

## **CREDIT DEFAULT RISK PREDICTION**

### **Abstract:**

Abstract: Credit risk plays a major role in the banking industry business. Banks' main activities involve granting loan, credit card, investment, mortgage, and others. Credit card has been one of the most booming financial services by banks over the past years. However, with the growing number of credit card users, banks have been facing an escalating credit card default rate. As such data analytics can provide solutions to tackle the current phenomenon and management credit risks. This paper provides a performance evaluation of credit card default prediction. Since the global financial crisis, risk management in banks has gained more prominence, and there has been a constant focus around how risks are being detected, measured, reported and managed. Considerable research in academia and industry has focused on the developments in banking and risk management and the current and emerging challenges. This paper, through a review of the available literature seeks to analyse and evaluate machine-learning techniques that have been researched in the context of banking risk management, and to identify areas or problems in risk management that have been inadequately explored and are potential areas for further research. The review has shown that the application of machine learning in the management of banking risks such as credit risk, market risk, operational risk and liquidity risk has been explored; however, it doesn't appear commensurate with the current industry level of focus on both risk management and machine learning. A large number of areas remain in bank risk management that could significantly benefit from the study of how machine learning can be applied to address specific problem.

## 1. Introduction

Machine learning (ML) is the study of computer algorithms that improve automatically through experience. It is seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications, such as email filtering and computer vision, where it is difficult or infeasible to develop conventional algorithms to perform the needed tasks.

Machine learning focuses on the development of computer programs that can access data and use it learn for themselves.

The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide.

Machine learning is a method of teaching computers to parse data, learn from it, and then make a determination or prediction regarding new data. Rather than hand-coding a specific set of instructions to accomplish a particular task, the machine is "trained" using large amounts of data and algorithms to learn how to perform the task. Machine learning overlaps with its lower-profile sister field, statistical learning. Both attempt to find and learn from patterns and trends within large datasets to make predictions. The machine learning field has a long tradition of development, but recent improvements in data storage and computing power have made them ubiquitous across many different fields and applications, many of which are very commonplace. Apple's Siri, Facebook feeds, and Netflix movie recommendations all rely upon some form of machine learning. One of the earliest uses of machine learning was within credit risk modeling, whose goal is to use financial data to predict default risk.

When a business applies for a loan, the lender must evaluate whether the business can reliably repay the loan principal and interest. Lenders commonly use measures of profitability and leverage to assess credit risk. A profitable firm generates enough cash to cover interest expense and principal due. However, a more-leveraged firm has less equity available to weather economic shocks. Given two loan applicants – one with high profitability and high leverage, and the other with low profitability and low leverage – which firm has lower credit risk? The complexity of answering this question multiplies when banks incorporate the many other dimensions they examine during credit risk assessment. These additional dimensions typically include other financial information such as

liquidity ratio, or behavioral information such as loan/trade credit payment behavior. Summarizing all of these various dimensions into one score is challenging, but machine learning techniques help achieve this goal.

### 1.1 Supervised Learning:

Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs. It infers a function from labeled training data consisting of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal).

A supervised learning algorithm analyzes the **training data** and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way.

**Training set:** It is the one on which we train and fit our model basically to fit the parameters. Training data's output is available to model.

The majority of practical machine learning uses supervised learning.

Supervised learning is where you have input variables (x) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output.

$$Y = f(X)$$

The goal is to approximate the mapping function so well that when you have new input data (x) that you can predict the output variables (Y) for that data.

It is called supervised learning because the process of an algorithm learning from the training dataset can be thought of as a teacher supervising the learning process. We know the correct answers, the algorithm iteratively makes predictions on the training data and is corrected by the teacher. Learning stops when the algorithm achieves an acceptable level of performance.

### Steps to Perform Supervised Learning:

- In order to solve a given problem of supervised learning, one has to perform the following steps:

- Determine the type of training examples. Before doing anything else, the user should decide what kind of data is to be used as a training set. In the case of handwriting analysis, for example, this might be a single handwritten character, an entire handwritten word, or an entire line of handwriting.
- Gather a training set. The training set needs to be representative of the real-world use of the function. Thus, a set of input objects is gathered and corresponding outputs are also gathered, either from human experts or from measurements.
- Determine the input feature representation of the learned function. The accuracy of the learned function depends strongly on how the input object is represented. Typically, the input object is transformed into a feature vector, which contains a number of features that are descriptive of the object. The number of features should not be too large, because of the curse of dimensionality; but should contain enough information to accurately predict the output.
- Determine the structure of the learned function and corresponding learning algorithm. For example, the engineer may choose to use support vector machines or decision trees.
- Complete the design. Run the learning algorithm on the gathered training set. Some supervised learning algorithms require the user to determine certain control parameters. These parameters may be adjusted by optimizing performance on a subset (called a validation set) of the training set, or via cross-validation.
- Evaluate the accuracy of the learned function. After parameter adjustment and learning, the performance of the resulting function should be measured on a test set that is separate from the training set.

### **Classification:**

A classification problem is when the output variable is a category, such as “red” or “blue” or “disease” and “no disease”. A classification model attempts to draw some conclusion from observed values. Given one or more inputs a classification model will try to predict the value of one or more outcomes. For example, when filtering emails “spam” or “not spam”, when looking at transaction data, “fraudulent”, or “authorized”. In short Classification either predicts categorical class labels or classifies data (construct a model) based on the training set and the values (class labels) in classifying attributes and uses it in classifying new data. There are a number of classification models. Classification models include logistic regression, decision tree, random forest, gradient-boosted tree, multilayer perceptron, one-vs-rest, and Naive Bayes.

### 1.2 Unsupervised Learning:

Unsupervised learning is a type of machine learning that looks for previously undetected patterns in a data set with no pre-existing labels and with a minimum of human supervision. In contrast to supervised learning that usually makes use of human-labeled data, unsupervised learning, also known as self-organization allows for modeling of probability densities over inputs.

Unsupervised learning is where you only have input data ( $X$ ) and no corresponding output variables.

The goal for unsupervised learning is to model the underlying structure or distribution in the data in order to learn more about the data.

These are called unsupervised learning because unlike supervised learning above there is no correct answers and there is no teacher. Algorithms are left to their own devises to discover and present the interesting structure in the data.

Unsupervised learning problems can be further grouped into clustering and association problems.

**Clustering:** A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior.

**Association:** An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy  $X$  also tend to buy  $Y$ .

## **2. Theoretical Background**

### **2.1. Risk Management at Banks**

The bank's management's pursuit to increase returns for its owners comes at the cost of increased risk. Banks are faced with various risks—interest rate risk, market risk, credit risk, off-balance-sheet risk, technology and operational risk, foreign exchange risk, country or sovereign risk, liquidity risk, liquidity risk and insolvency risk. Effective management of these risks is key to a bank's performance. Also, given these risks and the role that banks play in financial systems, they are subject to regulatory attention. The regulators require banks to hold capital for the many risks that arise and are carried due to a bank's varied operations. The Basel standards for the determination of capital requirements were developed in 1998, and since then, have developed and evolved. Capital is required for each of the main risk types. Credit risk has traditionally been the greatest risk facing banks, and usually the one requiring the most capital. Market risk arises primarily from the trading operations of a bank, while operational risk is the risk of losses from internal system failures or external events. In addition to calculating regulatory capital, most large banks also calculate economic capital, which is based on a bank's models rather than on prescriptions from regulators. The main risks that banks face are credit, market, and operational risks, with other types of risk including liquidity, business, and reputational risk. Banks are actively engaged in risk management to monitor, manage and measure these risks.

Market risk can be defined as the risk of losses “owing to movements in the level or volatility of market prices” . Market risk includes interest rate risk, equity risk, foreign exchange risk and commodity risk. Interest risk can be defined as the potential loss due to movements in interest rates. Equity risk can be defined as the potential loss consequent to an adverse change in the price of a stock. Foreign exchange risk can be defined as the risk that the value of the assets or liabilities of a bank changes due to fluctuations in the currency exchange rate. Commodity risk can be defined as the potential loss due to an adverse change in the price of commodities held. The market risk framework of the Basel accord consists of an internal models approach and a standardised approach. To capture tail risk better, the revised framework also saw a shift in the measure of risk under stress from the Value-at-Risk (VaR) to Expected Shortfall (ES).

Credit can be defined as the risk of potential loss to the bank if a borrower fails to meet its obligations (interest, principal amounts). Credit risk is the single largest risk banks face . The Basel Accord allows banks to take the internal ratings-based approach for credit risk. Banks can internally develop their own credit risk models for calculating expected loss. The key risk parameters to be estimated are probability of default (PD), loss given default (LGD) and exposure at default (EAD).  $\text{Expected Loss} = \text{PD} \times \text{LGD} \times \text{EAD}$ .

