វិទ្យាស្ថានជាតិប្រៃសណីយ៍ ទូរគមនាគមន៍បច្ចេកវិទ្យាគមនាគមន៍ និងពត៌មាន

NATIONAL INSTITUTE OF POSTS, TELECOMS AND ICT

SCHOOL OF COMPUTER SCIENCE

Thesis report on

"ការវិភាគទឹន្នន័យរបស់ស្ថាប័ណ្ណផ្តល់ប្រាក់កម្ចី"

Peer-to-Peer Lending Club Analysis

At

DATAU

CREATING TOMORROW'S DATA LEADERS, TODAY

July 14th - November 14th,2020

A thesis

In partial fulfillment of the requirement for the degree of

Bachelor degree of Computer science

Submitted by

Ms. Chisang MAM

Under supervision of

Mr. Bonpagna KANN

December 2020

វិទ្យាស្ថានជាតិ ព្រៃសណីយ៍ទូរគមនាគមន៍ បច្ចេកវិទ្យាគមនាគមន៍ និងព័ត៌មាន

មហាវិទ្យាល័យវិទ្យាសាស្ត្រកុំព្យូទ័រ

គម្រោងសញ្ញាបត្របរិញ្ញាបត្រ

របស់និស្សិត៖ ម៉ម ជីសាង

កាលបរិច្ឆេទការពារនិក្ខេបបទ៖

អនុញ្ញាតអោយការពារគំរោង៖

ប្រធានវិទ្យាស្ថានជាតិ៖ - _សេង សុភាព__

ថ្ងៃទី ខែ ឆ្នាំ ២០២០

ប្រធានបទ៖ ការិភាគទិន្នន័យរបស់ស្ថាប័ណ្ណផ្តល់ប្រាក់កម្ចី

សហគ្រាស៖	ជាតាយូបាយមេគង្គប៊ីកជាតា
ព្រឹទ្ធបុរស៖	ឯកឧត្តមបណ្ឌិត សំ សិទ្ធិសេរី
គ្រូជីកនាំគម្រោង៖	លោក កាន់ បុណ្ណបញ្ញា
អ្នកទទួលខុសត្រូវក្នុងសហគ្រាស៖	លោក ឈឹម លីម៉េង

រាជធានីភ្នំពេញ

NATIONAL INSTITUTE OF POSTS, TELECOMS AND ICT

School of Computer Science

A thesis submitted by:

Ms. MAM Chisang

Defense Date: 24 December, 2020

Thesis defense authorization

President: SENG Sopheap

December 24 ,2020

Topic: Peer to peer lending club analysis

Enterprise:	**DataU by Mekong Big Data**
Dean:	**H.E Dr. SAM Sethserey**
Advisor:	**Mr. KANN Bonpangna**
Supervisor:	**Mr. CHHIM Lymeng**

PHNOM PENH

Acknowledgement

First of all, I would like to give the greatest respect and thank my parents who raised me up from when I was born until now. They give me such a warm family and love unconditionally. I can't imagine my life without them, I would not have done so far to this point of success.

I would like to respectfully yours with sincere gratitude to President **Dr. SENG Sopheap**, President of the National Institute of Posts, Telecoms and ICT, and other respectful NIPTICT co-founders for essentially founding NIPTICT with providing such an amazing study course.

I would like to show my appreciation to **Mr. Bonpagna Kann** who is a professional Data scientist and also my valuable internship advisor for his value of time, kindness with encouragement of support and guidance until the completion of this internship. He's the only one who will always give me warm advice with an effective solution .

Lastly, I owe my deep gratitude to **DataU by Mekong Big data**, **Mr. Lymeng Chhim**, Data scientist and **Ioni Spinu**, soft skill director and every staff in the company provide such nice support and guidance all along the project, although they have a busy schedule managing the corporate affairs.

អូលឆ័យសង្ខេប

របាយការណ៍និក្ខេតបទនេះនឹងនិយាយអំពីកម្មសិក្សារបស់ខ្ញុំរយៈពេល៤ខែ ក្នុងនាមជាសិក្ខាកាម អ្នកវិទ្យាសាស្ត្រទិន្នន័យ។ កម្មសិក្សានេះ គឺជាតម្រូវការយ៉ាងសំខាន់នៃការបញ្ចប់ថ្នាក់បរិញ្ញាប័ត្រវិទ្យាសាស្ត្រកុំព្យូទ័រ ចាប់ពីថ្ងៃទី ១៤ ខែកក្កដា ដល់ ថ្ងៃទី ១៤ ខែវិច្ឆិការ ឆ្នាំ ២០២០ ក្រោមប្រធានបទ "ការវិភាគទិន្នន័យរបស់ក្លឹបផ្តល់ប្រាក់កម្ចី" ។

ក្នុងគម្រោងនេះនឹងពិពណ៌នាអំពីអតិថិជនដែលស្នើរសុំប្រាក់កម្ចីទៅក្នុហប្រព័ន្ធរបស់យើងដែលមានសារវាតានៃព័ត៌មាននីមួយៗដូចជា ប្រាក់ចំណូល ប្រវត្តិប្រាក់កម្ចី -ល-។ ខ្ញុំនឹងប្រើសំណុំទិន្នន័យនេះដើម្បីធ្វើការវិភាគចំនួន៤ដំហ៊ាន គឺ ប្រើSQL ដើម្បីសិក្សារអំពីទិន្នន័យដោយប្រើ visual tool ដើម្បីបង្ហាញទិន្នន័យលើក្រាប បន្ទាប់មក ខ្ញុំនឹងផ្តល់ព័ត៌មានទូលំទូលាយអំពីប្រាក់កម្ចីដែលបានចេញសម្រាប់ត្រីមាសចុងក្រោយនៃឆ្នាំ២០១៩ និង មើលថា តើមុខជំនួញនេះដំនើរការយ៉ាងដូចម្តេចដែរ? ម្យ៉ាងទៀត ខ្ញុំនឹងពិនិត្យមើលលក្ខណៈអ្នកខ្ចីដើម្បីយល់ដឹងថាអតិថិជននោះគួរផ្តល់ប្រាក់កម្ចីឬទេ ដោយផ្អែកលើប្រវត្តិខ្ចីរបស់ពួកគាត់? ចុងក្រោយខ្ញុំបង្កើតគំរូនៃការព្យាករណ៍ដូចជាដើមឈើសេចក្តីសំរេចព្រៃឈើព្រួដិននិងអ្នកជិតខាងដែលនៅជិតK ដើម្បីព្យាករណ៍ថាតើប្រាក់កម្ចីទាំងនោះនឹងក្លាយជាប្រាក់កម្ចីមិនសងដោយអ្នកខ្ចីឬអត់។

Abstract

This thesis report is about my 4 months internship as a data scientist trainee. This internship is an important requirement of a bachelor Degree in Computer Science from July 14 to November 14, 2020 under the heading **"Peer-to-peer Lending Club Analysis"**.

This dataset describes the borrowers who request loans into our system which contain the information background of each of them such as income, occupation, loan history, characteristics and so on.

In this project, I will use this dataset to do 4 steps which are as SQL to learn about data, using visual tools to visualize those data, Exploratory data analysis and prediction model. I will provide the comprehensive information about loans that LC issued for the last quarter of 2019 and see how the business looks like so far. On the other hand, I will inspect into the borrowers characteristics in order to get a better understanding of the borrowers. Then the next important part, I will dive deeper to see if the grade does really guarantee the good borrowers and figure out if there's any risk when people take the loan to repay their previous loans. Lastly, I build prediction model such as Decision tree, Random forest and K-nearest neighbor to predict whether those loans are going to be default loan or not.

Abbreviation

P2P : Peer to peer

EDA: Exploratory data analysis

KNN: K-nearest neighbor

AUC: Area under curve

CV: Cross validation

Table of Contents

Table of figure

List of tables

1. General presentation of the company

1.1 History

Honoring our founders' vision, DataU Academy remains intentional in its practice by providing a platform for innovative, collaborative and value driven learning on current and disruptive topics aimed at boosting our clients success and growth by supporting your journey into employment.

1.2 Service

DataU Academy is the educational business of Mekong Big Data, Cambodia. Our goal is to help improve the knowledge, skills, competence and expertise of data professionals and business.

- An array of business formulation strategies
- Revenue generation and user engagement plans
- Charting company growth via new metrics
- Establishing a strong foothold in the industry

The mission of DataU Academy, is to guide its trainees to develop an informed curiosity and a lasting passion for learning. We are dedicated to inspiring trainees to strive for excellence through the teaching of interpersonal and data science skills in order to make lasting contributions across the globe through critical thinking and problem solving.

We grow and develop the talent by providing training programs that build mindset, knowledge and skills for you to find the perfect data specialist job.

1.3 Location and contact

Address: Raintree Preah Ang Doung, Khan Doun Penh, Phnom Penh

Phone: 077 844 381

Email: Faria@Mekongbigdata.com

DATAU
CREATING TOMORROW'S DATA LEADERS, TODAY

Figure 1: DataU's Logo

2. Introduction to the project

2.1 Presentation of the project

Peer to peer lending is the practice of lending money to individuals or businesses through online services that match lenders with borrowers. Lending Club (LC) is a peer to peer online lending platform. It is one the world's largest marketplace connecting borrowers and investors, where consumers and small business owners lower the cost of their credit and enjoy a better experience than traditional bank lending, and investors earn attractive risk-adjusted returns. Customers interested in a loan complete a simple application at LendingClub.com. LC uses online data and technology to quickly assess risk, determine a credit rating and assign appropriate interest rates. Qualified applicants receive offers in just minutes and can evaluate loan options with no impact to their credit score. Investors ranging from individuals to institutions select loans in which to invest and can earn monthly returns. The entire process is online, using technology to lower the cost of credit and pass the savings back in the form of lower rates for borrowers and solid returns for investors. By allowing our members to directly invest in and borrow from each other, we avoid the cost and complexity of the bank system and pass saving on to you. Both sides can win at a lower rate for borrowers and better returns to investors.

2.1 Problematic

Peer to peer lending is an online platform that everyone can access and request loans. People who requested come from different areas and positions, they also have different income and history background records. So, clubs are hard to manage those records and also hard to make decisions to lend to those borrowers.

2.2 Objective

To solve this problem, Peer to peer lending clubs retrieve data to analyze and predict about borrowers who want to borrow from our platform. In this project, I will do EDA to understand the meaning behind the dataset to investigate and clean it. Moreover, I also study each variable to compare with interest rate, provide statistical detail of those data by creating correlation with relevant variables and virtualization with python.

For the purpose of finding any borrower who will default which means who won't return the money, I built 3 classification models as a means to find how many people who requested the loan are likely to have default loan status.

2.3 Planning

My internship duration is 16 weeks which start from 14st July to 14th November 2019.The table below illustrates my internship plan during these 16 weeks:

Tasks	Weeks															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Training																
Define Problem																
Setup environment																
Study data																
Data preparation																
Exploratory data analysis																
Predictive model																

Table 1: Planning

2.4 Limitation

In this dataset, There are 47 features and 128258 samples. As you can see, we have a lot of loans which are "current" with a fair amount of fully paid loans. Other categories (including) default have a really low number. This means the data is imbalanced and we might need to do something about this later in the analysis. For now we will drop all the columns except 'Fully Paid', 'Default' and 'Charged off'. We will also merge 'Charged off' and 'Default', "In grace period", 'Late (16-30 days)' and 'Late (31-120 days)' together meaning that anyone who fell into this category defaulted their loan.

