, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
```

In [126]:
```
### Coefficients
coff = pd.DataFrame({'Features':pd.Series(f_col),'coefficient': pd.Series(log_reg.coef_[0])})
coff
```

Out[126]:

	Features	coefficient
0	address	-0.102234
1	age	0.013289
2	creddebt	0.619944
3	debtinc	0.072219
4	ed	0.086084
5	employ	-0.216351
6	income	0.000919
7	othdebt	0.052365

In [127]:
```
log_reg.intercept_
```

Out[127]:
```
array([-1.33205279])
```

In [128]:
```
### Predicting test case
data_test_pred = pd.DataFrame({'actual':y_test,'predicted': log_reg.predict(X_test)})
data_test_pred = data_test_pred.reset_index()
data_test_pred.head()
```

Out[128]:

	index	actual	predicted
0	145	0.0	0.0
1	526	0.0	0.0
2	689	0.0	0.0
3	430	1.0	1.0
4	245	0.0	0.0

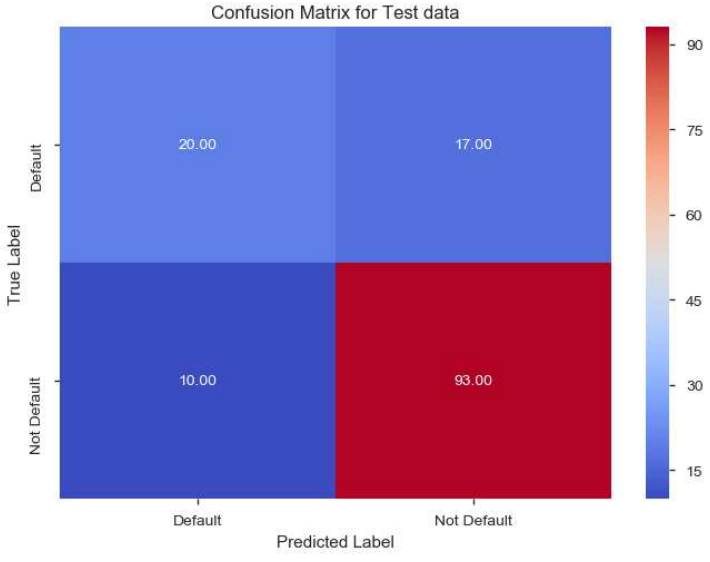
In [129]:
```
### Confusion Matrix

cm_lg = metrics.confusion_matrix(data_test_pred.actual,data_test_pred.predicted,labels = [1,0])
cm_lg
```

Out[129]:
```
array([[20, 17],
       [10, 93]], dtype=int64)
```

In [130]:
```
### visualizing confusion matrix

sns.heatmap(cm_lg,annot=True, fmt=".2f", cmap="coolwarm",
            xticklabels = ["Default", "Not Default"] , yticklabels = ["Default", "Not Default"])
plt.title("Confusion Matrix for Test data")
plt.ylabel("True Label")
plt.xlabel("Predicted Label")
plt.show()
```



In [131]:
```
## Precision Score
precision = metrics.precision_score(data_test_pred.actual,data_test_pred.predicted)
print('Precision score of model : ', round(precision,3))
## Accuracy of model
accuracy = metrics.accuracy_score(data_test_pred.actual,data_test_pred.predicted)
print('Accuracy of model :',round(accuracy,3))
```

```
('Precision score of model : ', 0.667)
('Accuracy of model :', 0.807)
```

In [132]:
```
predict_prob = pd.DataFrame(log_reg.predict_proba(X_test))
```

In [133]:
```
predict_prob.columns = ['default_0','default_1']
predict_prob.head()
```

Out[133]:

	default_0	default_1
0	0.807247	0.192753
1	0.940932	0.059068
2	0.710545	0.289455
3	0.178746	0.821254
4	0.633379	0.366621

In [134]:
```
data_test_pred = pd.concat([data_test_pred,predict_prob],axis=1)
data_test_pred.head()
```

Out[134]:

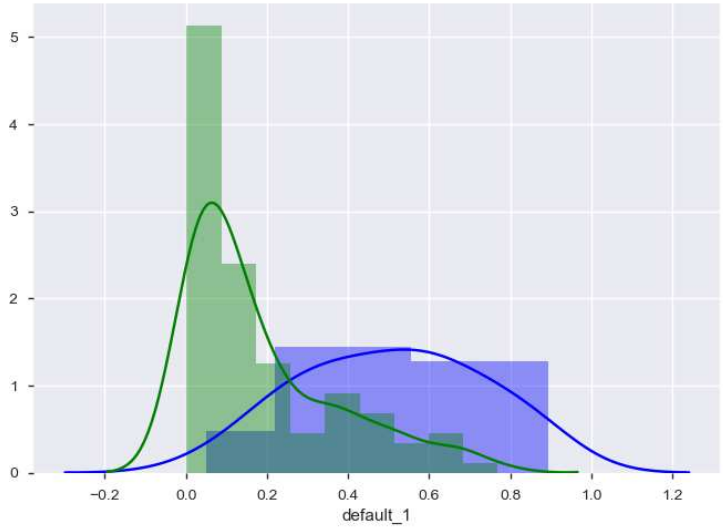
	index	actual	predicted	default_0	default_1
0	145	0.0	0.0	0.807247	0.192753
1	526	0.0	0.0	0.940932	0.059068
2	689	0.0	0.0	0.710545	0.289455
3	430	1.0	1.0	0.178746	0.821254
4	245	0.0	0.0	0.633379	0.366621

In [135]:
```
### AUC Score
auc= metrics.roc_auc_score(data_test_pred.actual,data_test_pred['default_1'])
auc
```

Out[135]:
```
0.8677512463920231
```

In [136]:
```
sns.distplot(data_test_pred[data_test_pred.actual ==1]['default_1'],color = 'b')
sns.distplot(data_test_pred[data_test_pred.actual ==0]['default_1'],color = 'g')
```

Out[136]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x153834a8>
```