Liquidity risk, treated separately from the other risks, takes two forms—asset liquidity risk and funding liquidity risk. A bank is exposed to asset-liquidity risk when a transaction cannot be executed at the prevailing market prices, which could be a consequence of the size of the position relative to the normal trading lot size. Funding liquidity risk refers to the inability to meet cash flow obligations, and is also known as cash flow risk. A sound process for the identification, measurement, monitoring and control of liquidity risk should be implemented. Operational risk is defined by BCBS as the risk of loss resulting from “inadequate or failed internal processes, people and systems or from external events” and is a “fundamental element of risk management” at banks. This definition includes legal risk, but excludes strategic and reputational risk. It is considered inherent in all banking products, activities, processes and systems.

In the annual reports, operational risk was varyingly presented and included a number of sub risks, and could be referred to more as non-financial risk. It included, among many others, fraud risk, cyber security, clients products and business practices, information and resiliency risk, money laundering and financial crime risks, vendor and outsourcing risks, technology risk, business disruption risks. In some instances, banks have reported compliance and legal risk also under operational risk.

To determine the risks specific to banks, as an alternate to leveraging the existing literature, a review was done of bank annual reports. Based on the review, a taxonomy was charted of the various risk types that banks typically seek to manage as part of their business and the methodologies and tools in use. The annual reports of 10 leading banks were reviewed to determine which risk areas were specifically being reported on by these banks.

A chart (Figure 1) depicting the taxonomy of the various risk types discussed in bank annual reports and also the various methodologies or tools (Figure 2) implemented to manage these risks is included below.

The chief risk officer has access to risk insight and intelligence that was more retrospective in nature, such as incident analyses focusing on understanding what happened and why. Now, increasingly, they are gearing up with tools that allow for a look ahead that facilitates the predicting of potential risk incidents. Data mining, scenario modelling and forecasting are built-in features of most risk management solutions. Cognitive (pattern recognition by visualising and identifying apparent and later trends in historical data) and algorithmic (establishing causal relationships between diverse events and data sets) intelligence is making way for augmented (natural language processing and machine learning) and assistive (contextual virtual intelligent assistance) intelligence that

augments and accelerates decision making.



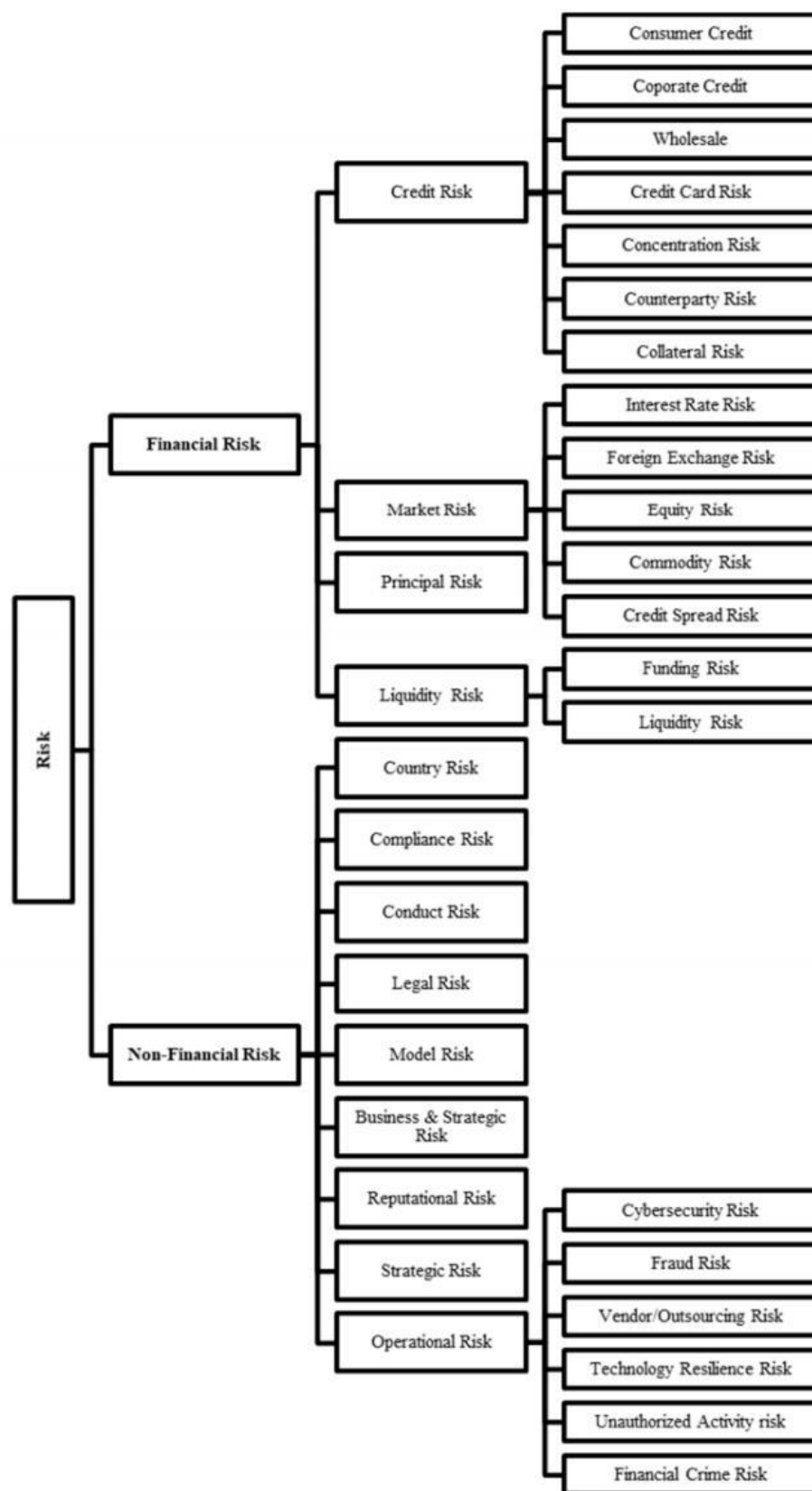


Figure 1. Taxonomy of risks.



	Market Risk	Credit Risk	Liquidity Risk	Non-Financial Risk (Operational Risk)
<b>Risk Management Tools</b>				
Risk Limits	√	√	√	
Credit Risk limits		√		
Value at Risk	√			
Earnings at Risk	√			
Expected Shortfall	√			
Economic Value Stress Testing	√			
Economic Capital	√	√	√	√
Risk Sensitivities	√			
Risk Assessment (RCSA)				√
Operational Risk Losses				√
Loss Distribution Approach				√
Scenario Analysis	√	√	√	√
Tail Risk Capture	√	√	√	√
Stress Testing	√	√	√	√
Scoring Models		√		
Rating Models		√		
Exposure				
- Probability of De fault		√		
- Loss Given De fault				
- Exposure at De fault				
Back Testing	√	√	√	
<b>Risk Management Framework Components</b>				
Risk Appetite	√	√	√	√
Risk Identification	√	√	√	√
Risk Assessment	√	√	√	√
Risk Measurement	√	√	√	√
Risk Testing	√	√	√	√
Risk Monitoring	√	√	√	√
Risk reporting	√	√	√	√
Risk Oversight	√	√	√	√
Capital Management (calculation and allocation)	√	√	√	√
- CCAR				
- ICAAP				

Figure 2. Risk Management Methods and Tools.

## **2.2. Machine Learning**

Machine learning has been explained as lying at the intersection of computer science, engineering and statistics. It has been highlighted as a tool that can be applied to various problems, especially in fields that require data to be interpreted and acted upon. Machine learning delivers the capability to detect meaningful patterns in data, and has become a common tool for almost any task faced with the requirement of extracting meaningful information from data sets. When faced with the requirement of extracting meaningful information from data, and the consequent complexity of patterns to be studied, a programmer may not be able to provide explicit and detailed specification on the execution process. Machine learning addresses this challenge by “endowing programs” with the ability to “learn and adapt”. The machine learning programs learn and improve, and can be applied when the problem that has to be dealt has the dual challenge of complexity and the need for adaptability.

Machine learning tools that are driving the advances in search engines and self-driving cars can be adopted and applied to the financial sector. A variety of technological developments have contributed to the financial sector being able to explore and mine a voluminous data infrastructure that includes diverse sets of unstructured forms of financial data about markets and consumers. Economists are increasingly adopting machine learning, in conjunction with other tools and expertise to evaluate complex relationships, despite machine learning’s limitations in being able to determine causality. The adoption of machine learning has been motivated by the potential opportunities for cost reduction, improved productivity and improved risk management. New regulations have also pushed the banks to automate with the need to have efficient regulatory compliance.

Data driven and computational-based, machine learning algorithms rely less on assumptions about the data, including about the distribution. While they are considered more robust and better at addressing complex non-linear relationships, they also are seen as being difficult to interpret.

Recent years have seen a surge in the amount of data gathered within financial institutions (FI). A big push towards the digitalisation of services and increased regulatory reporting requirements has resulted in a large amount of unstructured data being created and/or collected at a high frequency. This data comes from various sources, including consumer apps, client interactions, metadata and other external data sources. The desire to enhance their analytical capabilities and automate across business lines, including risk management, by managing and mining these increased volumes and a variety of data has led financial institutions to explore powerful and analytical solutions, a consequence of which is the rise in interest and the popularity of machine learning and artificial intelligence within the FI community. Machine learning is widely seen in the financial services sector as having the potential to deliver the analytical capability that FIs desire.