3. State Of The Art

3.1 Peer to peer lending club

Peer-to-peer lending could be a modern decentralized demonstration for financial specialists to loan capital and potential borrowers to profit credit. Numerous financial specialists and potential borrowers collaborate on a common online stage that doesn't involve any monetary teach or agents within the end-to-end handle. The borrower demands for an advance sum, which run is categorized on the premise of intrigued rates. The financial specialist has the freedom to select the sum of capital, the intrigued cap and indeed the borrower.

These advances are not totally secure as they include significant chance of default [1] and subsequently require included exertion to recognize and decide a borrower from a pool of obscure clients. Distinguishing potential credit defaults is obligatory in peer-to-peer loaning stages, but deciding the great credits and financing them has higher need, as the P2P income demonstrate is subordinate on the number of credits, volume of credit etc., but causing misfortunes on account of defaults will make the commerce unviable.

The handling of borrower's credit application includes collection of client information like yearly wage, credit history, bank adjust, other credits, etc. The preeminent need is to recognize the subset of these traits that's competent of classifying the advanced application as one with potential default hazard or not.

Since the number of properties collected for each credit record is much less than the number of credit records, tree based classifiers are utilized, as the run-time complexity of building the tree is directly relative to the number of highlights. They are too able to deal with the skewness of the information, which isn't conceivable within the case of "Calculated Relapse", where oversampling is needed.

Tree based classification may be an information mining procedure which recursively builds a set of rules based on numerous input factors that are either numerical or categorical and yields a course. Tree based classifiers like Choice Tree [2], Irregular Woodland [3], Sacking [4] and Additional Trees [5] are utilized to prepare expectation models for the peer-to-peer moneylenders data.

The return on speculation for a financial specialist on any credit is tall in the event that the credit encompasses a higher interest rate, but the chance of defaulting is additionally relatively tall. Speculators can select to play securely by contributing in moo intrigued credits which has higher chances of being reimbursed. Consequently the tree based models are optimized to induce distant better; a much better; a higher; a stronger; an improved">a higher accuracy ahead of exactness. The points of interest and part up of the loan's intrigued rates, is clarified within the up and coming areas.

3.2 Methodologies

A. Dataset description

The dataset consists of 656,724 loan records which were issued by "Lending Club", between the years 2013 and 2015. There are a total of 115 attributes describing the loan application. The "Loan Status" attribute which describes the current state of the loan, has the following values, "Issued", "Current", "Fully paid", "Default", "Charged off', "Late (16-30 days)", "Late (31-120 days)" and "In grace period". These statuses are used to reduce them to a binary classification problem, i.e., the loan applications with "Charged off', "Default", "Late (31-120 days)" and "Late (16-30 days)" are considered as "bad" or "defaulted" loans while "Current", "Fully Paid" and "In grace period" are classified as "good" loans and remaining are ignored. The "Ioan status" attribute is replaced with "is bad" which takes the values of 0 for a good credit and 1 for a bad credit or default.

The loan amount ranges from $1000 to $35,000 and each loan has a "grade" (ranging from A-G) associated with them. The grade specifies the range of interest rates in ascending order, starting from 5.32% to 29%. As expected, the loans with higher interest rates have a higher risk of default. 31% of Loans in grade G are bad while it's only 3% in grade A.

B. Preprocessing and cleaning

The freely accessible "Lending Club" dataset for the long time 2013-2015 gotten from the official site was displayed in a "Comma-Separated-Value" record. The dataset was divided into two sets, 2013-2014 and 2015, having 235,629 and 421,095 records, 111 and 74 traits separately. As it were the traits

which are common to both the datasets are considered, since "Lending Club" has expelled qualities which are not noteworthy for the examination and included a number of unused ones.

The records which had "credit status" as "Issued" are overlooked for investigation. Since this inquiry was carried out in early 2016, the 2015 dataset had as it were 60,000 records that may well be classified into either "great" or "awful" credit, while the remaining were in "Issued" state. 9.3% of 2013-14 dataset was moreover within the same bracket and thus disregarded. The preprocessed information contains 10.9% credit defaults.

Properties having non-numerical values like "Horne possession", "Application sort", "Worker title" are labeled with numbers beginning from 1. The combined dataset has 279,169 with 70 attributes.

1. Feature selection

a. Removing redundant attributes

The traits carrying the same or comparable data are evacuated. Properties like "Financed sum", "Financed sum by financial specialists" were found to be the same as advance endorsed sum in most of the records. Essentially, a add up of 16 traits are eliminated.

b. Missing values

Qualities with 80% (and over) missing values are overlooked, as most of them were found to be discretionary. The remaining properties with 10-15% of lost information, are dealt with by utilizing the middle of the accessible information, as the placeholder. There are 25 such properties disregarded in this handle.

c. Removing excessive attributes

Qualities such as "Final installment date", "Final credit pull date", and "Exceptional vital sum" are populated as it were when the borrower has begun reimbursing it. Preparing the show with these qualities might lead to undesired results, like amazingly tall precision and exactness (near to 100%), because it allows the algorithm to memorize from long-standing time. Consequently 14 such traits are expelled from the dataset.

d. Using domain knowledge

Qualities such as "Final installment date", "Final credit pull date", and "Exceptional vital sum" are populated as it were when the borrower has begun reimbursing it. Preparing the show with these qualities might lead to undesired results, like amazingly tall precision and exactness (near to 100%), because it allows the algorithm to memorize from long-standing time. Consequently 14 such traits are expelled from the dataset.

e. Extra tree classification

Additional Tree or Amazingly Randomized Timberland classifier is a tree based outfit classifier that employs Choice Tree as the base learner. It chooses samples from the whole dataset amid quality part at the hub, at irregular, for each combination within the highlight set and selecting the ideal fit among them. This is cheaper to prepare as the hub parts are totally randomized as opposed to Arbitrary Timberland calculation, where the part creation is optimized.

C. Classification

a. Decision tree

Decision Tree classifier could be an administered learning algorithm which builds a twofold tree where each hub level has an property esteem part. A Choice Tree is built top- down from a root hub and includes dividing the information into subsets that contain similar values. Entropy may be a degree that's utilized to calculate homogeneity. A totally homogeneous sample set will have entropy esteem of and an equitably dispersed sample will have the esteem 1 as its entropy. Data pick up at each level or quality is calculated by utilizing the entropy of the parent and the weighted entirety of entropy of its children. This data pick up esteem acts as the part at each hub. This tree speaks to the set of rules utilized for foreseeing.

	Predicted positive	Predicted negative
Condition positive	202586	46154

Condition negative	6051	24378

Table 2:Decision tree result

b. Random forest

Random forest is another tree based gathering directed learning calculation that creates different Choice Trees (timberland) for arbitrary subsets of the information, and predicts the course with most noteworthy recurrence after running the sample on all the Choice Trees generated. Random woodland makes a difference in overcoming the over-fitting issue experienced in Choice Tree classification. It gives a noteworthy increment within the exactness of the show. The trees created had "Intrigued rate" as the primary part a bit like Decision Tree but the remaining attributes shifted due to haphazardness within the subsets. This show includes an accuracy of 96.2% and 88.5% precision.

	Predicted positive	Predicted negative
Condition positive	225545	23195
Condition negative	8909	21502

Table 3:Random forest result

c. Bagging (Bootstrap Aggregating)

Bootstrapping is an examining procedure to get an surmised measurement gauge of the examining dissemination by choosing an arbitrary subset with substitution, different times and learning from it. Bagging includes bootstrapping the first dataset and utilizing the subsets to assess the execution of another classifying calculation, by voting. The haphazardness presented in bootstrapping diminishes the fluctuation or the over-fitting problem.

	Predicted positive	Predicted negative
Condition positive	224856	23884
Condition negative	8639	21709

Table 4: Bagging result

3.3 Algorithm analysis performance

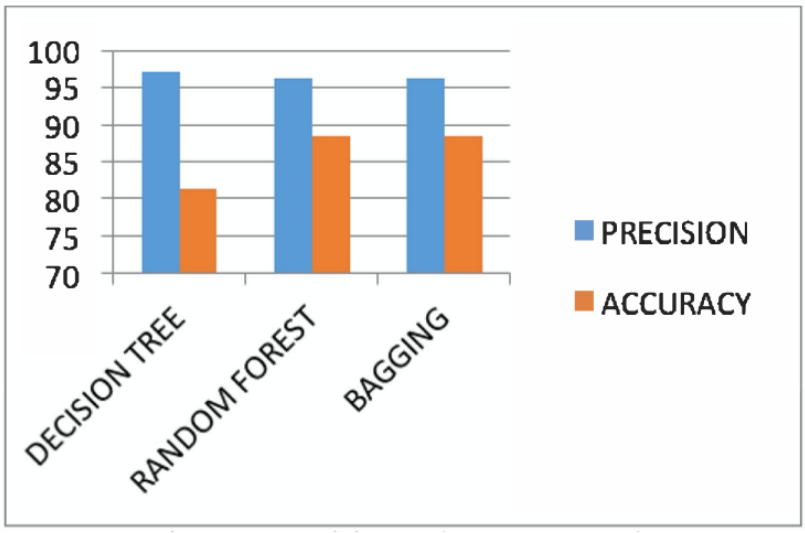


Figure 2: Algorithm analysis performance

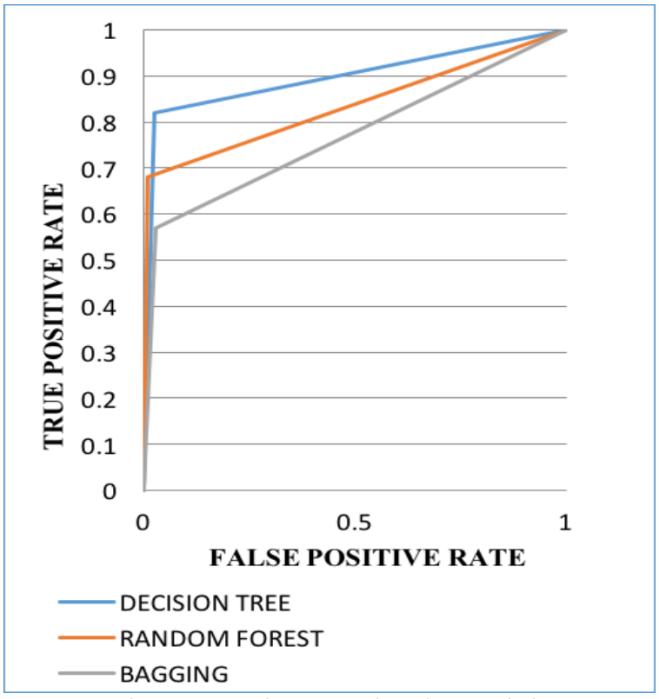


Figure 3: True positive rate

Making strides foreseeing control (exactness) utilizing Choice Tree, Arbitrary Woodland and Sacking calculations was the highlight of this paper. Choice Tree was the finest as appeared within the chart with the accuracy of 97.1% as compared to Arbitrary Woodland and Stowing having exactness of 96%. The accuracy of the latter is lower due to the haphazardness and the skewness included within the subsets utilized for producing the trees. Few of these subsets have favored the recognizable proof of defaults ahead of the great credits. Thus a much higher precision of 88.5% and 88.35% was gotten within the gathering classifiers.