In [137]:
```
### Classification Report
print(metrics.classification_report(data_test_pred.actual,data_test_pred.predicted))
```

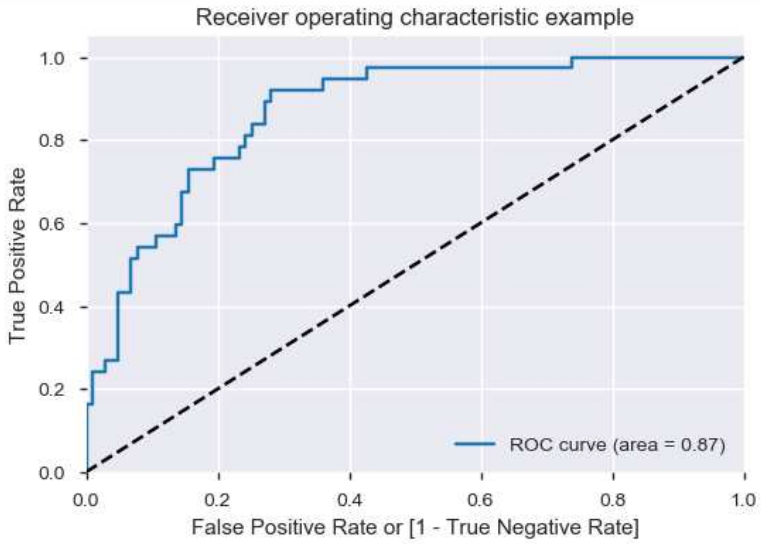
```
              precision    recall  f1-score   support

         0.0       0.85      0.90      0.87       103
         1.0       0.67      0.54      0.60        37

   micro avg       0.81      0.81      0.81       140
   macro avg       0.76      0.72      0.74       140
weighted avg       0.80      0.81      0.80       140
```

In [138]:
```
fpr, tpr, thresholds = metrics.roc_curve( data_test_pred.actual,
                                          data_test_pred.default_1,
                                          drop_intermediate = False )

plt.figure(figsize=(6, 4))
plt.plot( fpr, tpr, label='ROC curve (area = %0.2f)' % auc )
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate or [1 - True Negative Rate]')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic example')
plt.legend(loc="lower right")
plt.show()
```



In [139]:
```
print(thresholds[0:10])
print(fpr[0:10])
print(tpr[0:10])
```

```
[1.89293561 0.89293561 0.88996234 0.86223052 0.82125362 0.78296459
 0.77333193 0.76944991 0.7609111  0.73886713]
[0.         0.         0.         0.         0.         0.
 0.         0.00970874 0.00970874 0.00970874]
[0.         0.02702703 0.05405405 0.08108108 0.10810811 0.13513514
 0.16216216 0.16216216 0.18918919 0.21621622]
```

In [140]:
```
i = np.arange(len(tpr))

roc_like_df = pd.DataFrame({'falsepositiverate' : pd.Series(fpr, index=i),'sensitivity' : pd.Series(tpr, index = i),
              'specificity' : pd.Series(1-fpr, index = i),'cutoff' : pd.Series(thresholds, index = i)})
roc_like_df['total'] = roc_like_df['sensitivity'] + roc_like_df['specificity']
```

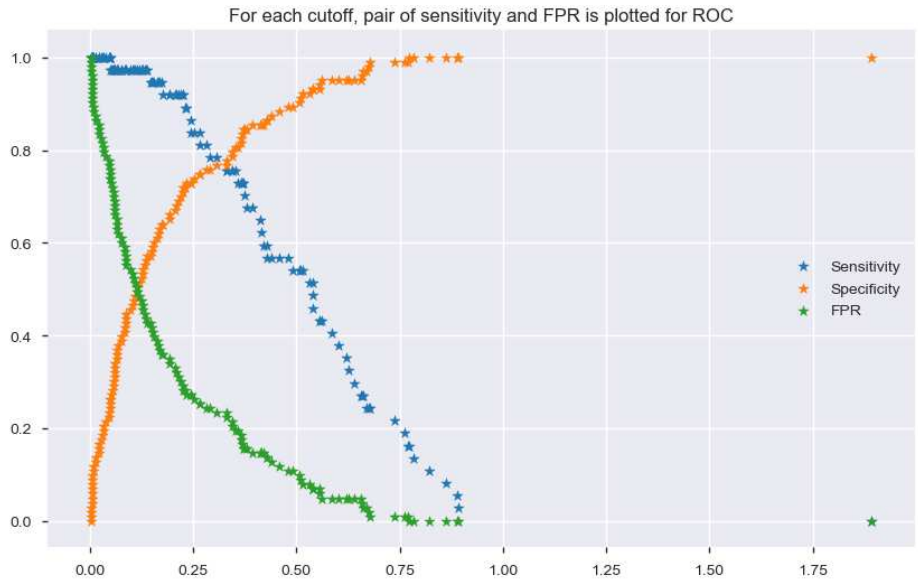
In [141]:
```
roc_like_df[roc_like_df['total']==roc_like_df['total'].max()]
```