Machine learning is capable of impacting every aspect of the FI's business model—improving insight into client preferences, risk management, fraud detection, conduct monitoring, client support automation and even automated identity verification when coupled with biometrics.

Van Liebergen (2017) introduces the field, and through discussions with the Institute of International Finance and technology ventures, explains use cases within financial institutions. He discusses applications in the area of credit risk modelling, detection of credit card fraud and money laundering and surveillance of conduct breaches at FIs. He also highlights that Machine learning seeks to predict “out-of-sample” while learning “found in-sample” (past) correlations, while falling short of providing an explanation for the analysed relationship. This could create complexities around model development and evaluation.

Machine learning also plays a role at the Securities and Exchange Commission (SEC) in the risk assessment process in identifying misconduct. While this is applicable from a supervisory perspective and for the oversight of systemic risks, it can also serve as a guide for a bank on how similar machine learning techniques can be applied in risk assessments for the detection of misconduct (internal or external) including risk assessments on corporate issuers or counterparties. In computational finance, machine learning has great potential and could be variedly used, ranging from the comprehensive exploratory data analysis to the presentation/visualisation of modelling results.

Some of the cons of machine learning, as argued, are that they are more “black box” in nature, with results at times being difficult to interpret. It is argued that they are also sensitive to outliers, resulting in the overfitting of the data and counterintuitive predictions. They are also argued to have the pros of being able to be a better fit for non-linear relationships between the explanatory variables and explained variables, and also that the ability for them to apply a broader set of variables tends to improve accuracy.

### 3. Materials and Methods

To carry out the review of literature that researches the application of machine learning in bank risk management, two sets of key words were used in the search for related papers. The search for papers was done using the scholar.google.com, SSRN and ProQuest databases. The search was largely focused on papers after 2007 to capture developments since the global financial crisis; however papers prior to that period were also included if they were referenced in other recent papers.

The first group of words was ‘machine learning’, in line with the topic. The second group comprised terms that were identified from the review of the bank annual reports. This includes risk types, as listed in the risk taxonomy and risk management tools or methods that were identified from the bank annual reports. Taxonomy is as shown in Figure 1, and methods as in Figure 2.

The search and review was limited to conference papers, journal articles and selected theses (post graduate or doctoral). The review has not considered articles, white papers, vendor papers or web articles that have just made reference to machine learning without providing details on how, or that made references to the application of any specific algorithm, though many such articles did come up in the search. In particular, there are a large number of articles, web and magazines, and publications that include machine learning as a solution or as a generic and general recommendation without providing further details on how a given specific problem can be addressed.

The review has looked at only papers that have analysed the topic with a level of depth, namely, by making references to specific algorithms or providing a design or model for how ML can be implemented. Articles or papers or conference proceedings that have made only a cursory or a general reference to the application of ML in the risk management space have not been considered for this research. It is noted that there are many references available where the authors or speakers have proposed that ML or AI can be applied in the management of risk; however, many of them stop short of providing clarity on which algorithms, or fail to provide examples of how ML/AI has been applied in a test or industry setup.

The methodological framework for this research was determined by analysing the various problem areas related to machine learning and risk management in banks. The articles were classified to understand: (i) the risk area they focused on; (ii) the risk management tool or risk management framework component they targeted; or (iii) the algorithms that were applied/studied/proposed. The survey was also seeking to review papers that focused more on risk assessment and measurement. Risk areas such as cybersecurity and fraud risk have been dealt with widely; however, the focus in this review has been only on cases where they specifically relate to banking risk management use cases.

Papers that focus the research on operational matters, such as credit risk management solutions that address the

operational process of credit review and approval, or tools that are focused on supporting traders and trading risk managers in the order and trade management process, have not been considered. Additionally, operational risk management solutions that fit within the operational process to mitigate operational events/incidents (e.g., robotics process automation, STP, anomaly detection) have not been researched.

### 3.1. Credit Risk

The assessment of credit risk remains an important and challenging research topic in the field of finance, with initial efforts dating back to the last century. On the back of the global financial crisis events and the consequent increased regulatory focus, the credit risk assessment process has seen an increased interest within the academic and business community. The general approach to credit risk assessment has been to apply a classification technique on past customer data, including on delinquent customers, to analyse and evaluate the relation between the characteristics of a customer and their potential failure. This could be used to determine classifiers that can be applied in the categorisation of new applicants or existing customers as good or bad.

Credit risk evaluation occupies an important place within risk management. Techniques such as Logistic regression and discriminant analysis are traditionally used in credit scoring to determine likelihood of default. Support Vector machines are successful in classifying credit card customers who default. They were also found to be competitive in discovering features that are most significant in determining risk of default when tested and compared against the traditional techniques. Credit risk modelling for the calculation of credit loss exposure involves the estimation of the Probability of Default (PD), the Exposure at Default (EAD) and the Loss Given Default (LGD). This is emphasised by the Basel II accord. Predominant methods to develop models for PD are classification and survival analysis, with the latter involving the estimation of whether the customer would default and when the default could occur. Classifier algorithms were found to perform significantly more accurately than standard logistic regression in credit scoring. Also, advanced methods were found to perform extremely well on credit scoring data sets such as artificial neural networks, performing better than extreme learning machine.

Through the Basel accord requirements, the need to allocate capital in an efficient and profitable manner has lead FIs to build credit scoring models to assess the default risk of their customers. Again, SVM has been shown to yield significantly better results in credit scoring.

An accurate prediction of estimated probability of default delivers more value to risk management in comparison to a binary classification of clients as either credible or not-credible. A number of techniques are used in credit scoring, such as discriminant analysis, logistic regression, Bayes classifier, nearest neighbour,

artificial neural networks and classification trees.

Artificial neural networks have been shown to perform classifications more accurately than the other five methods. Methods and models are being constantly developed to address a significant issue at banks, namely, the correct classification of customers and the estimation of credit risk. The various approaches applied in these methods seek to increase the accuracy of creditworthiness predictions that could lead to a bigger and profitable loan portfolio. Neural networks have proven to be of significant value in the credit risk decision process, and their application in company distress predictions was reported to be beneficial in credit risk evaluation.

While credit risk is the most researched and evaluated risk area for the application of machine learning, this is not a new phenomenon. Dating as far back as 1994, Altman and colleagues conducted an analysis comparing traditional statistical methods of distress and bankruptcy prediction with alternative neural network algorithm, and concluded that a combined approach of the two improved accuracy significantly.

Some papers focus on a comparison against traditional statistical methods to highlight the efficiency in applying machine learning algorithms. Galindo and Tamayo (2000) research, through a comparative analysis of statistical and machine learning classification techniques, the credit portfolios of institutions to find accurate predictions of individual risk. They built more than 9000 models as part of the study and ranked the performance of the various algorithms. They show that the CART decision tree models provided the best estimates for default, with neural networks coming second. Hamori et al. (2018) studied and compared the prediction accuracy and classification ability of bagging, random forest, boosting with neural network methods in analysing default payment data. They found boosting to be superior among the studied machine learning methods.

A number of researchers have also evaluated the application of hybrid techniques and ensemble methods to study credit scoring. In a hybrid system, one technique is employed for the final prediction after the use of several heterogeneous techniques in the analysis. In dealing with credit scoring problems, ensemble learning, using regularised logistic regression, can be applied. A method of applying clustering and bagging algorithms to balance and diversify the data, followed by lasso-logistic regression ensemble to evaluate credit risks, was found to outperform many popular credit-scoring models, to improve classification rates of credit card holder delinquencies and defaults, constructed a nonlinear, non-parametric forecast model. The consumer credit risk model was able to identify subtle non-linear relationships in massive datasets. These relationships were typically reportedly difficult to detect when using standard consumer credit-default models such as logit, discriminant analysis or credit scores. This allows for credit line risk management, the forecasting of aggregate consumer credit delinquencies and the forecasting of consumer credit cycle.



Yu et al. (2016) propose a novel multistage deep belief network based extreme machine learning as promising tool for credit risk assessment. The framework of multistage ensemble learning paradigms, working at three stages, is shown to outperform typical single classification techniques and similar multistage ensemble learning paradigms with high prediction accuracy.

“Support Vector Machine” (SVM) is a supervised machine-learning algorithm, and while it is widely used in classification problems, it is relatively new to credit scoring. In this algorithm, each data item is plotted as a point in  $n$ -dimensional space, the value of each feature is the value of a particular coordinate ( $n$ —is number of features). Classification is performed by finding the hyper-plane that is the frontier that segregates two classes. The SVM has been applied as is or in some varied form to design a credit risk evaluation and credit scoring models. Harris (2013) compares SVM-based credit scoring models using broad ( $<90$  days past due) and narrow ( $>90$  days past due), the latter being the more traditional approach. It was found that models built using a broader definition were more accurate, allowing for improvements in prediction accuracy.