Receiver operating characteristic (ROC) [9] curve is a 2D plot representing the performance of a binary classifier; the curve is plotted with true positive rate (TPR) versus the false positive rate (FPR) at multiple threshold values.

$$TPR = \frac{\sum True\ Positive}{\sum True\ Positive + \sum False\ Negative}$$

$$FPR = \frac{\sum False\ Positive}{\sum False\ Positive + \sum True\ Negative}$$

The following is the algorithm's performance for a given range of area under curve (AUC):

Aue ranges	Performance
0.90 to 1.0	Excellent
0.80 to 0.90	Good
0.70 to 0.80	Fair
0.60 to 0.70	Poor
0.50 to 0.60	Failure

Table 5: AUC

The AUC for Decision Tree is 0.91 which is larger compared to Random Forest's 0.83 and Bagging's 0.77. This confirms that the Decision Tree model will perform well while classifying a good credit.

4. Analysis and General Concept

4.2 Choice of technology

In order to achieve the project smoothly, considering a technology to apply is also an important part. I am considering using Python to apply in this project since it is:

- **Easy to learn:** his language can learn it easily and quickly. When compared to other data science languages like R, Python promotes a shorter learning curve and scores over others by promoting an easy-to-understand syntax.
- **Scalability**: Python has established a lead by emerging as a scalable language, and it is faster than other languages like Matlab and Stata. Python's scalability lies in the flexibility that it gives to solve problems, as in the case of YouTube

that migrated to Python. Python has come good for different usages in different industries and for rapid development of applications of all kinds.

- **Choices of data science libraries**: Python is the variety of data science/data analytics libraries made available for the aspirants. Pandas, StatsModels, NumPy, SciPy, and Scikit-Learn, are some of the libraries well known in the data science community. Python does not stop with that as libraries have been growing over time.
- **Python community**: The widespread and involved community promotes easy access for aspirants who want to find solutions to their coding problems. Whatever queries you need, it is a click or a Google search away. Enthusiasts can also find access to professionals on Codementor and Stack Overflow to find the right answers for their queries.
- **Graphic and virtualization**: Python comes with varied visualization options. The visualization packages help you get a good sense of data, create charts, graphical plot and create web-ready interactive plots.

4.1.1 Languages and scripts

These are libraries that use to implement the project:

SQL is a database computer language designed for the retrieval and management of data in a relational database. SQL stands for Structured Query Language. This tutorial will give you a quick start to SQL. It covers most of the topics required for a basic understanding of SQL and to get a feel of how it works.

Figure 4: SQL logo

Figure 5: Python logo

Python: is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity

and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Scikit-learn: is a library in Python that provides many unsupervised and supervised learning algorithms. It's built upon some of the technology you might already be familiar with, like NumPy, pandas, and Matplotlib.

Figure 6: Scikit learn logo

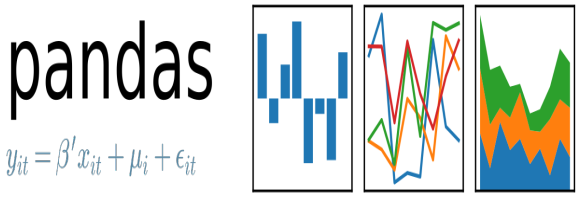


Figure 7: Pandas logo

Pandas: is a Python package providing fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language. It is already well on its way toward this goal.

Matplotlib: is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatterplots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery. For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full

Figure 8: Matplotlib logo

control of line styles, font properties, axes properties, etc, via an object-oriented interface or via a set of functions familiar to MATLAB users.

Figure 9: Seaborn logo

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

4.1.2 Tools

Anaconda: is a free and open source distribution of the Python and R programming languages for data science and machine learning related applications (large-scale data processing, predictive analytics, scientific computing), that aims to simplify package management and deployment. Package versions are managed by the package management system conda. The Anaconda distribution is used by over 6 million users, and it includes more than 250 popular data science packages suitable for Windows, Linux, and MacOS.

Figure 10: Anaconda logo

Figure 11: Jupyter notebook logo

Jupyter Notebook: is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

MySQL Workbench is a unified visual tool for database architects, developers, and DBAs. MySQL Workbench provides data modeling, SQL development, and comprehensive administration tools for server configuration, user administration, backup, and much more. MySQL Workbench is available on Windows, Linux and Mac OS X.

Figure 12: Mysql workbench logo

Figure 13: MicroStrategy logo

MicroStrategy is a Business Intelligence software, which offers a wide range of data analytics capabilities. ... Features such as highly formatted reports, ad hoc query, thresholds and alerts, and automated report distribution makes **MicroStrategy** an industry leader in BI software space.

4.2 Materials

The DataU team provided me with this data source for me to do the project in a CSV file. The dataset contains 128258 entries in total with 47 features. Among these features, some features contain the information related to the loans, some related to borrowers information and a few columns about investors. Besides, there are 2 types of variables such as continuous and binary variables. The attributes are:

Variables	Description
addr_state	The state provided by the borrower in the loan application
annual_inc	The self-reported annual income provided by the borrower during registration.
collection_recovery_fee	post charge off collection fee

delinq_2yrs	The number of 30+ days past-due incidences of delinquency in the borrower's credit file for the past 2 years
dti	Debt-to-Income ratio. A ratio calculated using the borrower’s total monthly debt payments on the total debt obligations, excluding mortgage and the requested LC loan, divided by the borrower’s self-reported monthly income.
earliest_cr_line	The month the borrower's earliest reported credit line was opened
emp_length	Employment length in years. Possible values are between 0 and 10 where 0 means less than one year and 10 means ten or more years.
emp_title	The job title supplied by the Borrower when applying for the loan.*
fico_range_high	The upper boundary range the borrower’s FICO at loan origination belongs to.
fico_range_low	The lower boundary range the borrower’s FICO at loan origination belongs to.
funded_amnt	The total amount committed to that loan at that point in time.
funded_amnt_inv	The total amount committed by investors for that loan at that point in time.

grade	LC assigned loan grade
home_ownership	The home ownership status provided by the borrower during registration or obtained from the credit report. Our values are: RENT, OWN, MORTGAGE, OTHER
id	A unique LC assigned ID for the loan listing.
initial_list_status	The initial listing status of the loan. Possible values are – W (Whole), F (Fraction)
inq_last_6mths	The number of inquiries in past 6 months (excluding auto and mortgage inquiries)
installment	The monthly payment owed by the borrower if the loan originates.
int_rate	Interest Rate on the loan
issue_d	The month which the loan was funded
last_credit_pull_d	The most recent month LC pulled credit for this loan
last_pymnt_amnt	Last total payment amount received
last_pymnt_d	Last month payment was received
loan_amnt	The listed amount of the loan applied for by the borrower. If at some point in time, the credit department reduces the loan amount, then it will be reflected in this value.

loan_status	Current status of the loan
mths_since_last_delinq	The number of months since the borrower's last delinquency.
mths_since_last_record	The number of months since the last public record.
next_pymnt_d	Next scheduled payment date
open_acc	The number of open credit lines in the borrower's credit file.
out_prncp	Remaining outstanding principal for total amount funded
out_prncp_inv	Remaining outstanding principal for portion of total amount funded by investors
pub_rec	Number of derogatory public records
purpose	A category provided by the borrower for the loan request.
recoveries	post charge off gross recovery
revol_bal	Total credit revolving balance
revol_util	Revolving line utilization rate, or the amount of credit the borrower is using relative to all available revolving credit.
sub_grade	LC assigned loan subgrade
term	The number of payments on the loan. Values are in months and can be either 36 or 60.

title	The loan title provided by the borrower
total_acc	The total number of credit lines currently in the borrower's credit file
total_pymnt	Payments received to date for total amount funded
total_pymnt_inv	Payments received to date for portion of total amount funded by investors
total_rec_int	Interest received to date
total_rec_late_fee	Late fees received to date
total_rec_prncp	Principal received to date
verification_status	Indicates if income was verified by LC, not verified, or if the income source was verified
zip_code	The first 3 numbers provided by the borrower

Table 6: Variables

4.3 Methods

The graph below show us about methods that I will apply in order to complete the project.

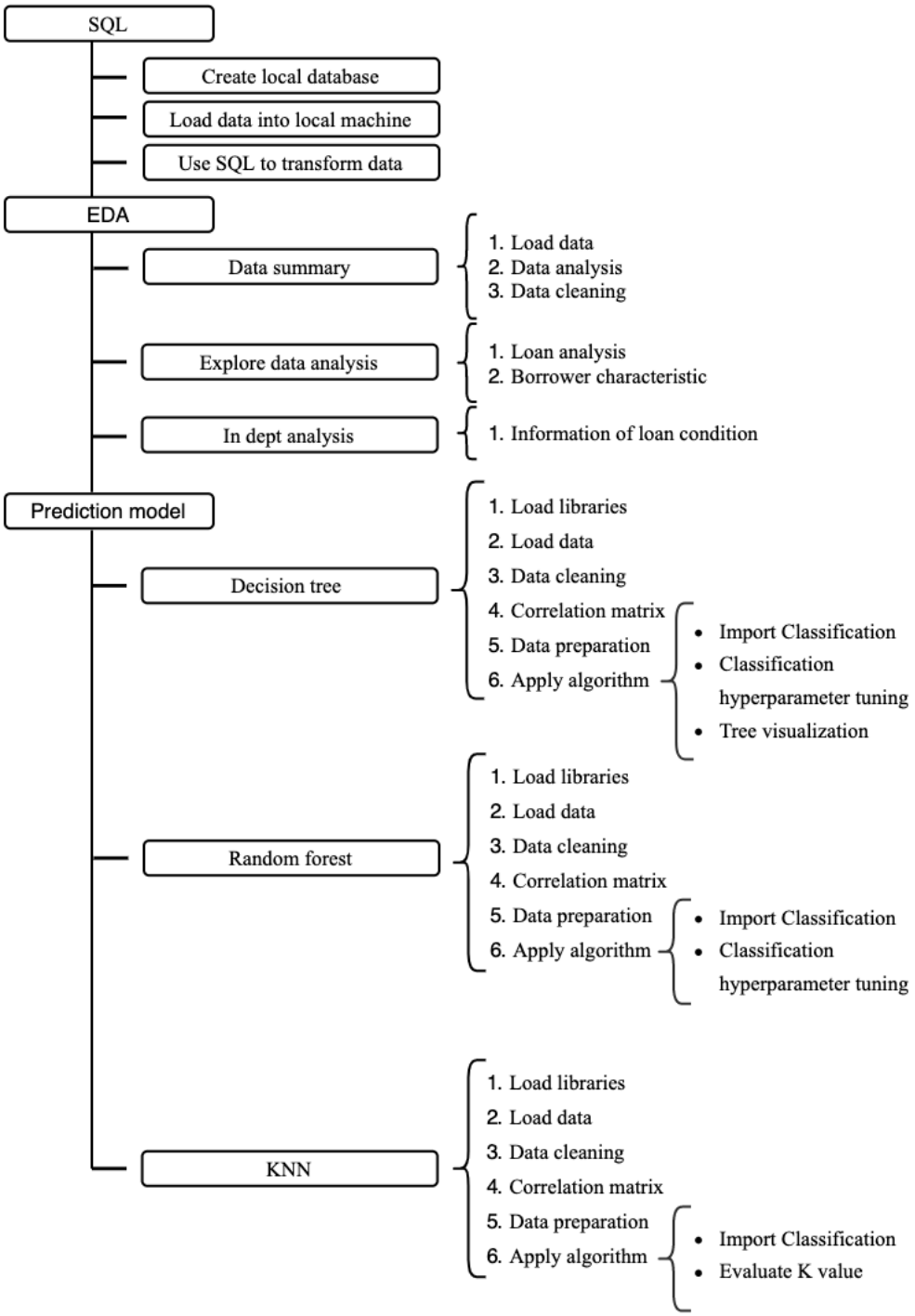


Figure 14: Method diagram

4.3.1 Project overview and setup

In this Module I will get an overview of the project that I have to complete by the end of the program. I am going to set up and upload my data into your own database in a local machine. I also have to produce a simple report by using SQL.