Out[141]:

	cutoff	falsepositiverate	sensitivity	specificity	total
63	0.225256	0.281553	0.918919	0.718447	1.637366

In [142]:
```
plt.subplots(figsize=(10,6))
plt.scatter(roc_like_df['cutoff'], roc_like_df['sensitivity'], marker='*', label='Sensitivity')
plt.scatter(roc_like_df['cutoff'], roc_like_df['specificity'], marker='*', label='Specificity')
plt.scatter(roc_like_df['cutoff'], roc_like_df['falsepositiverate'], marker='*', label='FPR')
plt.title('For each cutoff, pair of sensitivity and FPR is plotted for ROC')
plt.legend()

plt.show()
```



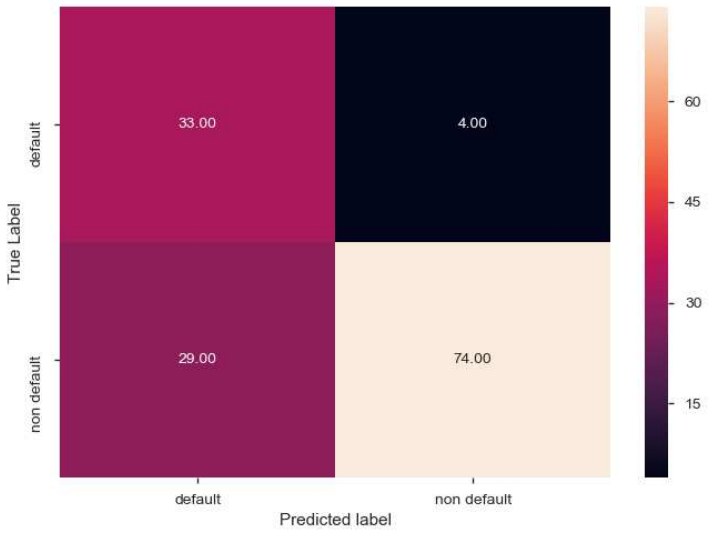
In [143]:
```
#Predicting with new cut-off probability
data_test_pred['new_labels'] = data_test_pred['default_1'].map(lambda x:1 if x >= 0.225317 else 0 )
data_test_pred.head(10)
```

Out[143]:

	index	actual	predicted	default_0	default_1	new_labels
0	145	0.0	0.0	0.807247	0.192753	0
1	526	0.0	0.0	0.940932	0.059068	0
2	689	0.0	0.0	0.710545	0.289455	1
3	430	1.0	1.0	0.178746	0.821254	1
4	245	0.0	0.0	0.633379	0.366621	1
5	491	1.0	0.0	0.718020	0.281980	1
6	696	0.0	0.0	0.785084	0.214916	0
7	504	1.0	1.0	0.338841	0.661159	1
8	380	0.0	0.0	0.913304	0.086696	0
9	566	0.0	0.0	0.978656	0.021344	0

In [144]:
```
#creating a confusion matrix
cm = metrics.confusion_matrix(data_test_pred.actual,
                              data_test_pred.new_labels,[1,0])
sns.heatmap(cm,annot= True,fmt = '.2f',xticklabels=['default','non default'],yticklabels=['default','non default'])
plt.ylabel('True Label')
plt.xlabel('Predicted label')
```

Out[144]: Text(0.5,28.3611,'Predicted label')



In [145]:
```
### Classification Report
print(metrics.classification_report(data_test_pred.actual,data_test_pred.new_labels))
```

```
              precision    recall  f1-score   support

         0.0       0.95      0.72      0.82       103
         1.0       0.53      0.89      0.67        37

   micro avg       0.76      0.76      0.76       140
   macro avg       0.74      0.81      0.74       140
weighted avg       0.84      0.76      0.78       140
```

In [146]:
```
##Intuitively the ability of the classifier to find all the positive samples

recall_score = metrics.recall_score(data_test_pred.actual,data_test_pred.new_labels)
print('recall_model :',round(recall_score,3))
```

```
('recall_model :', 0.892)
```

In [147]:
```
#find the overall accuracy of model
acc_score = metrics.accuracy_score(data_test_pred.actual,data_test_pred.new_labels)
print("Accuracy of Logistic Regression model :",round(acc_score,3))
```

```
('Accuracy of Logistic Regression model :', 0.764)
```

Building Decision Tree

In [148]:
```
from sklearn.linear_model import Lasso,Ridge,ElasticNet
from sklearn.ensemble import RandomForestClassifier,GradientBoostingClassifier
```

In [149]:
```
#make a pipeline for decision tree model
pipelines = {
 'dtclass': make_pipeline(DecisionTreeClassifier(random_state = 10))
 }
```

In [150]:
```
#To check the accuracy of the pipeline
scores = cross_validate(pipelines['dtclass'],X_train,y_train,return_train_score= True)
scores['test_score'].mean()
```

Out[150]:
```
0.732160312805474
```

Cross-Validation and Hyper Parameters Tuning Cross Validation is the process of finding the best combination of parameters for the model by traning and evaluating the model for each combination of the parameters Declare a hyper-parameters to fine tune the Decision Tree Classifier

Decision Tree is a greedy alogritum it searches the entire space of possible decision trees. so we need to find a optimum parameter(s) or criteria for stopping the decision tree at some point. We use the hyperparameters to prune the decision tree

In [151]:
```
#list of trouble hyper paarameters for decision tree classifier pipeline

pipelines['dtclass'].get_params().keys()
```