Stress testing requires the modelling of the link between macro-economic developments and banking variables to determine the impact of extreme scenarios on a bank. More frequently, bottom-up approaches are used where predictions about future profits/losses are made on mostly disaggregated portfolio levels, making it data intensive and difficult to identify the exact drivers of losses. Predictions on an aggregated portfolio using a top down method can complement this process. A supervised learning algorithm that does not need a pre-specified model is the Least Absolute Shrinkage and Selection Operator (Lasso) method. A more involved version of the Lasso is Adaptive Lasso, which possesses attractive convergence properties. Adaptive Lasso can be used in the absence of theoretical models, as in the case of top-down stress testing, to discover a parsimonious top-down model from a set of thousand possible specifications. It was shown to give sparse, approximately unbiased solutions, by searching for variables that describe the behaviour of credit loss rates best resulting in a parsimonious description of the relation between macro-economy and credit loss rates. A key issue is the need for substantial amounts of data to train a model.

Model selection and forecasting have become a challenge as stress scenarios become more comprehensive, encompassing an increasing number of primary variables. Machine learning techniques for identifying patterns and relationships between data can facilitate model selection and forecasting. These techniques don't seem to be widely applied in stress testing. When there are a large number of potential covariates and the number of observations is small, lasso regressions are found to be suited to building forecasting models. They are likely to outperform traditional statistical models in forecasting the performance indicators required in applied stress

testing. An advantage of the Lasso-type estimators is that they can handle complications arising from the high dimensional nature of stress tests. The Multivariate Adaptive Regression Splines (MARS), a machine learning technique, can be viewed as a generalisation of stepwise linear regression of the classification and regression tree (CART) method. Stress testing statistical regression models such as Vector Autoregression (VAR) is a common modelling approach which is known to be unable to explain the phenomenon of fat-tailed distributions. An empirical test of these models found that the MARS model exhibited greater accuracy in model testing and superior out-of-sample performance, with MARS producing more reasonable forecasts. Probabilistic graphs may be used for modelling and assessments of credit concentration risk with a tree-augmented Bayesian network providing a better understanding of the risk. This was also found to be suitable for stress testing analyses, with the ability to provide estimates of risk of losses consequent to changes in a borrower's financial condition. EMA workbench is a software toolbox, developed by a team at Technische Universiteit Delft, TBM Faculty (Policy Analysis Section). Particular machine learning algorithms and advanced visualisation tools are used to perform multiple experiments and analyse the results, providing the ability to explore possible uncertainties and identify causes based on the inputs.

Neural networks, Support Vector Models and Random Forest appear to be the most researched algorithms in the credit risk management area.

### 3.2. Market Risk

Risk can be measured by the standard deviation of unexpected outcomes, also called volatility. Value at Risk (VAR) calculates the worst loss over a target horizon that will not be exceeded with a given level of confidence and captures the combined effect of underlying volatility and exposure to financial risks. Volatility forecasting in the financial markets is important in the areas of risk management and asset pricing, among others. By using NN models, the performance of the volatility estimation method can be improved.

Zhang et al. (2017) propose a model that is based on the Generalized Autoregressive Conditional Heteroskedastic (GARCH) model and Extreme Machine Learning (ELM) algorithm to estimate volatility. The model predicts the volatility of target time series using GELM-RBF and extrapolating the predicted volatilities allows for the calculation of VaR with improved performance in terms of accuracy and efficiency. The model utilises a stochastic mapping method that doesn't require the Gaussian likelihood for estimation and is a non-linear data driven model.

Market risk also includes interest rate and equity risk. Interest rate curves, which is the relation between the

interest rate and the time to maturity of the debt for a given borrower in a given currency, is widely used in financial engineering and market risk management. A clustering method called the “Gaussian Mixture Model” can be used to develop nonlinear models of the evolution of the parameters and then to forecast interest rate curves. This can allow for better visualisation of interest rates. Machine learning clustering methods designed to address Stochastic Differential Equation (SDE) can be applied to develop anticipatable VAR models that aim at being a leading risk measure of market regime change. This can partially address some of the complexity introduced by the challenging regulatory environment, such as scenario coherence.

### **3.3. Liquidity Risk**

A number of liquidity risk problems can be solved through the use of machine learning. Measurement of liquidity risk, the analysis of key factors including the study of the interconnections between the factors can be achieved through the use of machine learning. For the purposes of estimating a risk measure Artificial Neural Networks (ANN), a genetic algorithm can be applied. ANN can be used in the approximation of the general risk trend and determination of the most influential factors. The probability that a liquidity risk event will occur can be estimated by the application of Bayesian Networks. The ANN and the BN implementations were capable of distinguishing the most critical liquidity risk factors measuring the risk by a functional approximation and a distributional estimation, respectively.

### **3.4. Operational Risk**

Machine learning is also applied in operational areas that enable the mitigation of risk, i.e., detection and/or prevention of risks. In the area of operational risk, aside from cyber security cases, machine learning is predominantly focused on problems related to fraud detection and suspicious transactions detection.

Khrestina et al. (2017), in their paper, propose a prototype for the generation of a report that allows for the detection of suspicious transactions. The prototype uses a logistical regression algorithm. It is noteworthy that they have also included a survey of six software solutions that are currently implemented at various banks for the automation of suspicious transaction detection and monitoring processes. While the authors make a reference to algorithms, it is unclear whether these products apply machine learning techniques, and if so, with which algorithms. No further research was done on these products as this was not in the scope of the paper.

One such area where an intelligent system based on machine learning is known to add value is in the defence

against spammers where the attackers' techniques evolve. Losses from spam potentially include lost productivity, disrupted communications, malware attacks and theft of data, including financial loss. Proofpoint's MLX technology uses advanced machine learning techniques to provide comprehensive spam detection that guards against the threat of spam. Millions of messages can be analysed by the technology, which also automatically refines the detection algorithm to identify and detect newer threats. While it was not in scope for this research, being more of an operational control to manage risk, it has been highlighted as a case of how machine learning is used in managing cybersecurity risks.

In money laundering, criminals route money through various transactions, layering them with legitimate transactions to conceal the true source of the funds. The funds typically originate from criminal or illegal activities, and can be further used in other illegal activities including the financing of terrorist activities. There has been extensive research on detecting financial crimes using traditional statistical methods, and more recently, using machine-learning techniques. Clustering algorithms identify customers with similar behavioural patterns and can help to find groups of people working together to commit money laundering. A major challenge for banks, given the large volume of transactions per day and the non-uniform nature of many, is to be able to sort through all the transactions and identify those that are of suspicious nature. Financial institutions utilise anti-money laundering systems to filter and classify transactions based on degrees of suspiciousness. Structured processes and intelligent systems are required to enable the detection of these money laundering transactions.

Money laundering is another area that poses a significant challenge to financial institutions, given the high volumes and complexity of transactions coupled with the dynamic and fast evolving nature of financial crimes, and the need to do so on real-time data sets.

In the area of financial crime detection, there has been a significant amount of research in the application of statistical learning and data mining for developing classification models to flag suspicious transactions.

The systems are oriented towards increasing the detection rate while minimising the false positive rate. Models are estimated based on samples of fraudulent and legitimate transactions in supervised detection methods while in unsupervised detection methods outliers or unusual transactions are identified as potential cases of fraud. Both seek to predict the probability of fraud in a given transaction. Some reported challenges in credit card fraud detection are the non-availability of real data sets, unbalanced data sets, size of the data sets and the dynamic behaviour of fraudster. Bayesian algorithms, K-Nearest neighbor, Support Vector machines (SVM) and bagging ensemble classifier based on decision tree have been varyingly used in fraud detection systems.

A comparative evaluation showed that the bagging ensemble classifier based on decision tree algorithms works

well, as it is independent of attribute values, and is also able to handle class imbalance. False alarms, namely transactions labelled as fraudulent that are in fact legitimate, are significant, causing concerns for customers and delaying the detection of actual fraud transactions. Large Canadian banks rely heavily on NN scores, ranging from 1 to 999, with 1 being the lowest chance of a fraudulent transaction, determined by neural network algorithms.

Reportedly, 20% of transactions with a NN score greater than or equal to 990 are fraudulent, causing fraud analysts to inefficiently spend time investigating legitimate transactions. A meta-classifier (a multiple algorithm learning technique) applied to a post-neural network was shown to provide quantifiable savings improvements with a larger percentage of fraudulent transactions being caught.

Also, in the areas of operational risk, there are a few papers on fraud risk detection in credit cards and online banking. They concern credit card fraud detection in domains not specifically related to bank risk management or the banking industry. One would note that the algorithms they refer to were SVM, KNN, Naïve Bayes Classifier, Bagging ensemble classifier based on decision tree.

**Example 1:** The software, Correctional Offender Management Profiling for Alternative Sanctions (COMPAS), measures the risk of a person to recommit another crime. Judges use COMPAS to decide whether to release an offender, or to keep him or her in prison. An investigation into the software found a bias against African-Americans: COMPAS is more likely to assign a higher risk score to African-American offenders than to Caucasians with the same profile.