In this exercise, I will look at the csv file containing customer loan information and their features. As a junior data engineer, I am asked to transform the data, create a local database, and load it into your local machine. I need to produce an insight from the data by using SQL language. In addition, I will need to prepare a simple report by providing an overview of statistical detail of data by using SQL query. Key syntax to use:

SELECT, WHERE, GROUP BY, ORDER BY, HAVING, AND, OR, NOT, LIKE, NULL, BETWEEN, DISTINCT, IN
COUNT, SUM, MIN, MAX, AVG

4.3.2 Data preprocessing and cleaning

First of all, I load libraries into the project. There are matplotlib.pyplot for plot configuration, numpy for numerical operations, pandas for database management and seaborn for data visualization.

Second, I load the dataset into my project in order to start. I display the basic information by using the **.info()** function to see Range Index and column details. Then, I display statistical information using **.describe()** to see count, mean, std, min, 25%, 50%,75% and max value of the data. After that, I check if there is missing data and calculate its percentage. As I can see there are some features that have wrong data types, so I change those data types to the right one.

Lastly, I clean the features that have symbols, 'year' or letters which we cannot calculate because they are not numbers. Then, I drop unnecessary columns such as: funded_amnt ,title, zip_code and total_pymnt_inv.

4.3.3 Exploratory data analysis

In this Module, I will focus on exploring the data by using statistical analysis in combination. I also have to visualize your findings and to identify anomalies with python. In this part, we

will look into detail of the database containing P2P loan portfolios. As an analyst, I am asked to produce a statistical baseline and to identify the abnormality with python language. In addition, I will need to prepare a Jupyter report to show the value that you have extracted from the data.

First, I explain and provide comprehensive information on loan performance at LC. We will take a look at the number of loans issued by different dates and as well as the loan amount. Along with that, we're going to figure out loans' characteristics such as tracking the numbers of loans by term does the company provide and also the purpose that people take the loan from LC. Another important part is loans' grades. LC approves loans by looking at the borrowers grade and provides the different interest rate base on it. It sounds interesting to track the number of loans by grade and sub grades and also the average interest rate in each grade.

Second part, we will provide the borrowers' characteristics information of selecting profile factors across a multitude of loans. These profile factors are selected based on what a lender would typically consider when issuing loans and those factors like the length of employment, the debt-to-income ratio of an applicant, utilization of existing bank cards and their historical record of their previous loans and public records.

Next part, we will give you an in-depth analysis through the loan performance so far which related to how much money we issue the loan, how much we received from repaying the loan so far with the interest rate, the percentage of loan amount which are fully paid and charge off and see if there's any recoveries from charge-off loans. On the other hand, we will provide the comprehensive characteristics of default and charge off loans which are considered as the bad loans for the company to make a better prevention for current and the new loans.

Lastly, we will summarize all crucial keys-finding from our analytical process and give some recommendations which will be helpful for the company.

4.3.4 Predictive models

In this Module you will focus on making a predictive model based on supervised learning. Evaluate my model with different metrics and improve your model. As a ML specialist, I am asked to produce a prediction model from the data by using machine learning algorithms (Supervised learning) on classification.

A. Decision tree

Decision Tree is a **Supervised learning technique** that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where **internal nodes represent the features of a dataset, branches represent the decision rules** and **each leaf node represents the outcome.**

In a Decision tree, there are two nodes, which are the **Decision Node** and **Leaf Node.** Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

The decisions or the test are performed on the basis of features of the given dataset.

It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.

It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.

In order to build a tree, we use the **CART algorithm,** which stands for **Classification and Regression Tree algorithm.**

A decision tree simply asks a question, and based on the answer (Yes/No), it further splits the tree into subtrees.

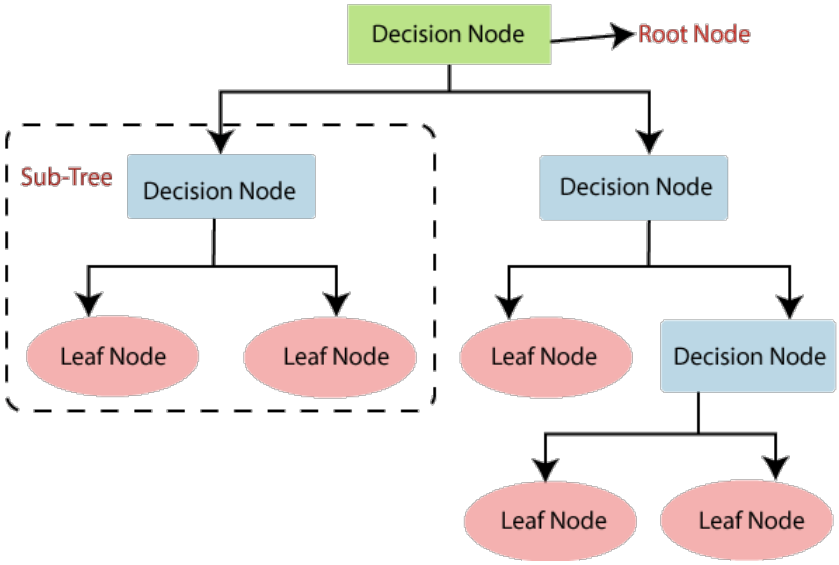


Figure 15: Decision tree algorithm

There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision tree:

- Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.
- The logic behind the decision tree can be easily understood because it shows a tree-like structure.

➢ **Decision Tree Terminologies are:**

- **Root Node:** Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
- **Leaf Node:** Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.

- **Splitting:** Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
- **Branch/Subtree:** A tree formed by splitting the tree.
- **Pruning:** Pruning is the process of removing the unwanted branches from the tree.
- **Parent/Child node:** The root node of the tree is called the parent node, and other nodes are called the child nodes.

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of the root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

For the next node, the algorithm again compares the attribute value with the other sub-nodes and moves further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

Step-1: Begin the tree with the root node, says S, which contains the complete dataset.

Step-2: Find the best attribute in the dataset using Attribute Selection Measure (ASM).

Step-3: Divide the S into subsets that contain possible values for the best attributes.

Step-4: Generate the decision tree node, which contains the best attribute.

Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3 to Continue this process until a stage is reached where you cannot further classify the nodes and call the final node as a leaf node.

⇒ **Attribute Selection Measures**

While implementing a Decision tree, the main issue arises how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called an **Attribute selection measure or ASM.** By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:

- Information Gain
- Gini Index

➢ **Information Gain:**

- Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.
- It calculates how much information a feature provides us about a class.
- According to the value of information gain, we split the node and build the decision tree.
- A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:

Information Gain= Entropy(S)- [(Weighted Avg)

***Entropy(each feature)**

Entropy: Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as:

$$\textbf{Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no)}$$

Where,

- **S**= Total number of samples
- **P(yes)**= probability of yes
- **P(no)**= probability of no

⇒ **Gini Index:**

- Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm.
- An attribute with the low Gini index should be preferred as compared to the high Gini index.
- It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.
- Gini index can be calculated using the below formula:

$$\textbf{Gini Index= 1- } \sum_j P_j^2$$

➢ **Advantages of the Decision Tree**

- It is simple to understand as it follows the same process which a human follows while making any decision in real-life.
- It can be very useful for solving decision-related problems.
- It helps to think about all the possible outcomes for a problem.
- There is less requirement of data cleaning compared to other algorithms.

- **Disadvantages of the Decision Tree**
  - The decision tree contains lots of layers, which makes it complex.
  - It may have an overfitting issue, which can be resolved using the **Random Forest algorithm.**
  - For more class labels, the computational complexity of the decision tree may increase.

B. Random forest

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning,** which is a process of *combining multiple classifiers to solve a complex problem and to improve the performance of the model.*

As the name suggests, ***"Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset."*** Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

The below diagram explains the working of the Random Forest algorithm:

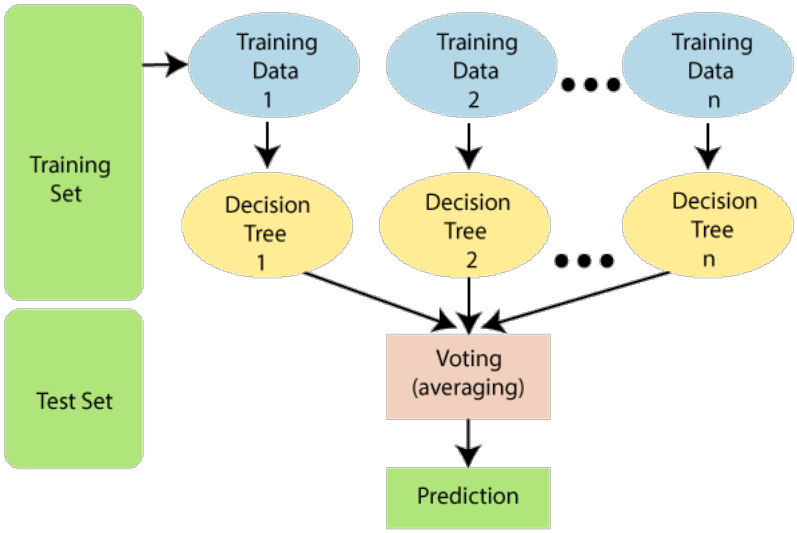


Figure 16: Random forest algorithm

How does Random Forest algorithm work?

Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

The Working process can be explained in the below steps and diagram:

Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

Step-3: Choose the number N for decision trees that you want to build.

Step-4: Repeat Step 1 & 2.

Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

- Applications of Random Forest

There are mainly four sectors where Random forest mostly used:

1. **Banking:** Banking sector mostly uses this algorithm for the identification of loan risk.
2. **Medicine:** With the help of this algorithm, disease trends and risks of the disease can be identified.
3. **Land Use:** We can identify the areas of similar land use by this algorithm.
4. **Marketing:** Marketing trends can be identified using this algorithm.

- Advantages of Random Forest
  - Random Forest is capable of performing both Classification and Regression tasks.
  - It is capable of handling large datasets with high dimensionality.
  - It enhances the accuracy of the model and prevents the overfitting issue.

- Disadvantages of Random Forest
  - Although random forest can be used for both classification and regression tasks, it is not more suitable for Regression tasks.

- Python Implementation of Random Forest Algorithm

Now we will implement the Random Forest Algorithm tree using Python. For this, we will use the same dataset "user_data.csv", which we have used in previous classification models. By using the same dataset, we can compare the Random Forest classifier with other classification models such as Decision tree Classifier, KNN, SVM, Logistic Regression, etc.

Implementation Steps are given below:

- Data Pre-processing step
- Fitting the Random forest algorithm to the Training set
- Predicting the test result
- Test accuracy of the result (Creation of Confusion matrix)
- Visualizing the test set result.

C. K-nearest neighbor

⇒ KNN is one of the simplest Machine Learning algorithms based on Supervised Learning technique.

⇒ The K-NN algorithm assumes the similarity between the new case/data and available cases and puts the new case into the category that is most similar to the available categories.

⇒ K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.

⇒ The K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.

⇒ K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.

⇒ It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.

The KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

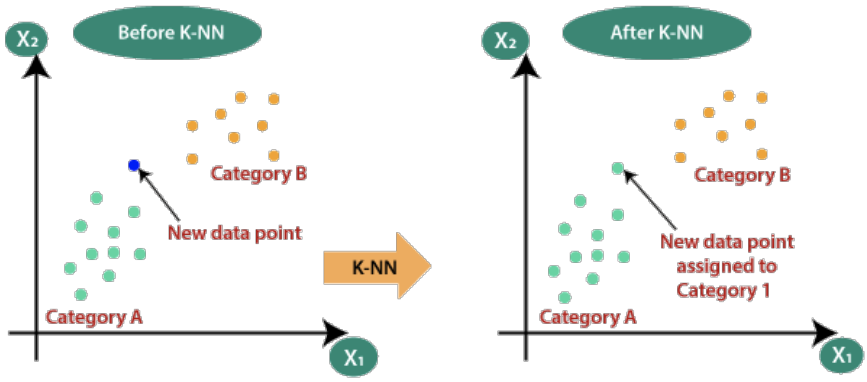


Figure 17: KNN training phase

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram :

The K-NN working can be explained on the basis of the below algorithm:

- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.

Suppose we have a new data point and we need to put it in the required category. Consider the below image:

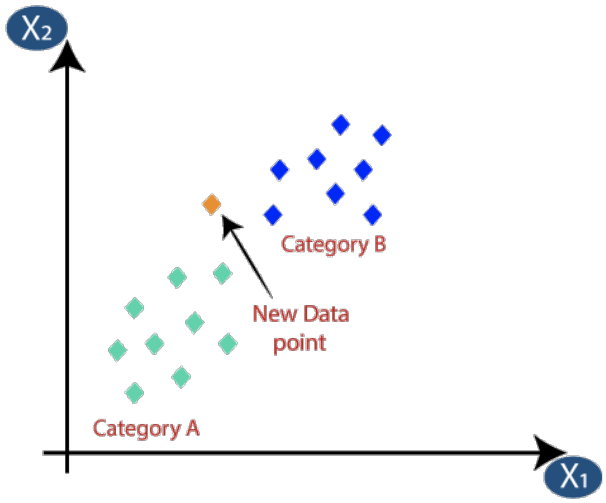


Figure 18:example of KNN

- Firstly, we will choose the number of neighbors, so we will choose the k=5.
- we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:

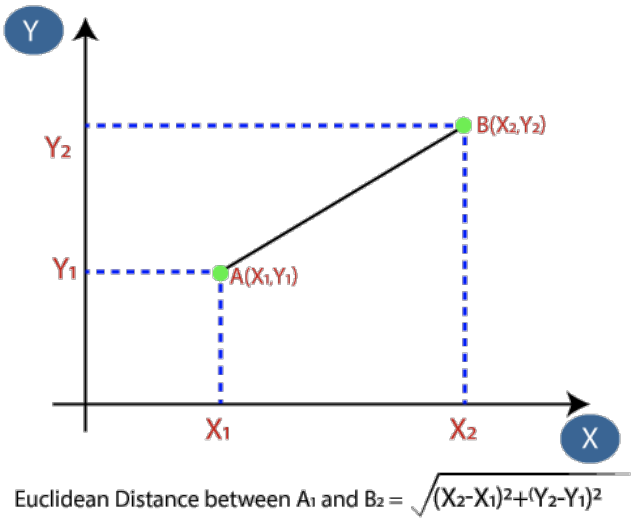


Figure 19: KNN Euclidean distance

- By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:

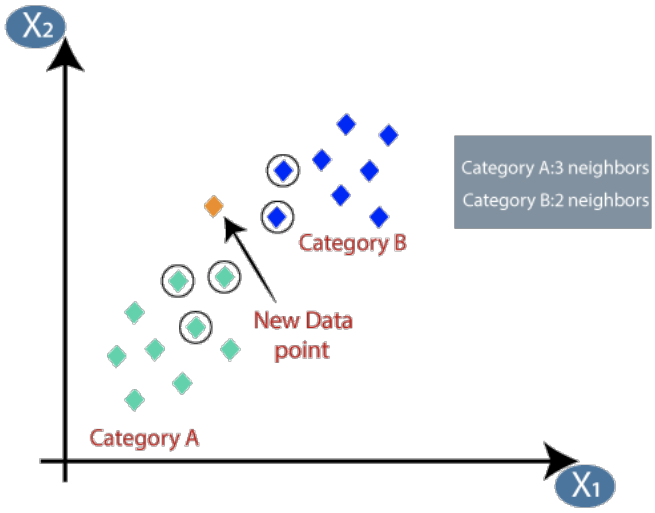


Figure 20: Nearest neighbors in category

- As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

❖ Below are some points to remember while selecting the value of K in the K-NN algorithm:

- There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.
- A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.
- Large values for K are good, but it may find some difficulties.

❖ Advantages of KNN Algorithm:

- It is simple to implement.
- It is robust to the noisy training data
- It can be more effective if the training data is large.

❖ Disadvantages of KNN Algorithm:

- Always needs to determine the value of K which may be complex some time.
- The computation cost is high because of calculating the distance between the data points for all the training samples.

❖ **Python implementation of the KNN algorithm:**

To do the Python implementation of the K-NN algorithm, we will use the same problem and dataset which we have used in Logistic Regression. But here we will improve the performance of the model. Below is the problem description:

Problem for K-NN Algorithm: There is a Car manufacturer company that has manufactured a new SUV car. The company wants to give the ads to the users who are interested in buying that SUV. So for this problem, we have a dataset that contains multiple user's information through the social network. The dataset contains lots of information but the **Estimated Salary** and **Age** we will consider for the independent variable and the **Purchased variable** is for the dependent variable.

❖ **Steps to implement the K-NN algorithm:**

- Data Preprocessing step
- Fitting the K-NN algorithm to the Training set
- Predicting the test result
- Test accuracy of the result(Creation of Confusion matrix)
- Visualizing the test set result

4.3.5 Hyperparameter tuning

A Machine Learning model is defined as a mathematical model with a number of parameters that need to be learned from the data. By

training a model with existing data, we are able to fit the model parameters.

However, there is another kind of parameter, known as ***Hyperparameters***, that cannot be directly learned from the regular training process. They are usually fixed before the actual training process begins. These parameters express important properties of the model such as its complexity or how fast it should learn.

Some examples of model hyperparameters include:

- The penalty in Logistic Regression Classifier i.e. L1 or L2 regularization
- The learning rate for training a neural network.
- The C and sigma hyperparameters for support vector machines.
- The k in k-nearest neighbors.

A. Decision tree

Decision trees are prone to overfitting. They can easily become over-complex which prevents them from generalizing well to the structure in the dataset. In that case, the model is likely to end up **overfitting** which is a serious issue in machine learning. To overcome this issue, we need to carefully adjust the hyperparameters of the Decision tree.

a. GridSearchCV

Grid search is a technique for tuning hyperparameters that may facilitate building a model and evaluate a model for every combination of algorithms parameters per grid.

We might use 10 fold cross-validation to search the best value for that tuning hyperparameter. Parameters like in decision criterion, max_depth, min_sample_split, etc. These values are called hyperparameters. To get the simplest set of hyperparameters we will use the Grid Search method. In the Grid Search, all the mixtures of hyperparameters combinations will pass through one by one into the model and check the score on each model. It gives us the set of hyperparameters which gives the best score. Scikit-learn package as

a means of automatically iterating over these hyperparameters using cross-validation. This method is called Grid Search**.**

Grid Search takes the model or objects you'd prefer to train and different values of the hyperparameters. It then calculates the error for various hyperparameter values, permitting you to choose the best values.

First, we import the libraries that we need, including GridSearchCV, the dictionary of parameter values. We create a decision tree object or model. We then create a GridSearchCV object. The inputs are the decision tree object, the parameter values, and the number of folds. We will use classification performance metrics. This is the default scoring method. We fit the object. We can find the best values for the parameters using the attribute best estimator. The advantages of Grid Search is how quickly we can test multiple parameters.

The decision is an estimator implemented using sklearn. The parameter cv is the cross-validation method if this parameter is set to be None, to use the default 5-fold cross-validation. The parameter verbose controls the verbosity and the parameter **n_jobs** is the number of jobs to run in parallel. none means 1 unless in a **joblib.parallel_backend** context. -1 means using all processors.
As you can see the image, There are some decision tree parameters:

- **criterion**: The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.
- **splitter**: The strategy used to choose the split at each node. Supported strategies are "best" to choose the best split and "random" to choose the best random split.
- **max_depth**: The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.
- **min_samples_split**: The minimum number of samples required to split an internal node.

- **min_samples_leaf**: The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least min_samples_leaf training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression.
- **min_weight_fraction_leaf**: The minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node. Samples have equal weight when sample_weight is not provided.
- **max_features**: The number of features to consider when looking for the best split.
- **max_leaf_nodes**: Grow a tree with max_leaf_nodes in best-first fashion. Best nodes are defined as relative reduction in impurity. If None then unlimited number of leaf nodes.
- **min_impurity_decrease**: A node will be split if this split induces a decrease of the impurity greater than or equal to this value.
- **min_impurity_split**: Threshold for early stopping in tree growth. A node will split if its impurity is above the threshold, otherwise it is a leaf.

```
GridSearchCV(cv=3, error_score=nan,
             estimator=DecisionTreeClassifier(ccp_alpha=0.0, 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='deprecated',
                                              random_state=42,
                                              splitter='best'),
             iid='deprecated', n_jobs=-1,
             param_grid={'criterion': ('gini', 'entropy'),
                         'max_depth': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,
                                       13, 14, 15, 16, 17, 18, 19],
                         'min_samples_leaf': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
                                              12, 13, 14, 15, 16, 17, 18, 19],
                         'min_samples_split': [2, 3, 4],
                         'splitter': ('best', 'random')},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring='accuracy', verbose=1)
```

Figure 21: Grid search CV

B. Random forest

Learning the parameters of a prediction function and testing it on the same data is a methodological mistake: a model that would just repeat the labels of the samples that it has just seen would have a perfect score but would fail to predict anything useful on yet-unseen data. This situation is called overfitting. To avoid it, it is common practice when performing a (supervised) machine learning experiment to hold out part of the available data as a test set **X_test**, **y_test**. Note that the word "experiment" is not intended to denote academic use only, because even in commercial settings machine learning usually starts out experimentally. Here is a flowchart of typical cross validation workflow in model training. The best parameters can be determined by grid search techniques.

a. Random search cross validation

RandomizedSearchCV implements a "fit" and a "score" method. It also implements "predict", "predict_proba", "decision_function", "transform" and "inverse_transform" if they are implemented in the estimator used.

The parameters of the estimator used to apply these methods are optimized by cross-validated search over parameter settings.

In contrast to GridSearchCV, not all parameter values are tried out, but rather a fixed number of parameter settings is sampled from the specified distributions. The number of parameter settings that are tried is given by n_iter.

If all parameters are presented as a list, sampling without replacement is performed. If at least one parameter is given as a distribution, sampling with

replacement is used. It is highly recommended to use continuous distributions for continuous parameters.