Out[151]:
```
['decisiontreeclassifier__min_samples_leaf',
 'decisiontreeclassifier__random_state',
 'decisiontreeclassifier__max_leaf_nodes',
 'decisiontreeclassifier__min_samples_split',
 'decisiontreeclassifier',
 'decisiontreeclassifier__min_weight_fraction_leaf',
 'decisiontreeclassifier__presort',
 'decisiontreeclassifier__min_impurity_split',
 'steps',
 'decisiontreeclassifier__min_impurity_decrease',
 'memory',
 'decisiontreeclassifier__class_weight',
 'decisiontreeclassifier__max_depth',
 'decisiontreeclassifier__max_features',
 'decisiontreeclassifier__splitter',
 'decisiontreeclassifier__criterion']
```

In [152]:
```
decisiontree_pyperparameters = {
    'decisiontreeclassifier__max_depth': np.arange(3,10),
    'decisiontreeclassifier__max_features': np.arange(3,8),
    'decisiontreeclassifier__min_samples_split': np.arange(2,15),
    'decisiontreeclassifier__min_samples_leaf': np.arange(1,3)
}
```

Decision Tree classifier with gini index

Fit and tune models with cross-validation Now that we have our pipelines and hyperparameters dictionaries declared, we're ready to tune our models with cross-validation.

- We are doing 5 fold cross validation

In [153]:
```
#Create a crosss validation object from decision tree classifier and its hyperparameters
dtclass_model = GridSearchCV(pipelines['dtclass'],decisiontree_pyperparameters,cv =5,n_jobs = -1)
#fit the model
dtclass_model.fit(X_train,y_train)
```

Out[153]:
```
GridSearchCV(cv=5, error_score='raise-deprecating',
       estimator=Pipeline(memory=None,
     steps=[('decisiontreeclassifier', DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=10,
            splitter='best'))]),
       fit_params=None, iid='warn', n_jobs=-1,
       param_grid={'decisiontreeclassifier__min_samples_leaf': array([1, 2]), 'decisiontreeclassifier__min_samples_split': arra
y([ 2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14]), 'decisiontreeclassifier__max_depth': array([3, 4, 5, 6, 7, 8, 9]), 'de
cisiontreeclassifier__max_features': array([3, 4, 5, 6, 7])},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring=None, verbose=0)
```

In [154]:
```
#display the best parameters for decision tree model
dtclass_model.best_params_
```

Out[154]:
```
{'decisiontreeclassifier__max_depth': 5,
 'decisiontreeclassifier__max_features': 3,
 'decisiontreeclassifier__min_samples_leaf': 1,
 'decisiontreeclassifier__min_samples_split': 11}
```

In [155]:
```
# best score of decision tree model
dtclass_model.best_score_
```

Out[155]:
```
0.7839285714285714
```

In [156]:
```
#In Pipeline we can use the string names to get the decisiontreeclassifier
dtclass_best_model = dtclass_model.best_estimator_.named_steps['decisiontreeclassifier']
dtclass_best_model
```