**Example 2:** In addition to COMPAS, discriminatory behaviour was also evident in an algorithm that would deliver advertisements promoting jobs in Science, Technology, Engineering and Math fields (STEM). This advertisement was designed to deliver advertisements in a gender-neutral way. However, less women compared to men saw the advertisements due to gender-imbalance which would result in younger women being considered as a valuable subgroup and more expensive to show advertisements to. This optimization algorithm would deliver ads in a discriminatory way although its original type and pure intention was to be gender-neutral.

The nation's consumer protection agency, enforces the Equal Credit Opportunity Act (ECOA), which prohibits

credit discrimination on the basis of race, color, religion, national origin, sex, marital status, age, or because you get public assistance. This is why it is important to have a fair algorithm.

So it is required to think responsibly, and recognize that these tools might be used in making decisions which affect peoples' lives. Therefore considering fairness constraints is a crucial task while designing and implementing these types of sensitive tools or models.

### **4. Problem Statement**

Many new customers have approached the bank for the purpose of acquiring loans. The bank is interested in finding out whether these customers would be able to repay the amount of loan if provided to them so that the risk of loss can be reduced. Pertaining to this objective, a sample dataset (train set) has been assembled which relates to the previous loan takers of the bank and whether these customers were able to repay the amount of loan taken by them based upon certain factors.

The task is to design a predictive model to estimate the likelihood of the bank loan default on account of a borrower based on multiple factors like Loan Term, Credit score, Gender, Type of Loan, Employment Status and other such characteristic metrics. The goal is to predict bank loan default instances, while ensuring that if a client is rejected, we are capable of demonstrating the decision was fair and providing underlying reasons. Hence the next task is mitigating the bias(based on protected/sensitive features of the dataset) in the classification to create a fair model .

### 5. Dataset and its Description

Data	Note
------	------

---

## CREDIT DEFAULT RISK PREDICTIPON WITH MACHINE LEARNING

---

Train Set	This dataset involves the characteristics of old customers i.e previous loan takers of the bank
Test Set	This dataset involves the characteristics of new loan acquirers

Variables	Descriptions
ID	Customer ID
Checking_amount	Amount of money in the checking account
Term	Duration of loan borrowed
Credit_score	Credit score to ascertain the risk profile of a borrower
Gender	Sex of customer
Marital_status	Whether the customer is married or singlet
Car_loan	Whether the customer has taken car loan or not
Personal_loan	Whether the customer has taken personal loan or not
Home_loan	Whether the customer has taken home loan or not



---

## CREDIT DEFAULT RISK PREDICTIPON WITH MACHINE LEARNING

---

Education_loan	Whether the customer has taken education loan or not
Emp_status	Whether the customer is employed or not
Amount	Value of loan amount borrowed
Saving_amount	Balance of saving account
Emp_durtion	Total duration of employment
Age	Age of the customer
No_of_credit_ac	Number of credit account a customer has
Default	Whether the customer defaulted or not (Defaulter is the person who is unable to repay loan)

ID,Checking\_amount,Term,Credit\_score,Amount,Saving\_amount,Emp\_duration,Age,No\_of\_credit\_acc are numeric.

All other attributes are in string format which are then converted into numeric values using labelencoder.

Attribute	Value 1	Value 2
Gender	Male	Female

---

## CREDIT DEFAULT RISK PREDICTIPON WITH MACHINE LEARNING

---

Marital_status	Single	Married
Car_loan	Yes	No
Personal_loan	Yes	No
Home_loan	Yes	No
Education_loan	Yes	No
Emp_status	Yes	No
No_of_credit_acc	Yes	No

Out of these, Gender, Marital\_status and Age are the Protected attributes.

## 6. Implementation Details

### Data Observation:

'data.frame': 804 obs. of 17 variables:

\$ ID : num 101 102 103 104 105 106 107 108 109 110 ...

\$ Default : Factor w/ 2 levels "No","Yes": 1 1 1 2 2 1 1 1 1 2 ...

\$ Checking\_amount : num 988 458 158 300 63 ...

\$ Term : num 15 15 14 25 24 20 13 16 20 19 ...

\$ Credit\_score : num 796 813 756 737 662 828 856 763 778 649 ...

\$ Gender : Factor w/ 2 levels "Female","Male": 1 1 1 1 1 2 2 1 1 2 ...

\$ Marital\_status. : Factor w/ 2 levels "Married","Single": 2 2 2 2 2 1 2 2 2 1....

\$ Car\_loan : Factor w/ 2 levels "No","Yes": 2 2 1 1 1 2 2 2 2 2 ...

\$ Personal\_loan : Factor w/ 2 levels "No","Yes": 1 1 2 1 1 1 1 1 1 1 ...

\$ Home\_loan : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...

\$ Education\_loan : Factor w/ 4 levels "0","1","No","Yes": 3 3 3 4 4 3 3 3 3 3 ...

\$ Emp\_status : Factor w/ 2 levels "employed","unemployed": 1 1 1 1 2 1 1 1 2 1..

\$ Amount : num 1536 947 1678 1804 1184 ...

\$ Saving\_amount : num 3455 3600 3093 2449 2867 ...

\$ Emp\_duration. : num 12 25 43 0 4 12 11 12 12 0 ...

\$ Age : num 38 36 34 29 30 32 38 36 36 29 ...

\$ No\_of\_credit\_acc: num 1 1 1 1 1 2 1 1 1 1 ...

## **7. Exploratory Data Analysis (EDA)**

Exploratory Data Analysis (EDA) is used on the one hand to answer questions, test business assumptions, generate hypotheses for further analysis. On the other hand, you can also use it to prepare the data for modeling. The thing that these two probably have in common is a good knowledge of your data to either get the answers that you need or to develop an intuition for interpreting the results of future modeling.

There are a lot of ways to reach these goals: you can get a basic description of the data, visualize it, identify patterns in it, identify challenges of using the data, etc.

One of the things that you'll often see when you're reading about EDA is Data profiling. Data profiling is concerned with summarizing your dataset through descriptive statistics. You want to use a variety of measurements to better understand your dataset. The goal of data profiling is to have a solid understanding of your data so you can afterwards start querying and visualizing your data in various ways. However, this doesn't mean that you don't have to iterate: exactly because data profiling is concerned with summarizing your dataset, it is frequently used to assess the data quality. Depending on the result of the data profiling, you might decide to correct, discard or handle your data differently.

The data consists of 804 rows and 17 columns. Out of the 17 columns, there is 1 target variable and 16 predictor variables. Since the target variable i.e., whether a person defaults on loan or not is a categorical variable, this is a classification problem.

Looking at the type of predictor variables, it can be observed that there are 7 categorical predictor variables and 10 discrete predictor variables.

All but one of the 7 categorical variables are nominal. The other categorical variable is shown as an ordinal variable with 4 levels. But, if we look at the classes, {"No", "Yes", "0", "1"}, it doesn't make sense to have all these classes. "0" values can be reassigned to "No" and "1" to "Yes" class respectively.

A basic sanity check of the data shows that NA or missing values exist.

	ID Default	Checking_amount	Term	Credit_score
0	1	1	1	2
Gender	Marital_status.	Car_loan	Personal_loan	Home_loan
0	0	1	3	2
Education_loan	Emp_status	Amount	Saving_amount	Emp_duration.
1	0	1	0	3
Age	No_of_credit_acc			
1	1			

The EDA is started by performing a cross table analysis.

### **Crosstable Analysis:**

The cross tables revealed the following interesting insights:

1. Higher defaults among single (32.9%) than married (25.3%)
2. Higher defaults among females (33.9%) than males (26.7%)
3. Higher defaults among those having car loan (37%) to those without (25.6%)
4. Higher defaults among those without personal loan (41.2%) to those with (17.9%)
5. Higher defaults among those without home loan (30.3%) to those with (11.9%)
6. Almost equal defaults among those employed (28.9%) and those unemployed (29.5%)

### DATA PREPROCESSING AND EXPLORATION:

- Identified potentially discriminatory features, i.e., protected attributes, in the dataset to be the age group, marital\_status and the gender.
- Handled the missing values in the data in an appropriate way
- To start out, the number of rows having missing values in the data are checked, then for each column the number of missing values are checked.
- Then, the remaining columns are inspected to determine the optimal way to fill the missing values of each column like calculating mean or replacing with a “Yes” or “No” etc.
- The correlation is analysed between each pair of columns by drawing a correlation matrix  
to identify potential columns to drop

### Libraries:

**Numpy:** A third party package allowing us to work with multidimensional arrays.

**Pandas:** Allow us to organize data in a tabular form and to attach descriptive labels to rows and columns.

**Mathplotlib.pyplot:** Used for plotting a graph.

## Import necessary libraries

```
In [1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

## Load train and test file

```
In [2]: trainD = pd.read_csv("epsilon_train.csv")
trainD = pd.DataFrame(trainD)
testD = pd.read_csv("epsilon_test.csv")
testD = pd.DataFrame(testD)
trainD.head()
```

Out[2]:

	ID	Default	Checking_amount	Term	Credit_score	Gender	Marital_status	Car_loan	Personal_loan	Home_loan	Education
0	101	No	988.0	15.0	796.0	Female	Single	Yes	No	No	
1	102	No	458.0	15.0	813.0	Female	Single	Yes	No	No	
2	103	No	158.0	14.0	756.0	Female	Single	No	Yes	No	
3	104	Yes	300.0	25.0	737.0	Female	Single	No	No	No	
4	105	Yes	63.0	24.0	662.0	Female	Single	No	No	No	