If **n_jobs** was set to a value higher than one, the data is copied for each parameter setting(and not **n_jobs** times). This is done for efficiency reasons if individual jobs take very little time, but may raise errors if the dataset is large and not enough memory is available. A workaround in this case is to set pre_dispatch. Then, the memory is copied only pre_dispatch many times. A reasonable value for pre_dispatch is **2 * n_jobs.**

```
RandomizedSearchCV(cv=3, error_score=nan,
                   estimator=RandomForestClassifier(bootstrap=True,
                                                    ccp_alpha=0.0,
                                                    class_weight=None,
                                                    criterion='gini',
                                                    max_depth=None,
                                                    max_features='auto',
                                                    max_leaf_nodes=None,
                                                    max_samples=None,
                                                    min_impurity_decrease=0.0,
                                                    min_impurity_split=None,
                                                    min_samples_leaf=1,
                                                    min_samples_split=2,
                                                    min_weight_fraction_leaf=0.0,
                                                    n_estimators=100,
                                                    n_jobs...
                   param_distributions={'bootstrap': [True, False],
                                        'max_depth': [10, 20, 30, 40, 50, 60,
                                                      70, 80, 90, 100, 110,
                                                      None],
                                        'max_features': ['auto', 'sqrt'],
                                        'min_samples_leaf': [1, 2, 4],
                                        'min_samples_split': [2, 5, 10],
                                        'n_estimators': [200, 400, 600, 800,
                                                         1000, 1200, 1400, 1600,
                                                         1800, 2000]},
                   pre_dispatch='2*n_jobs', random_state=42, refit=True,
                   return_train_score=False, scoring=None, verbose=2)
```

Figure 22: Random search CV

b. Cross validation

When evaluating different settings ("hyperparameters") for estimators, such as the **C** setting that must be manually set for an SVM, there is still a risk of overfitting *on the test set* because the parameters can be tweaked until the estimator performs optimally. This way, knowledge about the test set can "leak" into the model and evaluation metrics no longer report on

generalization performance. To solve this problem, yet another part of the dataset can be held out as a so-called "validation set": training proceeds on the training set, after which evaluation is done on the validation set, and when the experiment seems to be successful, final evaluation can be done on the test set.

However, by partitioning the available data into three sets, we drastically reduce the number of samples which can be used for learning the model, and the results can depend on a particular random choice for the pair of (train, validation) sets.

A solution to this problem is a procedure called **cross-validation** (CV for short). A test set should still be held out for final evaluation, but the validation set is no longer needed when doing CV. In the basic approach, called *k*-fold CV, the training set is split into *k* smaller sets (other approaches are described below, but generally follow the same principles). The following procedure is followed for each of the *k* "folds":

- A model is trained using
- `k-1` of the folds as training data;
- The resulting model is validated on the remaining part of the data (i.e., it is used as a test set to compute a performance measure such as accuracy).

The performance measure reported by *k*-fold cross-validation is then the average of the values computed in the loop. This approach can be computationally expensive, but does not waste too much data (as is the case when fixing an arbitrary validation set), which is a major advantage in problems such as

inverse inference where the number of samples is very small.

As you can see in the figure, There are some algorithm parameters:

- **n_estimators**: The number of trees in the forest.
- **criterion**: The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.
- **max_depth**: The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.
- **min_samples_split**: The minimum number of samples required to split an internal node.
- **min_samples_leaf**: The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least min_samples_leaf training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression.
- **min_weight_fraction_leaf**: The minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node. Samples have equal weight when sample_weight is not provided.
- **max_features**: The number of features to consider when looking for the best split.
- **max_leaf_nodes**: Grow a tree with max_leaf_nodes in best-first fashion. Best nodes are defined as relative reduction in impurity. If None then unlimited number of leaf nodes.

- **min_impurity_decrease**: A node will be split if this split induces a decrease of the impurity greater than or equal to this value.
- **min_impurity_split**: Threshold for early stopping in tree growth. A node will split if its impurity is above the threshold, otherwise it is a leaf.
- **bootstrap**: Whether bootstrap samples are used when building trees. If False, the whole dataset is used to build each tree.
- **oob_score**: Whether to use out-of-bag samples to estimate the generalization accuracy.

c. K-nearest neighbor

Different from Decision tree and Random forest, the method that we use to do hyperparameter tuning is choose the K value. The determination of the K value varies greatly depending on the case. If using the **Scikit-Learn** Library the default value of K is 5.

To calculate distances, 3 distance metrics that are often used are **Euclidean** Distance, **Manhattan** Distance, and **Minkowski** Distance. When you use Scikit-Learn, the default distance used is Euclidean. It can be seen in the Minkowski distance formula that there is a Hyperparameter p, if set p = 1 then it will use the Manhattan distance and p = 2 to be Euclidean. After calculating the distance, then look for K-Neighbors that are closest to the new data. If using K = 3, look for 3 training data that is closest to the new data. Calculate the **accuracy** of the model, if the accuracy is still low, then this process can be repeated again from step 1.

KNN algorithm parameters:

- **n_neighbors (*int, default=5):*** Number of neighbors to use by default for **kneighbors** queries.

- **metric (*str) or callable, default='minkowski':*** the distance metric to use for the tree. The default metric is minkowski, and with p=2 is equivalent to the standard Euclidean metric. See the documentation of **DistanceMetric** for a list of available metrics. If the metric is "precomputed", X is assumed to be a distance matrix and must be square during fit. X may be a sparse graph, in which case only "nonzero" elements may be considered neighbors.
- **metric_params*dict (default=None) :*** Additional keyword arguments for the metric function.
- **leaf_size (*int ,default=30):*** Leaf size passed to BallTree or KDTree. This can affect the speed of the construction and query, as well as the memory required to store the tree. The optimal value depends on the nature of the problem.
- **p (*int, default=2) :*** Power parameter for the Minkowski metric. When p = 1, this is equivalent to using manhattan_distance (l1), and euclidean_distance (l2) for p = 2. For arbitrary p, minkowski_distance (l_p) is used.
- **n_jobs (*int, default=None) :*** The number of parallel jobs to run for neighbors search. None means 1 unless in a **joblib.parallel_backend** context. -1 means using all processors. See Glossary for more details. Doesn't affect **fit** method.

5. Conclusion

5.1 Result

A. Exploratory data analysis

In this part, we will look at overall information about the loan such as the number of the loans the company issued by dates, the amount of money, loan's term, purpose of taking loans and so on. This will help us to get a better understanding of the loan's characteristics and its performance so far.

The total number of loans is 121796 loans which were issued in the last quarter of 2019 (October, November, December). Among these 3 months, loans were issued in October much more than other 2 months, which represent 36% of the total loans with an amount around 713 million dollars. The minimum and maximum amount are the same which starts from $1,000 to $40,00 and the averages are quite the same as well **(figure 23 & figure 24).** It seems the majority of investors are interested in investing in these ranges because the amount requested by borrowers seems to match well with the investors when we look into the initial list status. There's only 5% that one loan was provided by multiple investors. **(figure 25)**

	issue_d	Nb_loan	Percentage	loan_amount	% loan amount	Min_amount	Max_amount	Avg_amount
0	Dec-19	39962	0.33	639128300	0.32	1000	40000	15993.40
1	Nov-19	38212	0.31	616138825	0.31	1000	40000	16124.22
2	Oct-19	43622	0.36	713391725	0.36	1000	40000	16353.94

Figure 23:Loan by date

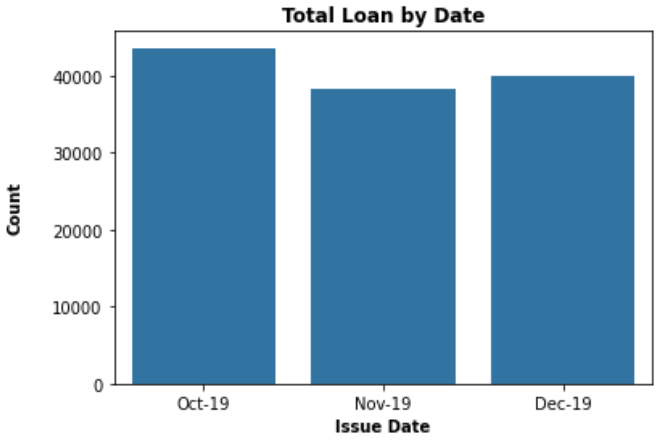


Figure 24: Total loan by date

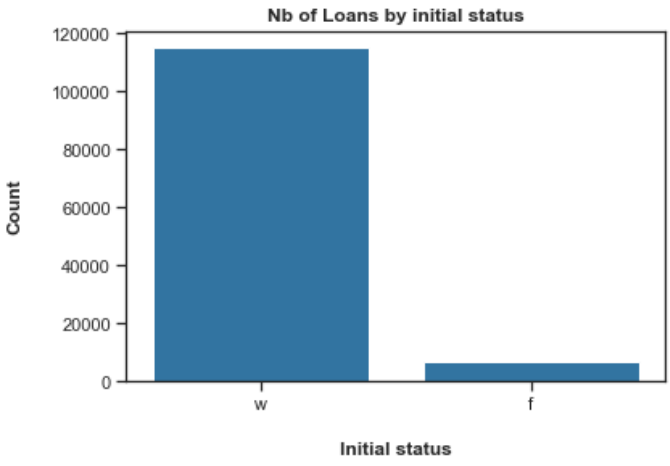


Figure 25: Number of loans by initial status

The loans that the company issued the last quarter of 2019 contains 4 grades which are A, B, C, and D and the interest rate varies from one to another. On

the other hand, there are 5 sub-grades inside each grade which represent the level of grade. Most of the loans that LC approves are in grade A which is the best grade among all grades and the vast majority of grade A choose a 3 years term of payment while there isn't much borrower of grade A choose a long term one. **(figure 26)**

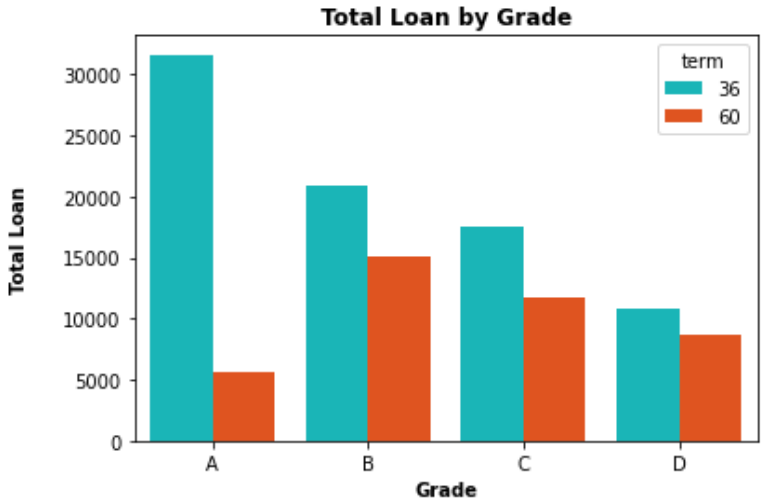


Figure 26: Total loan by Grade

The findings indicate that the vast majority of the borrowers have a medium annual income and most of them are in 3years term of payment **(figure27 & 28).** Furthermore, we notice that most of the medium income people are the good borrowers because they are in grade A and B and start to reduce in C and D **(figure 29).** On top of that, we see that they mortgage and rent the house for their property. **(figure13)**