Out[156]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=5,
            max_features=3, max_leaf_nodes=None, min_impurity_decrease=0.0,
            min_impurity_split=None, min_samples_leaf=1,
            min_samples_split=11, min_weight_fraction_leaf=0.0,
            presort=False, random_state=10, splitter='best')
```

Model Performance Evaluation

- On Test Data

In [157]:
```
#Predictio=ng the test cases
data_test_pred_dtclass = pd.DataFrame({'achual':y_test,'predicted': dtclass_best_model.predict(X_test)})
data_test_pred_dtclass = data_test_pred_dtclass.reset_index()
data_test_pred_dtclass.head()
```

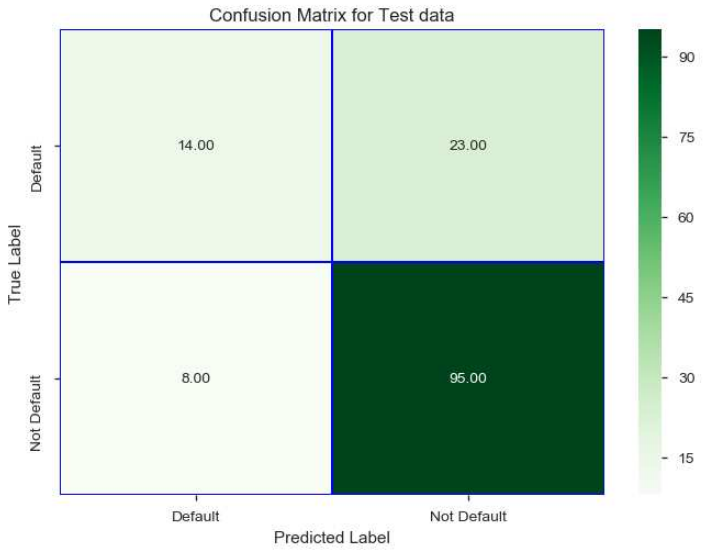
Out[157]:

	index	achual	predicted
0	145	0.0	0.0
1	526	0.0	0.0
2	689	0.0	0.0
3	430	1.0	1.0
4	245	0.0	0.0

In [158]:
```
#creating a confusion matrix
cm_dtclass = metrics.confusion_matrix(data_test_pred_dtclass.achual,data_test_pred_dtclass.predicted,labels = [1,0])
cm_dtclass
```

Out[158]:
```
array([[14, 23],
       [ 8, 95]], dtype=int64)
```

In [159]:
```
sns.heatmap(cm_dtclass,annot= True,fmt='.2f',cmap='Greens',linewidths= .5 , linecolor= 'blue',xticklabels=['Default','Not Defaul
plt.title('Confusion Matrix for Test data')
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.show()
```



In [160]:
```
#probability of prediction
predict_prob_dt = pd.DataFrame(dtclass_best_model.predict_proba(X_test),columns=['default_0','default_1'])
predict_prob_dt.head()
```

Out[160]:

	default_0	default_1
0	0.560000	0.440000
1	0.958763	0.041237
2	0.560000	0.440000
3	0.000000	1.000000
4	0.800000	0.200000

In [161]:
```
data_test_pred_dtclass = pd.concat([data_test_pred_dtclass,predict_prob_dt],axis=1)
data_test_pred_dtclass.head()
```

Out[161]:

	index	achual	predicted	default_0	default_1
0	145	0.0	0.0	0.560000	0.440000
1	526	0.0	0.0	0.958763	0.041237
2	689	0.0	0.0	0.560000	0.440000
3	430	1.0	1.0	0.000000	1.000000
4	245	0.0	0.0	0.800000	0.200000

In [162]:
```
# find the auc score
auc_score = metrics.roc_auc_score(data_test_pred_dtclass.achual, data_test_pred_dtclass.default_1)
round(auc_score,4)
```

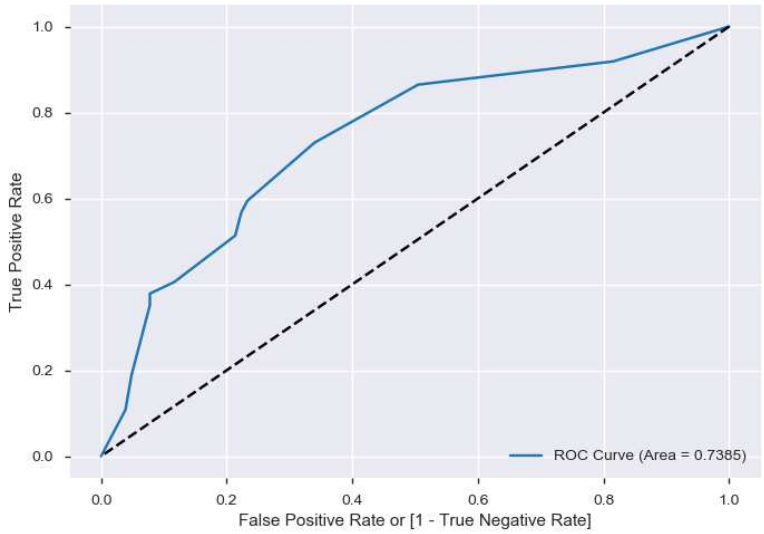
Out[162]: 0.7385

In [163]:
```
#plotting the roc curve
fpr, tpr, thresholds = metrics.roc_curve(data_test_pred_dtclass.achual, data_test_pred_dtclass.default_1,drop_intermediate=False

plt.plot(fpr, tpr, label = "ROC Curve (Area = %0.4f)" % auc_score)
plt.plot([1,0],[1,0],'k--')

plt.ylabel("True Positive Rate")
plt.xlabel("False Positive Rate or [1 - True Negative Rate]")

plt.legend(loc = "lower right")
plt.show()
```



In [164]:
```
# find precision score
prec_score = metrics.precision_score(data_test_pred_dtclass.achual,data_test_pred_dtclass.predicted)
print('Precision score :',round(prec_score,3))
```

```
('Precision score :', 0.636)
```

In [165]:
```
# finding overall accuracy of model
acc_score = metrics.accuracy_score(data_test_pred_dtclass.achual,data_test_pred_dtclass.predicted)
print('Accuracy score :',round(acc_score,3))
```

```
('Accuracy score :', 0.779)
```

In [166]:
```
print(metrics.classification_report(data_test_pred_dtclass.achual,data_test_pred_dtclass.predicted))
```

```
              precision    recall  f1-score   support

         0.0       0.81      0.92      0.86       103
         1.0       0.64      0.38      0.47        37