```
In [3]: # Train datatype info
trainD.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 804 entries, 0 to 803
Data columns (total 17 columns):
ID                804 non-null int64
Default           803 non-null object
Checking_amount   803 non-null float64
Term              803 non-null float64
Credit_score      802 non-null float64
Gender            804 non-null object
Marital_status    804 non-null object
Car_loan          803 non-null object
Personal_loan     801 non-null object
Home_loan         802 non-null object
Education_loan    803 non-null object
Emp_status        804 non-null object
Amount            803 non-null float64
Saving_amount     804 non-null int64
Emp_duration      801 non-null float64
Age               803 non-null float64
No_of_credit_acc  803 non-null float64
dtypes: float64(7), int64(2), object(8)
memory usage: 106.9+ KB
```

```
In [4]: # Check columnwise null values
print(trainD.isnull().sum())
```

```
ID                0
Default           1
Checking_amount    1
Term              1
Credit_score      2
Gender            0
Marital_status     0
Car_loan          1
Personal_loan      3
Home_loan         2
Education_loan     1
Emp_status        0
Amount            1
Saving_amount      0
Emp_duration       3
Age               1
No_of_credit_acc   1
dtype: int64
```



## Handling the missing values (Fill with appropriate mean, mode or median value)

```
In [5]: # Default column
trainD.Default.value_counts(sort=True)
trainD.Default.fillna('Yes', inplace=True)

#Checking amount column
trainD.Checking_amount.value_counts(sort=True)
trainD.Checking_amount.fillna(trainD["Checking_amount"].mean(), inplace=True)

#Term column
trainD.Term.value_counts(sort=True)
trainD.Term.fillna(trainD["Term"].mean(), inplace=True)

# credit score column
trainD.Credit_score.value_counts(sort=True)
trainD.Credit_score.fillna(trainD["Credit_score"].mean(), inplace=True)

# Car_loan column
trainD.Car_loan.value_counts(sort=True)
trainD.Car_loan.fillna("Yes", inplace=True)

# personal loan column
trainD.Personal_loan.value_counts(sort=True)
trainD.Personal_loan.fillna('Yes', inplace=True)

# home loan column
trainD.Home_loan.value_counts(sort=True)
trainD.Home_loan.fillna('Yes', inplace=True)

# Education loan column
trainD.Education_loan.value_counts(sort=True)
trainD.Education_loan.fillna('Yes', inplace=True)

# amount
trainD.Amount.value_counts(sort=True)
trainD.Amount.fillna(trainD["Amount"].mean(), inplace=True)

# Emp_duration
trainD.iloc[:,14].value_counts(sort=True)
trainD.iloc[:,14].fillna(trainD.iloc[:,14].mean(), inplace=True)

#age
trainD.Age.value_counts(sort=True)
trainD.Age.fillna(trainD["Age"].mean(), inplace=True)

#No_of_credit_acc column
trainD.No_of_credit_acc.value_counts(sort=True)
trainD.No_of_credit_acc.fillna(1, inplace=True)
```

```
In [6]: # Drop all the rows which contains null values
# trainD = trainD.dropna(axis = 0, how = 'any')
```

```
In [7]: # check whether all null values are removed or not
print(trainD.isnull().sum())
```

```
ID          0
Default      0
Checking_amount  0
Term         0
Credit_score  0
Gender       0
Marital_status 0
Car_loan     0
Personal_loan 0
Home_loan    0
Education_loan 0
Emp_status   0
Amount       0
Saving_amount 0
Emp_duration  0
Age          0
No_of_credit_acc 0
dtype: int64
```

```
In [8]: testD.head()
```

Out[8]:

	ID	Checking_amount	Term	Credit_score	Gender	Marital_status	Car_loan	Personal_loan	Home_loan	Education_loan
0	1001	111.0	20	668.0	Male	Married	No	Yes	No	No
1	1002	270.0	19	612.0	Male	Married	No	No	No	Yes
2	1003	211.0	19	751.0	Male	Married	Yes	No	No	No
3	1004	325.0	16	722.0	Male	Married	Yes	No	No	No
4	1005	1237.0	21	841.0	Male	Married	Yes	No	No	No

In [9]: testD.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 203 entries, 0 to 202
Data columns (total 16 columns):
ID                203 non-null int64
Checking_amount   202 non-null float64
Term             203 non-null int64
Credit_score     202 non-null float64
Gender           203 non-null object
Marital_status   202 non-null object
Car_loan         202 non-null object
Personal_loan    203 non-null object
Home_loan        201 non-null object
Education_loan   201 non-null object
Emp_status       203 non-null object
Amount           202 non-null float64
Saving_amount    202 non-null float64
Emp_duration     201 non-null float64
Age              203 non-null int64
No_of_credit_acc 202 non-null float64
dtypes: float64(6), int64(3), object(7)
memory usage: 25.5+ KB
```

In [10]: print(testD.isnull().sum())

```
ID                0
Checking_amount    1
Term              0
Credit_score      1
Gender            0
Marital_status    1
Car_loan          1
Personal_loan     0
Home_loan         2
Education_loan    2
Emp_status        0
Amount            1
Saving_amount     1
Emp_duration      2
Age              0
No_of_credit_acc  1
dtype: int64
```

```
In [12]: testD=testD.dropna(axis=0,how='any')
```

```
In [45]: print(testD.isnull().sum())
```

```
ID          0
Checking_amount  0
Term         0
Credit_score  0
Gender        0
Marital_status  0
Car_loan      0
Personal_loan  0
Home_loan     0
Education_loan  0
Emp_status    0
Amount        0
Saving_amount  0
Emp_duration  0
Age           0
No_of_credit_acc  0
dtype: int64
```

### Label Encoding:

Label Encoding refers to converting the labels into numeric form so as to convert it into the machine-readable form. Machine learning algorithms can then decide in a better way on how those labels must be operated. It is an important pre-processing step for the structured dataset in supervised learning.

The categorical variables are label encoded.

Scikit learn library is used for label encoding. Scikit learn accepts data in numeric format. Each character variable is converted into numeric using the labelencoder function. In label encoding, each unique value of a variable gets assigned a number.

## Label Encoding of Categorical Variables

```
In [14]: from sklearn.preprocessing import LabelEncoder
```

```
In [15]: number=LabelEncoder()
trainD['Gender']=number.fit_transform(trainD['Gender'].astype('str'))
testD['Gender']=number.fit_transform(testD['Gender'].astype('str'))
```

```
In [16]: trainD.head()
```

Out[16]:

	ID	Default	Checking_amount	Term	Credit_score	Gender	Marital_status	Car_loan	Personal_loan	Home_loan	Education
0	101	No	988.0	15.0	796.0	0	Single	Yes	No	No	
1	102	No	458.0	15.0	813.0	0	Single	Yes	No	No	
2	103	No	158.0	14.0	756.0	0	Single	No	Yes	No	
3	104	Yes	300.0	25.0	737.0	0	Single	No	No	No	
4	105	Yes	63.0	24.0	662.0	0	Single	No	No	No	

```
In [17]: testD.head()
```

Out[17]:

	ID	Checking_amount	Term	Credit_score	Gender	Marital_status	Car_loan	Personal_loan	Home_loan	Education_loan
0	1001	111.0	20	668.0	1	Married	No	Yes	No	No
1	1002	270.0	19	612.0	1	Married	No	No	No	Yes
2	1003	211.0	19	751.0	1	Married	Yes	No	No	No
3	1004	325.0	16	722.0	1	Married	Yes	No	No	No
4	1005	1237.0	21	841.0	1	Married	Yes	No	No	No

```
In [18]: #labelEncoding for on more variables
number=LabelEncoder()
trainD['Personal_loan']=number.fit_transform(trainD['Personal_loan'].astype('str'))
testD['Personal_loan']=number.fit_transform(testD['Personal_loan'].astype('str'))
```

## CREDIT DEFAULT RISK PREDICTIPON WITH MACHINE LEARNING

```
In [19]: trainD.head()
```

```
Out[19]:
```

	ID	Default	Checking_amount	Term	Credit_score	Gender	Marital_status	Car_loan	Personal_loan	Home_loan	Education
0	101	No	988.0	15.0	796.0	0	Single	Yes	0	No	
1	102	No	458.0	15.0	813.0	0	Single	Yes	0	No	
2	103	No	158.0	14.0	756.0	0	Single	No	1	No	
3	104	Yes	300.0	25.0	737.0	0	Single	No	0	No	
4	105	Yes	63.0	24.0	662.0	0	Single	No	0	No	

```
In [20]: trainD['Emp_status']=number.fit_transform(trainD['Emp_status'].astype('str'))  
testD['Emp_status']=number.fit_transform(testD['Emp_status'].astype('str'))
```

```
In [21]: trainD.head()
```

```
Out[21]:
```