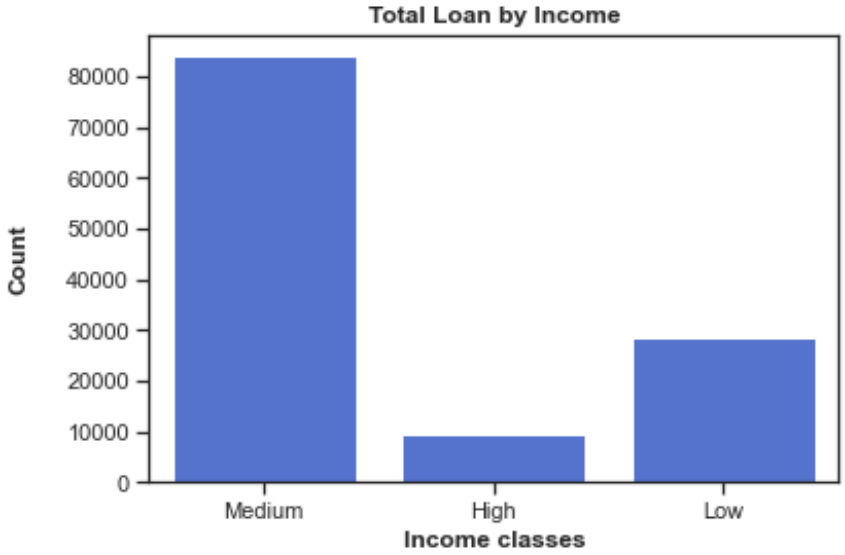


Figure 27: Total loan by income

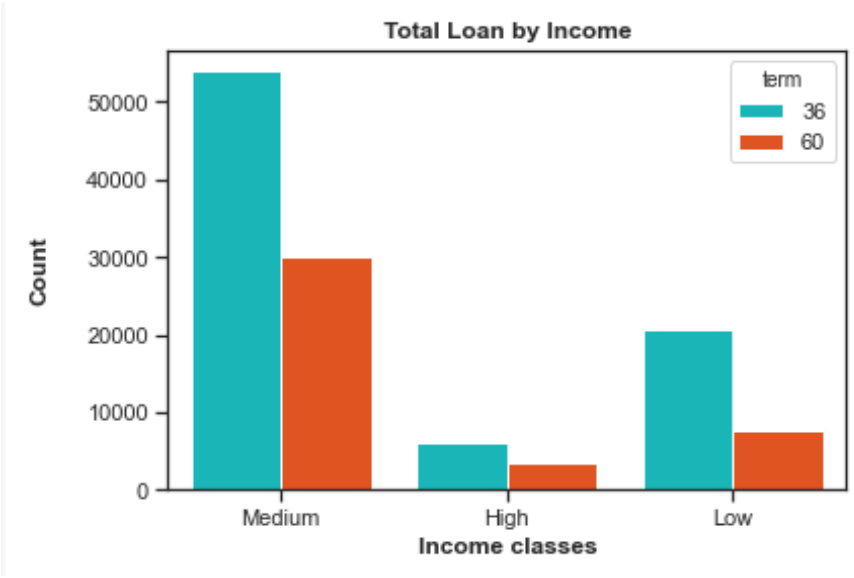


Figure 28: Total loan by income

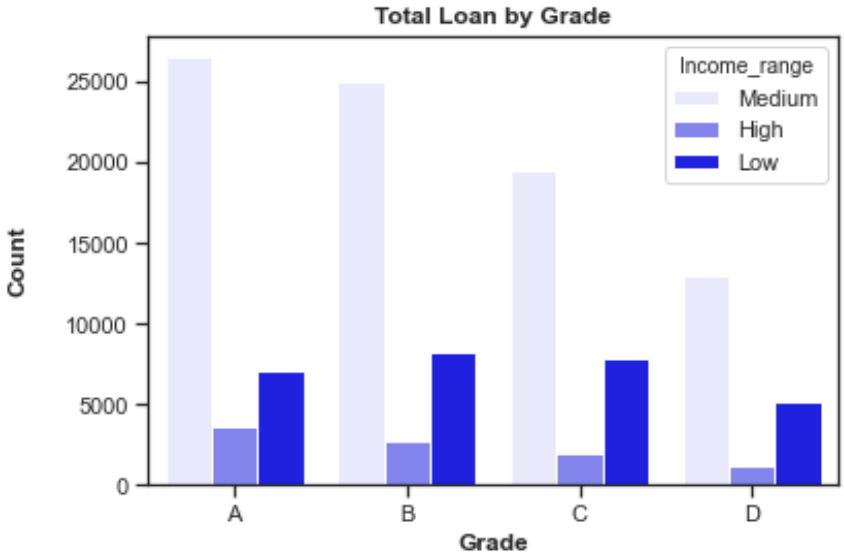


Figure 29:Total loan by grade

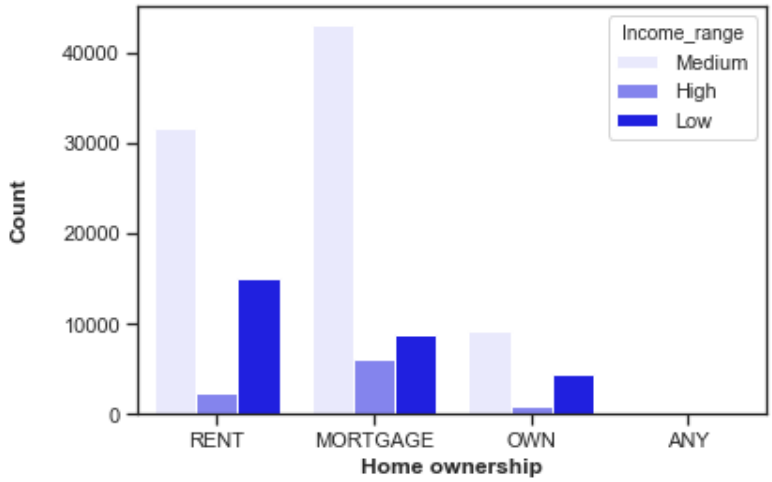


Figure 30: Total home ownership

After checking the loan status, it indicates that the vast majority of loans are still paying with the company, some have been fully paid, some are late and an unwanted point to see is that some are charged off which means they fail to pay back their loans. **(figure14)**

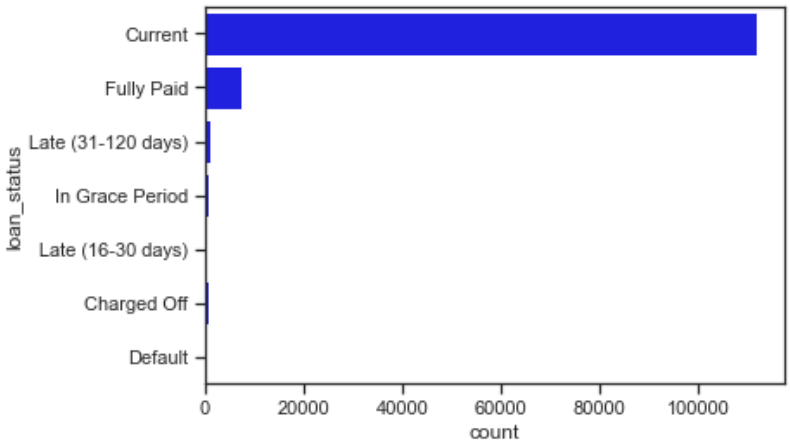


Figure 31: Number of loan by loan status

There are multiple loan statuses which are a bit hard to get the specific characteristics of each status. Thus, we decided to group them into 3 categories such as good, caution and bad loans. We group the current loan and fully paid as the good loans, late and in grace period as caution and default and charged off as the bad loan. After grouping, the good loans represent 97.7% of the whole data while caution is 1.73% and 0.56% for bad loans as we show in the pie chart below **(figure 32).** We will focus on the bad loan which affects the company's revenues. There are 670 bad loans which represent less than 1%. At first glance, we may think it's not a big matter since it's a small number, yet if we take a look at the loan amount that company issued with those loans, it's around 11 millions dollars. Thus, that's what the company has to pay attention to. To deal with this, we will provide comprehensive information about its characteristics in order to give clues to the company to have a better prevention toward the current loans.

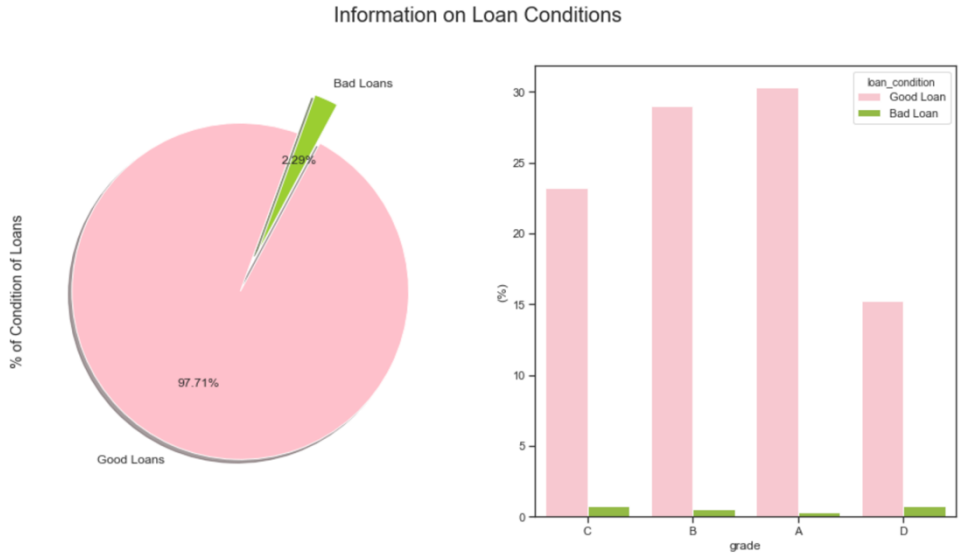


Figure 32: Information of loan conditions

As you can see in the (**figure 33**), Good loan with low interest rate have more than 70000 and good loan with high interest rate have around 50000. Moreover, there are small amount of bad loan with low and high interest rate.

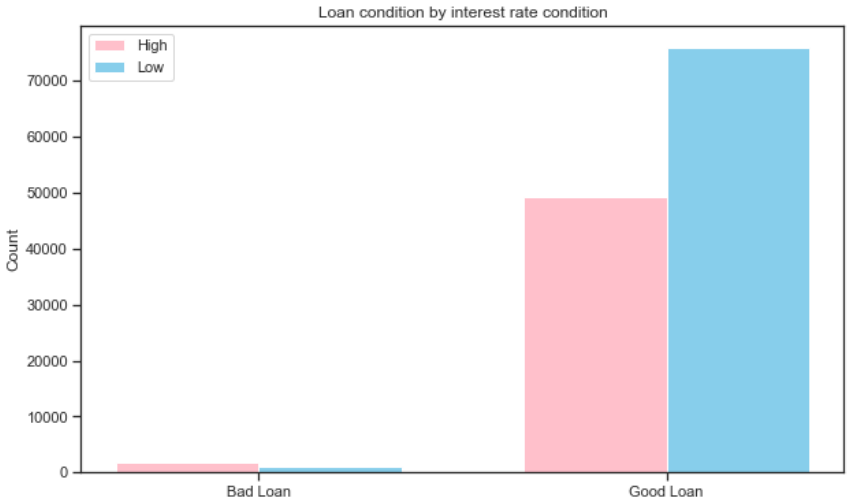


Figure 33: Loan condition by interest rate condition

B. Decision Tree

➢ Baseline Result with Default Parameters

The decision tree obtained for the "Lending Club" dataset has the attribute "Loan status" that I identify them into “bad loan” and “good loan” by alias those “In grace period”, “Late (16-30 days)”, “Late (31-120 days)” to “Default” and “current” to “fully paid”. According to the result, the accuracy of the model is 63%. It works not really well with the prediction of “Default” loans. However, it does not predict accurately with the “Fully paid” loans. This may be the result of the imbalance dataset we had.

	precision	recall	f1-score	support
0	0.33	0.34	0.33	1192
1	0.74	0.74	0.74	3119
accuracy			0.63	4311
macro avg	0.54	0.54	0.54	4311
weighted avg	0.63	0.63	0.63	4311

Figure 34: Decision tree before Tuning

➢ Baseline Result with Hyperparameter Tuning (GridSearch CV)