   micro avg       0.78      0.78      0.78       140
   macro avg       0.72      0.65      0.67       140
weighted avg       0.76      0.78      0.76       140
```

In [167]:
```
from sklearn.externals.six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus as pdot
```

In [168]:
```
# writing the dot data
dot_data= StringIO()
```

In [169]:
```
# export the decesion tree along with the features names into a dot file format
export_graphviz(dtclass_best_model,out_file = dot_data,filled= True,special_characters=True,rounded=True,feature_names=X_train.c
```

In [170]:
```
# make a graph from dot line
graph = pdot.graph_from_dot_data(dot_data.getvalue())
```

In [171]:
```
Image(graph.create_png())
```

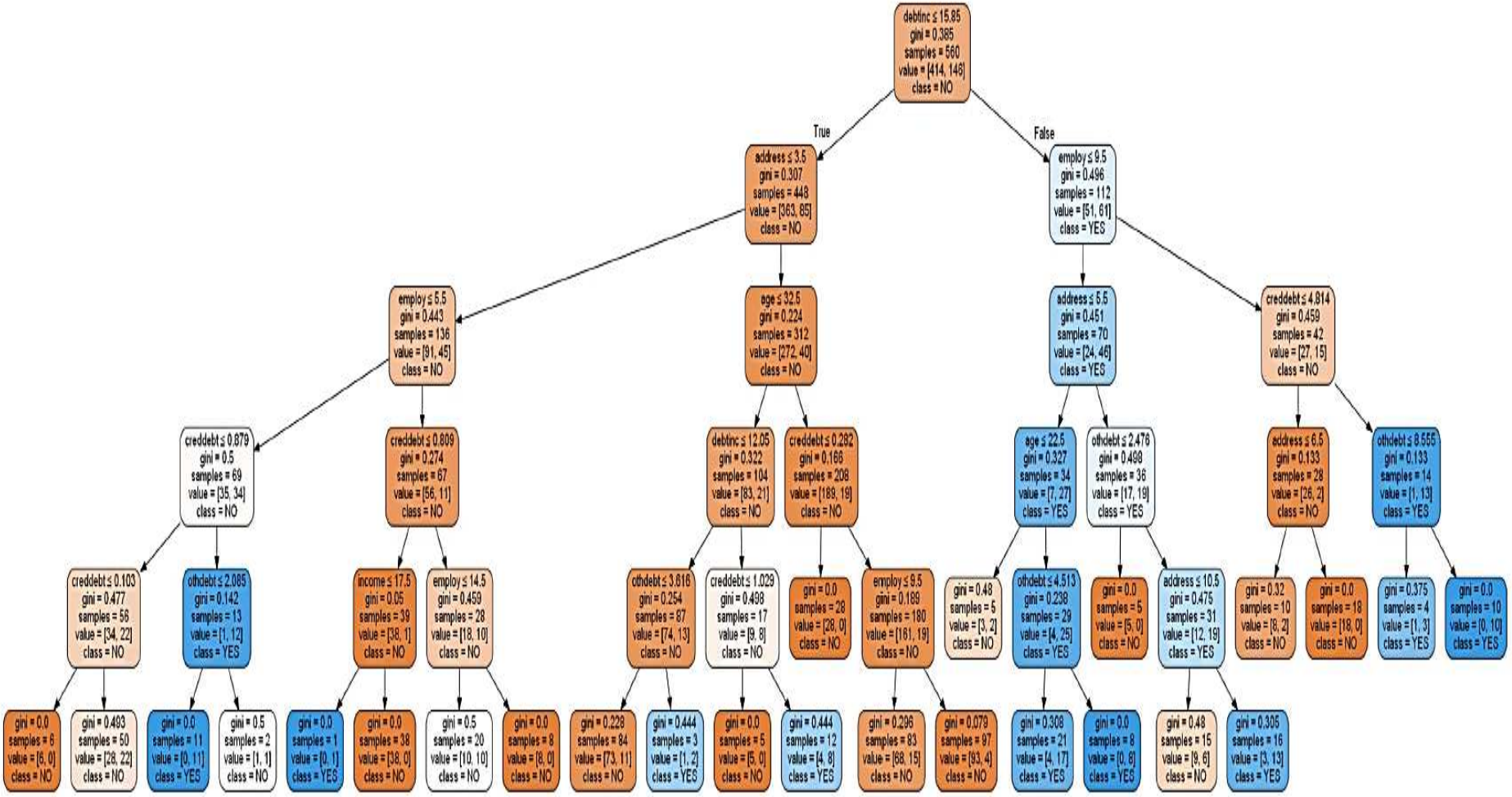


Figure 24 - Decision Tree graph

Model Selecton and Bussiness Insights

- Based on the F1-score (harmonic mean of precision and recall), logistic model with f1 score (for positive labels - default customers) of 0.66 is giving better results than decision tree model with f1 score of 0.44. So we will use the logistic regression model to predict the credit worthiness of the customers

We will Predict the credit risk for remaining 150 customers using the logistic modelwith cuttoff as 0.225

In [172]:
```
#probability for new customers
new_cust_prob = pd.DataFrame(log_reg.predict_proba(data_new[f_col]))
new_cust_prob.columns = ['prob_default_0','prob_default_1']
new_cust_prob.index = data_new.index
new_cust_prob.head()
```

Out[172]:

	prob_default_0	prob_default_1
700	0.985938	0.014062
701	0.942671	0.057329
702	0.322931	0.677069
703	0.918011	0.081989
704	0.634287	0.365713

In [173]:
```
data_new_predicted = pd.concat([data_new,new_cust_prob],axis=1)
data_new_predicted.head()
```

Out[173]:

	age	ed	employ	address	income	debtinc	creddebt	othdebt	default	prob_default_0	prob_default_1
700	36	1	16	13	32	10.9	0.544128	2.943872	NaN	0.985938	0.014062
701	50	1	6	27	21	12.9	1.316574	1.392426	NaN	0.942671	0.057329
702	40	1	9	9	33	17.0	4.880700	0.729300	NaN	0.322931	0.677069
703	31	1	5	7	23	2.0	0.046000	0.414000	NaN	0.918011	0.081989
704	29	1	4	0	24	7.8	0.866736	1.005264	NaN	0.634287	0.365713

In [174]:
```
#using the cutoff value we will predict the default
data_new_predicted['predicted_default'] = data_new_predicted['prob_default_1'].map(lambda x:1 if x >= 0.225317 else 0 )
```

In [175]:
```
data_new_predicted.head()
```

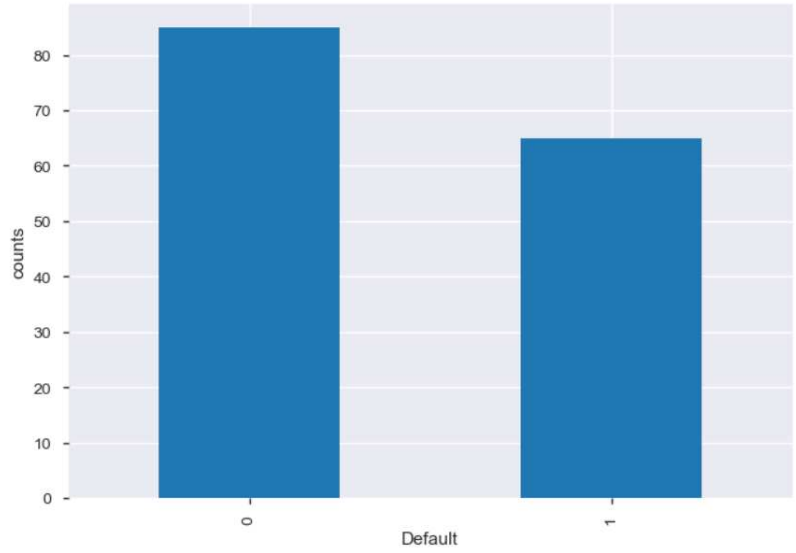
Out[175]:

	age	ed	employ	address	income	debtinc	creddebt	othdebt	default	prob_default_0	prob_default_1	predicted_default
700	36	1	16	13	32	10.9	0.544128	2.943872	NaN	0.985938	0.014062	0
701	50	1	6	27	21	12.9	1.316574	1.392426	NaN	0.942671	0.057329	0
702	40	1	9	9	33	17.0	4.880700	0.729300	NaN	0.322931	0.677069	1
703	31	1	5	7	23	2.0	0.046000	0.414000	NaN	0.918011	0.081989	0
704	29	1	4	0	24	7.8	0.866736	1.005264	NaN	0.634287	0.365713	1

In [176]:
```
#model Prediction
data_new_predicted.predicted_default.value_counts()
```

Out[176]:
```
0    85
1    65
Name: predicted_default, dtype: int64
```

In [177]:
```
#model Prediction
data_new_predicted.predicted_default.value_counts().plot.bar()
plt.ylabel('counts')
plt.xlabel('Default')
plt.show()
```



Insights

- Out of 150 customers, model as predicted that 85 customer are not going to default on the bank loan and remaining 65 customer would most likely default on the loan.

Model Performance Validation

- KS Chart
- Lift and Gain Chart

we will use the concept of dicile analysis for these validations

In [178]:
```
#For train data