	ID	Default	Checking_amount	Term	Credit_score	Gender	Marital_status	Car_loan	Personal_loan	Home_loan	Education
0	101	No	988.0	15.0	796.0	0	Single	Yes	0	No	
1	102	No	458.0	15.0	813.0	0	Single	Yes	0	No	
2	103	No	158.0	14.0	756.0	0	Single	No	1	No	
3	104	Yes	300.0	25.0	737.0	0	Single	No	0	No	
4	105	Yes	63.0	24.0	662.0	0	Single	No	0	No	



```
In [22]: trainD['Default']=number.fit_transform(trainD['Default'].astype('str'))
```

```
In [23]: trainD['Car_loan']=number.fit_transform(trainD['Car_loan'].astype('str'))
testD['Car_loan']=number.fit_transform(testD['Car_loan'].astype('str'))
```

```
In [24]: trainD['Home_loan']=number.fit_transform(trainD['Home_loan'].astype('str'))
testD['Home_loan']=number.fit_transform(testD['Home_loan'].astype('str'))
```

```
In [25]: trainD['Education_loan']=number.fit_transform(trainD['Education_loan'].astype('str'))
testD['Education_loan']=number.fit_transform(testD['Education_loan'].astype('str'))
```

```
In [26]: trainD['Marital_status']=number.fit_transform(trainD['Marital_status'].astype('str'))
testD['Marital_status']=number.fit_transform(testD['Marital_status'].astype('str'))
```

```
In [27]: trainD.head()
```

Out[27]:

	ID	Default	Checking_amount	Term	Credit_score	Gender	Marital_status	Car_loan	Personal_loan	Home_loan	Education
0	101	0	988.0	15.0	796.0	0	1	1	0	0	
1	102	0	458.0	15.0	813.0	0	1	1	0	0	
2	103	0	158.0	14.0	756.0	0	1	0	1	0	
3	104	1	300.0	25.0	737.0	0	1	0	0	0	
4	105	1	63.0	24.0	662.0	0	1	0	0	0	

## Separating the Feature Variables and Target Variable

```
In [28]: X=trainD[['ID', 'Checking_amount', 'Term', 'Credit_score', 'Gender', 'Marital_status', 'Car_loan',
'Personal_loan', 'Home_loan', 'Education_loan', 'Emp_status', 'Amount', 'Saving_amount', 'Emp_dura
tion', 'Age', 'No_of_credit_acc']]
```

```
In [29]: y=trainD['Default']
```

## Split the train data into train-validation data

```
In [30]: #split X and y into training and testing datasets
from sklearn.model_selection import train_test_split
X_train,X_val,y_train,y_val=train_test_split(X,y,test_size=0.25,random_state=0)
```

## **8. Machine Learning Model**

### **8.1 Logistic Regression:**

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model (a form of binary regression). Mathematically, a binary logistic model has a dependent variable with two possible values, such as pass/fail which is represented by an indicator variable, where the two values are labeled "0" and "1". In the logistic model, the log-odds (the logarithm of the odds) for the value labeled "1" is a linear combination of one or more independent variables ("predictors"); the independent variables can each be a binary variable (two classes, coded by an indicator variable) or a continuous variable (any real value).

The corresponding probability of the value labeled "1" can vary between 0 (certainly the value "0") and 1 (certainly the value "1"), hence the labeling; the function that converts log-odds to probability is the logistic function, hence the name. The unit of measurement for the log-odds scale is called a logit, from logistic unit, hence the alternative names.

Analogous models with a different sigmoid function instead of the logistic function can also be used, such as the probit model; the defining characteristic of the logistic model is that increasing one of the independent variables multiplicatively scales the odds of the given outcome at a constant rate, with each independent variable having its own parameter; for a binary dependent variable this generalizes the odds ratio.

In a binary logistic regression model, the dependent variable has two levels (categorical). Outputs with more than two values are modeled by multinomial logistic regression and, if the multiple categories are ordered, by ordinal logistic regression (for example the proportional odds ordinal logistic model).

The logistic regression model itself simply models probability of output in terms of input and does not perform statistical classification (it is not a classifier), though it can be used to make a classifier, for instance by choosing a cutoff value and classifying inputs with probability greater than the cutoff as one class, below the cutoff as the other; this is a common way to make a binary classifier..



### 8.2 Random Forest:

Random forest is a supervised learning algorithm which is used for both classification as well as regression. But however, it is mainly used for classification problems. As we know that a forest is made up of trees and more trees means more robust forest. Similarly, random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting. It is an ensemble method which is better than a single decision tree because it reduces the over-fitting by averaging the result.

The random forest is a model made up of many decision trees. Rather than just simply averaging the prediction of trees (which we could call a “forest”), this model uses two key concepts that gives it the name *random*:

1. Random sampling of training data points when building trees
2. Random subsets of features considered when splitting nodes

#### **Random sampling of training observations:**

When training, each tree in a random forest learns from a random sample of the data points. The samples are drawn with replacement, known as bootstrapping, which means that some samples will be used multiple times in a single tree. The idea is that by training each tree on different samples, although each tree might have high variance with respect to a particular set of the training data, overall, the entire forest will have lower variance but not at the cost of increasing the bias.

#### **Random Subsets of features for splitting nodes:**

The other main concept in the random forest is that only a subset of all the features are considered for splitting each node in each decision tree. Generally this is set to  $\sqrt{n\_features}$  for classification meaning that if there are 16 features, at each node in each tree, only 4 random features will be considered for splitting the node. (The random forest can also be trained considering all the features at every node as is common in regression. These options can be controlled in the Scikit-Learn Random Forest implementation).

If you can comprehend a single decision tree, the idea of bagging, and random subsets of features, then you have a pretty good understanding of how a random forest works:

The random forest combines hundreds or thousands of decision trees, trains each one on a slightly different set of the observations, splitting nodes in each tree considering a limited number of the features. The final predictions of the random forest are made by averaging the predictions of each individual tree.

To understand why a random forest is better than a single decision tree imagine the following scenario: you have to decide whether Tesla stock will go up and you have access to a dozen analysts who have no prior knowledge about the company. Each analyst has low bias because they don't come in with any assumptions, and is allowed to learn from a dataset of news reports.

This might seem like an ideal situation, but the problem is that the reports are likely to contain noise in addition to real signals. Because the analysts are basing their predictions entirely on the data — they have high flexibility — they can be swayed by irrelevant information. The analysts might come up with differing predictions from the same dataset. Moreover, each individual analyst has high variance and would come up with drastically different predictions if given a different training set of reports.

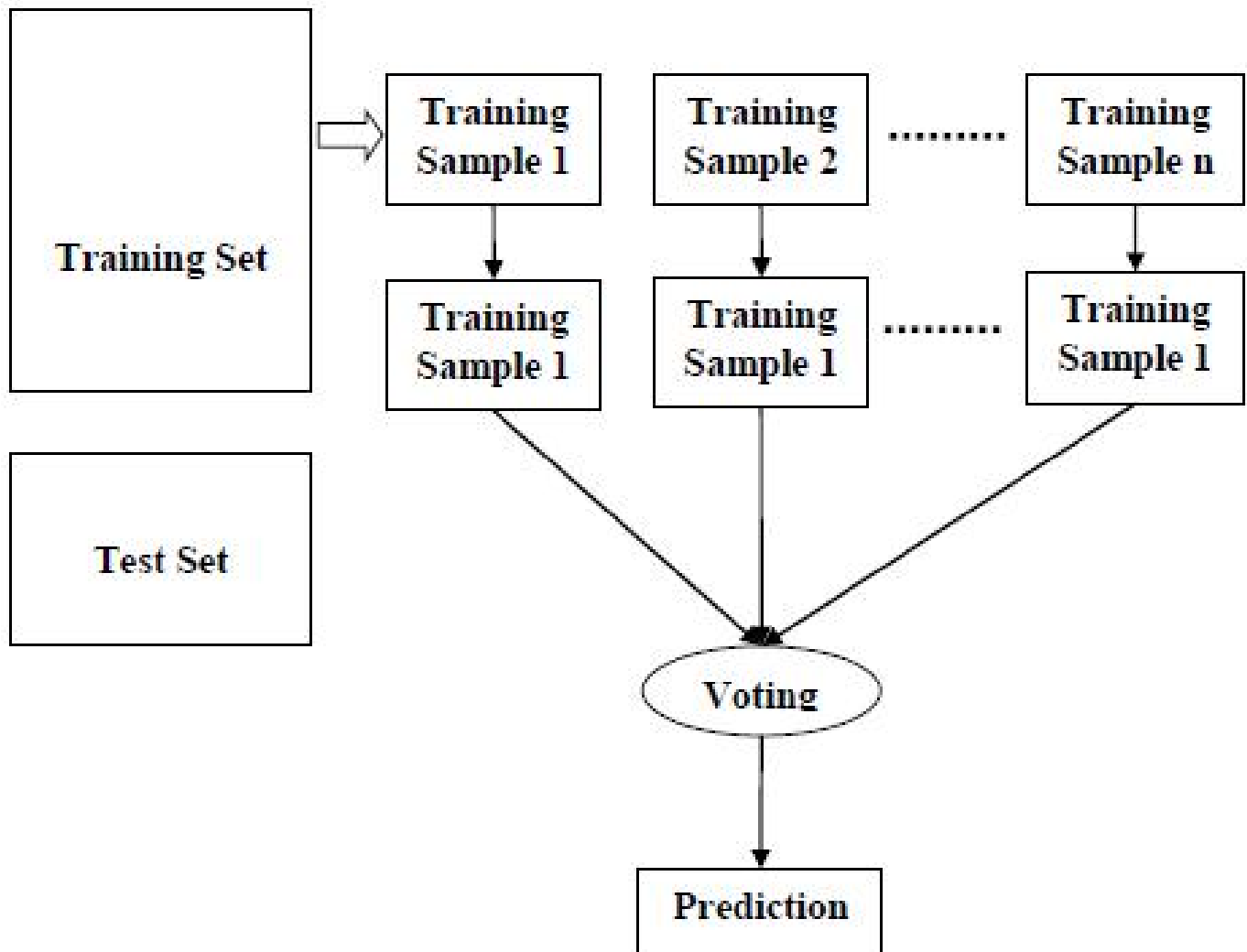
The solution is to not rely on any one individual, but pool the votes of each analyst. Furthermore, like in a random forest, allow each analyst access to only a section of the reports and hope the effects of the noisy information will be cancelled out by the sampling. In real life, we rely on multiple sources (never trust a solitary Amazon review), and therefore, not only is a decision tree intuitive, but so is the idea of combining them in a random forest.