Then, I perform parameter tuning using Grid Search CV and we have found that there is a slight improvement in the precision of "Default" response but a decrease in the recall, making f1-score lower. Hence, this tuning still do not make a significant improvement to the model yet.

	precision	recall	f1-score	support
0	0.44	0.13	0.20	1192
1	0.74	0.94	0.83	3119
accuracy			0.71	4311
macro avg	0.59	0.53	0.51	4311
weighted avg	0.65	0.71	0.65	4311

Figure 35: Decision tree result after tuning

C. Random Forest

- **Baseline Result with Default Parameters**

Random forest helps in overcoming the over-fitting problem experienced in Decision Tree classification. It provides a significant increase in the accuracy of the model. The trees generated had "Loan status" as the primary split just like Decision Tree but the remaining attributes varied due to randomness in the subsets. This is the result from the random forest algorithm with default parameters. The accuracy is 71%, slightly the same as decision tree due to the number of trees generated from this algorithm to prevent overfitting.

	precision	recall	f1-score	support
0	0.38	0.09	0.15	1192
1	0.73	0.94	0.82	3119
accuracy			0.71	4311
macro avg	0.56	0.52	0.49	4311
weighted avg	0.64	0.71	0.64	4311

Figure 37: Random forest result before tuning

- **Baseline Result with Hyperparameter Tuning (GridSearch CV)**

Then, we perform the Grid Search CV and we got a similar result, yet a slight improvement to the recall and f1-score, not really significant at all.

	precision	recall	f1-score	support
0	0.44	0.15	0.23	1192
1	0.74	0.93	0.82	3119
accuracy			0.71	4311
macro avg	0.59	0.54	0.53	4311
weighted avg	0.66	0.71	0.66	4311

Figure 39: Random forest result after GridSearch tuning

- **Baseline Result with Hyperparameter Tuning (Randomized Search CV)**

Next, we conduct the Random Search CV and we also got the similar result. Hence, we can see that this is the best result we can generate from random forest.

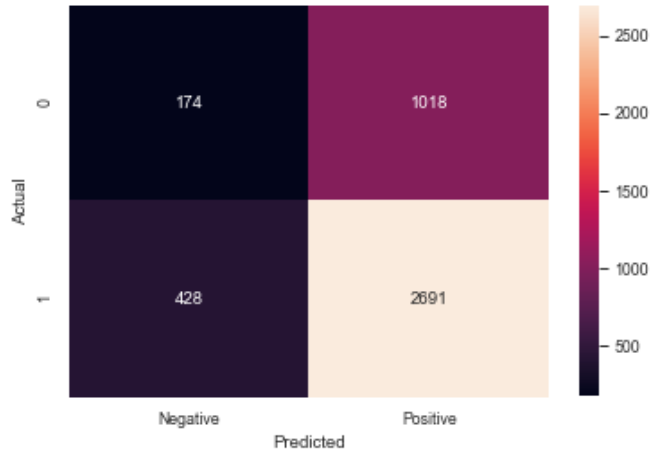
	precision	recall	f1-score	support
0	0.46	0.12	0.19	1192
1	0.74	0.95	0.83	3119
accuracy			0.72	4311
macro avg	0.60	0.53	0.51	4311
weighted avg	0.66	0.72	0.65	4311

Figure 41: Random forest result after Random Search tuning

D. K-Nearest Neighbor (KNN)

- **Baseline Result with Default Parameter**

K-nearest neighbor also had “Loan status” as the primary split just like Decision tree and Random forest. This is the result of KNN algorithm with default parameter, selecting 5 as a number of neighbor for validation. I found the accuracy is 0.66, quite low compare to other algorithm.



	precision	recall	f1-score	support
0	0.29	0.15	0.19	1192
1	0.73	0.86	0.79	3119
accuracy			0.66	4311
macro avg	0.51	0.50	0.49	4311
weighted avg	0.60	0.66	0.62	4311

Figure 43: KNN result before tuning

Figure 44: KNN result before tuning

- **Baseline with Parameter Tuning (Find K number)**

Then, we perform parameter tuning by selecting a range of the k numbers of neighbors for the testing to find the value of k which gives the best result. With the implementation of 5-fold CV, we have found that k=35 provides the best result.

	precision	recall	f1-score	support
0	0.51	0.02	0.03	1192
1	0.73	0.99	0.84	3119
accuracy			0.72	4311
macro avg	0.62	0.51	0.43	4311
weighted avg	0.67	0.72	0.62	4311

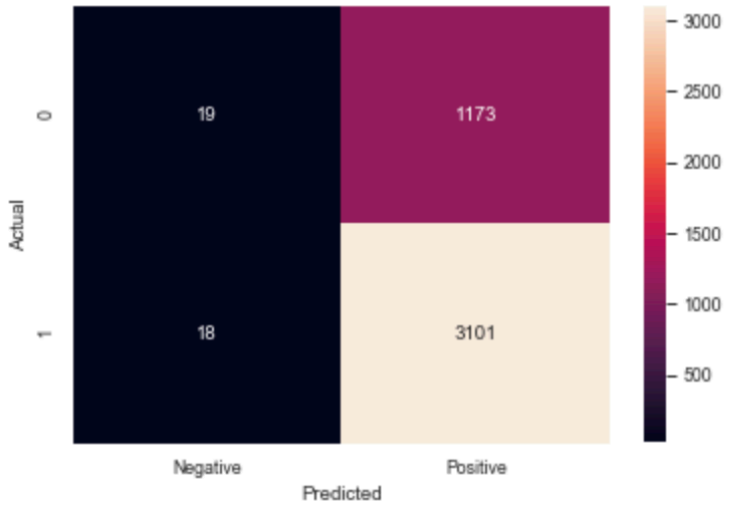


Figure 47: KNN result after tuning

Figure 48: KNN result after tuning

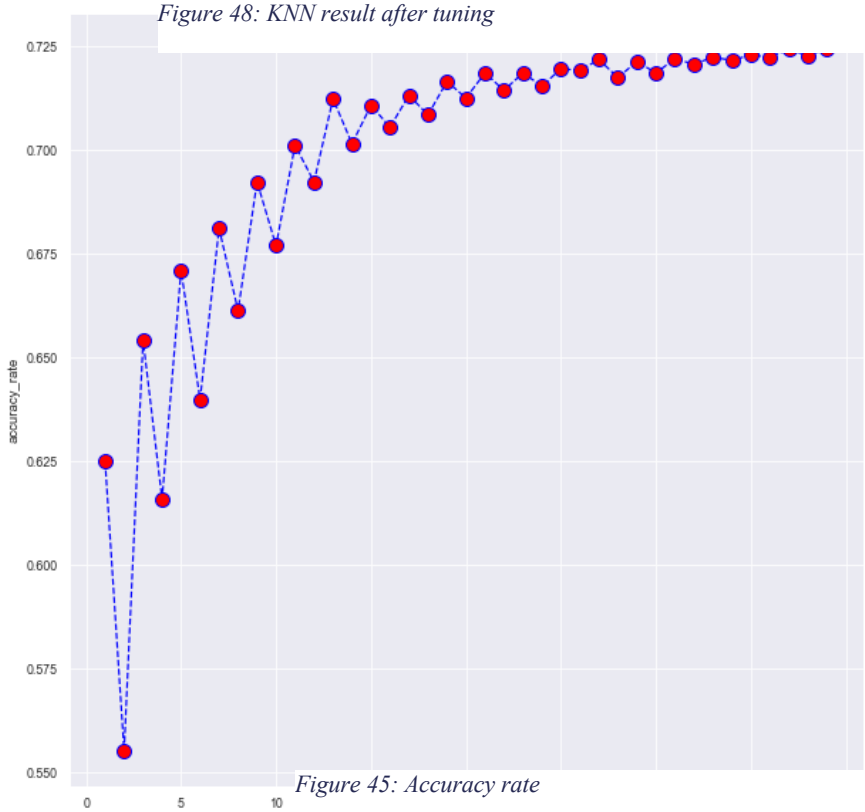


Figure 45: Accuracy rate

Figure 46: Accuracy rate

5.2 Strong Point And Weak Point Of Data Quality

5.2.1 Strong Point

- The company can decide to provide the loan based on the grade they assigned to the borrowers and the interest rate will vary from each grade.
- The Company can determine who should give them a loan and avoid who has a bad background.
- The investor won't waste money and borrowers will get a lower interest rate so that they can pay back.

5.2.2 Weak Point

- This project cannot drop some missing data, those features are good because it has to be blank.

5.3 Difficulties

Life is a struggle, born as a human you have to face through thick and thin to success. All along this internship I have challenged myself both technical and soft skills.

We have to do research and solve all the problems by ourselves with critical thinking and cleverness. Adapt to the environment of the company and learn new technology are what we have to do. Understanding algorithms of each machine learning model is the second obstacle that I must manage time to overcome by starting learning and asking from both google and seniors. To be honest, sometimes I really want to give up when I have no idea how to deal with the obstacle. But, life is a journey, I have to fight no matter what and finally, I am here.

5.4 Experiences

After doing this internship, I have learnt both soft skill and hard skill. I have improved my communication skill by communicating with other co-worker in the company and I have improved my analytical skill. Furthermore, I have learnt how to work as a team and how to work in pressure. In addition, it made me know that I still need to improve my programming skill and solve problems quickly.

5.5 Future Perspective

After do this project, In the future I would like to build an application or website that apply this analysis prediction model in order to make bank or lending club can track customer more efficiently.

5.6 Conclusion

After finishing my internship, I have learnt a lot about how to analyze data and insight data. In addition, I experienced working in a real environment and I feel employees that I have never experienced before. I feel so excited that I could do such an internship. Finally, I want to thank my advisor and everyone that supports and helps to complete this internship.

6. Reference

[I] T. Van Gestei, B. Baesens, Credit Risk Management: Basic Concepts: Financial Risk Components, Rating Analysis, Models, Economic and Regulatory Capital: Oxford University Press, 2008.

[2] 1.R. Quinlan, "Induction of Decision Tree", Machine Learning Journal, Volume I,Issue I,Pages 81-106,1986.

[3] L. Breiman, "Random Forest", Machine Learning Journal, Volume 45,Tssue I,Pages 5-32,2001.

[4] L. Breiman, "Bagging Predictors", Machine Learning Journal, Volume 24,Tssue 2,Pages 123-140,1996.

[5] P. Geurts, D. Ernst., and L. Wehenkel, "Extremely randomized trees", Machine Learning Journal, Volume 63,Issue I,Pages 3-42,2006.

[6] K. Tsai, S.Ramiah, S. Singh, "Peer Lending Risk Predictor", Stanford University, 2014.

[7] F.N Koutanaei, H. Sajedi, M. Khanbabaei, "A hybrid data mining model of feature selection algorithms and ensemble learning classifiers for credit scoring", Journal of Retailing and Consumer Services 27,Pages 11-23,2015.

[8] G. Wang, 1. Mac, A hybrid ensemble approach for enterprise credit risk assessment based on Support Vector Machine, Expert Systems with Applications 39 (2012) 5325-5331,2012.

[9] T. Fawcett, An introduction to ROC analysis, Pattern Recognition Letters 27,2006.

[10] E. Angelini, G. di Tollo, A. Roli, A Neural Network Approach for Credit Risk Evaluation, The Quarterly Review of Economics and Finance 48(4): Pages: 733-755,2008

https://scikit-learn.org/stable/modules/cross_validation.html

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

https://towardsdatascience.com/hyperparameters-of-decision-trees-explained-with-visualizations-1a6ef2f67edf

https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

7. Appendix

Appendix 1: Decision tree