train_predict = pd.DataFrame({'achual':y_train.reset_index(drop = True),
                              'prob': pd.DataFrame(log_reg.predict_proba(X_train))[1]})
train_predict['predicted']= train_predict['prob'].map(lambda x: 1 if x > 0.225 else 0 )
train_predict.head()
```

Out[178]:

	achual	prob	predicted
0	0.0	0.265732	1
1	0.0	0.008766	0
2	0.0	0.033063	0
3	0.0	0.152962	0
4	1.0	0.134143	0

In [179]:
```
#for test data
test_predict = pd.DataFrame({'achual': y_test.reset_index(drop = True),
                             'prob':pd.DataFrame(log_reg.predict_proba(X_test))[1]})
test_predict['predicted'] = test_predict['prob'].map(lambda x: 1 if x > 0.225 else 0 )
test_predict.head()
```

Out[179]:

	achual	prob	predicted
0	0.0	0.192753	0
1	0.0	0.059068	0
2	0.0	0.289455	1
3	1.0	0.821254	1
4	0.0	0.366621	1

In [180]:
```
#splitting train data into different diciles

train_predict['Deciles']=pd.qcut(train_predict['prob'],10,labels = False)
train_predict.head()
```

Out[180]:

	achual	prob	predicted	Deciles
0	0.0	0.265732	1	6
1	0.0	0.008766	0	0
2	0.0	0.033063	0	1
3	0.0	0.152962	0	4
4	1.0	0.134143	0	3

In [181]:
```
#splitting test data into different diciles

test_predict['Deciles']=pd.qcut(test_predict['prob'],10,labels = False)
test_predict.head()
```

Out[181]:

	achual	prob	predicted	Deciles
0	0.0	0.192753	0	5
1	0.0	0.059068	0	2
2	0.0	0.289455	1	6
3	1.0	0.821254	1	9
4	0.0	0.366621	1	6

In [182]:
```
#sumation od deciles for the train data
train_predict[['Deciles','achual']].groupby(train_predict.Deciles).sum().sort_index(ascending = False)
```

Out[182]:

	Deciles	achual
Deciles		
9	504	44.0
8	448	31.0
7	392	18.0
6	336	20.0
5	280	12.0
4	224	8.0
3	168	7.0
2	112	4.0
1	56	2.0
0	0	0.0

In [183]:
```
#sumation of deciles for the train data
test_predict[['Deciles','achual']].groupby(test_predict.Deciles).sum().sort_index(ascending = False)
```

Out[183]:

	Deciles	achual
Deciles		
9	126	10.0
8	112	10.0
7	98	6.0
6	84	4.0
5	70	4.0
4	56	2.0
3	42	0.0
2	28	0.0
1	14	1.0
0	0	0.0

In [184]:
```
train_predict[['Deciles','achual']].groupby(train_predict.Deciles).count().sort_index(ascending = False)
```

Out[184]:

	Deciles	achual
Deciles		
9	56	56
8	56	56
7	56	56
6	56	56
5	56	56
4	56	56
3	56	56
2	56	56
1	56	56
0	56	56

In [185]:
```
test_predict[['Deciles','achual']].groupby(test_predict.Deciles).count().sort_index(ascending = False)
```

Out[185]:

Deciles	Deciles	achual
9	14	14
8	14	14
7	14	14
6	14	14
5	14	14
4	14	14
3	14	14
2	14	14
1	14	14
0	14	14

KS & LIft, Gain Chart

- Training dataset
- Testing dataset

In [186]:
```
from IPython.display import Image
```

In [187]:
```
#for Training data set
Image(filename= 'Images/KS-Traindata.png',width=600)
```

Out[187]:

Training Sample

Decile	Defaulters	Non Defaulters	Total	Default RATE	Default PERCENTAGE	CUMU. Default PERCENT	Non Default PERCENT	CUMU. Non Default PERCENT	KS
1	44	12	56	78.57%	30.14%	30.14%	2.90%	2.90%	27.24%
2	31	25	56	55.36%	21.23%	51.37%	6.04%	8.94%	42.43%
3	18	38	56	32.14%	12.33%	63.70%	9.18%	18.12%	45.58%
4	20	36	56	35.71%	13.70%	77.40%	8.70%	26.81%	50.59%
5	12	44	56	21.43%	8.22%	85.62%	10.63%	37.44%	48.18%
6	8	48	56	14.29%	5.48%	91.10%	11.59%	49.03%	42.06%
7	7	49	56	12.50%	4.79%	95.89%	11.84%	60.87%	35.02%
8	4	52	56	7.14%	2.74%	98.63%	12.56%	73.43%	25.20%
9	2	54	56	3.57%	1.37%	100.00%	13.04%	86.47%	13.53%
10	0	56	56	0.00%	0.00%	100.00%	13.53%	100.00%	0.00%
	146	414	560	26%				KS	50.59%

In [188]:
```
#For Test data set
Image(filename="Images/KS-Testdata.png",width=600)
```

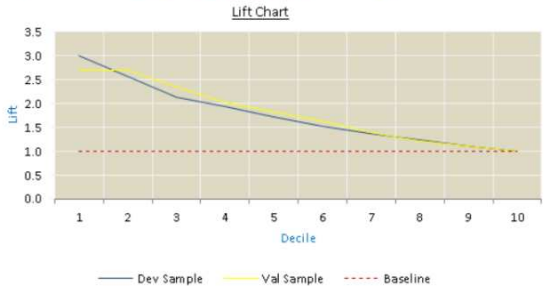
Out[188]:

Test Sample

Decile	Defaulters	Non Defaulters	Total	Default RATE	Default PERCENTAGE	CUMU. Default PERCENT	Non Default PERCENT	CUMU. Non Default PERCENT	KS
1	10	4	14	71.43%	27.03%	27.03%	3.88%	3.88%	23.14%
2	10	4	14	71.43%	27.03%	54.05%	3.88%	7.77%	46.29%
3	6	8	14	42.86%	16.22%	70.27%	7.77%	15.53%	54.74%
4	4	10	14	28.57%	10.81%	81.08%	9.71%	25.24%	55.84%
5	4	10	14	28.57%	10.81%	91.89%	9.71%	34.95%	56.94%
6	2	12	14	14.29%	5.41%	97.30%	11.65%	46.60%	50.70%
7	0	14	14	0.00%	0.00%	97.30%	13.59%	60.19%	37.10%
8	0	14	14	0.00%	0.00%	97.30%	13.59%	73.79%	23.51%
9	1	13	14	7.14%	2.70%	100.00%	12.62%	86.41%	13.59%
10	0	14	14	0.00%	0.00%	100.00%	13.59%	100.00%	0.00%
	37	103	140					KS	56.94%

In [189]:
```
##Lift Chart
Image(filename="Images/LiftChart.png",width=500)
```

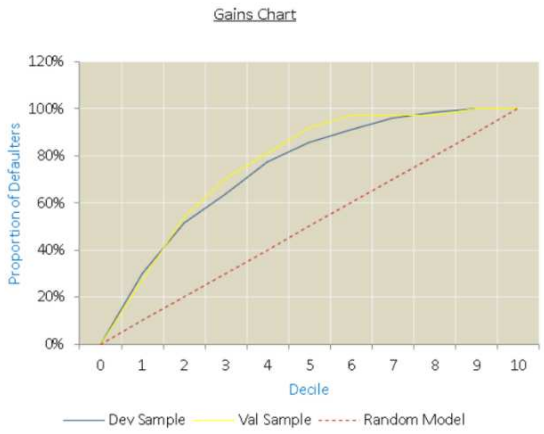
Out[189]:



t

In [190]:
```
##Gain Chart
Image(filename="Images/GainsChart.png",width=500)
```

Out[190]:



Obsevations

- Gain chart tells % of targets(events)covered at a given decile level. In the current case ,we can say that we can identify 90% of the defaulters who are likely to default on the loan by just analyzing 50% of the total customers.
- Lift chart measures how much better one can expect to do with the predictive model comparing without a model. In the current model, cummulative lift for top two deciles is 2.7, means that by selecting 20% of the records based on the model. One can expect 2.7 times the total number of defaulters to be found than the randomly selecting 20% of the data without a model.

Finally, let's save the winning model.

- We need to save our prediction models to file, and then restore them in order to reuse our previous work to: test our model on new data, compare multiple models, or anything else.
- Pickle is the standard way of serializing objects in Python.Pickle operation to serialize your machine learning algorithms and save the serialized format to a file.
- Later we can load this file to deserialize your model and use it to make new predictions.

In [191]:
```
import pickle
```

Let's save the winning Logistic Model Object into a pickle file.

In [192]:
```
with open('Final_model.pkl','wb') as f:
    pickle.dump(log_reg,f)
```

In [108]:
```
# some time later...

# load the model from disk - use to classify the default customers directly
#loaded_model = pickle.load(open('OutPutModel/final_model.pkl', 'rb'))
```