### **Working of Random Forest Algorithm:**

We can understand the working of Random Forest algorithm with the help of following steps –

- **Step 1** – First, start with the selection of random samples from a given dataset.
- **Step 2** – Next, this algorithm will construct a decision tree for every sample. Then it will get the prediction result from every decision tree.
- **Step 3** – In this step, voting will be performed for every predicted result.
- **Step 4** – At last, select the most voted prediction result as the final prediction result.

The following diagram will illustrate its working –



The following are the advantages of Random Forest algorithm –

- It overcomes the problem of overfitting by averaging or combining the results of different decision trees.
- Random forests work well for a large range of data items than a single decision tree does.
- Random forest has less variance than single decision tree.
- Random forests are very flexible and possess very high accuracy.
- Scaling of data does not require in random forest algorithm. It maintains good accuracy even after providing data without scaling.

- Random Forest algorithms maintains good accuracy even a large proportion of the data is missing.

The following are the disadvantages of Random Forest algorithm –

- Complexity is the main disadvantage of Random forest algorithms.
- Construction of Random forests are much harder and time-consuming than decision trees.
- More computational resources are required to implement Random Forest algorithm.
- It is less intuitive in case when we have a large collection of decision trees.
- The prediction process using random forests is very time-consuming in comparison with other algorithms.

## 2. Random Forest

```
In [39]: from sklearn.ensemble import RandomForestClassifier

model2 = RandomForestClassifier(n_estimators=1000,random_state=20)

rf = model2.fit(X_train,y_train)
```

```
In [40]: prediction_RF = rf.predict(X_val)
```

```
In [41]: from sklearn import metrics

print("Accuracy=",metrics.accuracy_score(y_val, prediction_RF))
print("Precision:",metrics.precision_score(y_val,prediction_RF))
print("Recall:",metrics.recall_score(y_val,prediction_RF))

Accuracy= 0.9203980099502488
Precision: 0.8392857142857143
Recall: 0.8703703703703703
```

```
In [42]: # Confusion Matrix
cnf_matrix_RF =metrics.confusion_matrix(y_val,prediction_RF)
cnf_matrix_RF
```

```
Out[42]: array([[138,  9],
               [ 7, 47]], dtype=int64)
```

### Prediction on test data using Random Forest

```
In [44]: # Create a submission file
submission2 = pd.DataFrame()
testD2 = testD.copy()

testD2['Default'] = rf.predict(testD2)

submission2['ID'] = testD2['ID']
submission2['Default'] = testD2['Default']

# Submission file to csv
submission2.to_csv('RF_Predictions.csv', index=False)
submission2.head()
```

Out[44]:

	ID	Default
0	1001	1
1	1002	1
2	1003	0
3	1004	0
4	1005	0

### Happy Learning !!

## **9. Evaluation Model**

The above issues can be handled by evaluating the performance of a machine learning model, which is an integral component of any data science project. Model evaluation aims to estimate the generalization accuracy of a model on future (unseen/out-of-sample) data.

Methods for evaluating a model's performance are divided into 2 categories: namely, holdout and Cross-validation.

### **Holdout**

The purpose of holdout evaluation is to test a model on different data than it was trained on. This provides an unbiased estimate of learning performance.

In this method, the dataset is randomly divided into three subsets:

Training set is a subset of the dataset used to build predictive models.

Validation set is a subset of the dataset used to assess the performance of the model built in the training phase. It provides a test platform for fine-tuning a model's parameters and selecting the best performing model. Not all modeling algorithms need a validation set.

Test set, or unseen data, is a subset of the dataset used to assess the likely future performance of a model. If a model fits to the training set much better than it fits the test set, overfit

### **Cross-Validation**

Cross-validation is a technique that involves partitioning the original observation dataset into a training set, used to train the model, and an independent set used to evaluate the analysis.

The most common cross-validation technique is k-fold cross-validation, where the original dataset is partitioned into k equal size subsamples, called folds. The k is a user-specified number, usually with 5 or 10 as its preferred value. This is repeated k times, such that each time, one of the k subsets is used as the test set/validation set and the other k-1 subsets are put together to form a training set. The error estimation is averaged over all k trials to get the total effectiveness of our model.

For instance, when performing five-fold cross-valida.

## 9.1 Confusion Matrix :

In the field of machine learning and specifically the problem of statistical classification, a confusion matrix, also known as an error matrix. A confusion matrix is a table that is often used to describe the performance of a classification model (or “classifier”) on a set of test data for which the true values are known. It allows the visualization of the performance of an algorithm.

It allows easy identification of confusion between classes e.g. one class is commonly mislabeled as the other. Most performance measures are computed from the confusion matrix.

A confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix. The confusion matrix shows the ways in which your classification model is confused when it makes predictions. It gives us insight not only into the errors being made by a classifier but more importantly the types of errors that are being made.

	<i>Class 1 Predicted</i>	<i>Class 2 Predicted</i>
<b>Class 1 Actual</b>	TP	FN
<b>Class 2 Actual</b>	FP	TN

Here,

- Class 1 : Positive
- Class 2 : Negative

### Definition of the Terms:

- **Positive (P)** : Observation is positive (for example: is an apple).
- **Negative (N)** : Observation is not positive (for example: is not an apple).
- **True Positive (TP)** : Observation is positive, and is predicted to be positive.

- **False Negative (FN)** : Observation is positive, but is predicted negative.
- **True Negative (TN)** : Observation is negative, and is predicted to be negative.
- **False Positive (FP)** : Observation is negative, but is predicted positive.

### True Positive Rate (or) Recall:

Recall can be defined as the ratio of the total number of correctly classified positive examples divide to the total number of positive examples. High Recall indicates the class is correctly recognized (a small number of FN).

$$\text{Recall} = \frac{TP}{TP + FN}$$

### True Negative Rate:

True negative rate measures the proportion of actual negatives that are correctly identified as such (e.g., the percentage of healthy people who are correctly identified as not having the condition).

$$\text{TNR} = \frac{TN}{TN + FP}$$

### Positive predictive value:

The positive predictive value (PPV) is defined as

$$\text{PPV} = \frac{TP}{TP + FP}$$

where a "true positive(TP)" is the event that the test makes a positive prediction, and the subject has a positive result under the gold standard, and a "false positive(FP)" is the event that the test makes a positive prediction, and the subject has a negative result under the gold standard. The ideal value of the PPV, with a perfect test, is 1 (100%), and the worst possible value would be zero.

### Negative predictive value:

The negative predictive value is defined as:

$$\text{NPV} = \frac{TN}{TN + FN}$$

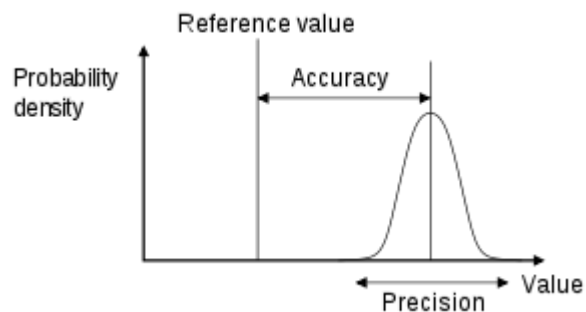


where a "true negative(TN)" is the event that the test makes a negative prediction, and the subject has a negative result under the gold standard, and a "false negative(FN)" is the event that the test makes a negative prediction, and the subject has a positive result under the gold standard. The ideal value of the NPV, with a perfect test, is 1 (100%), and the worst possible value would be zero.

## 9.2 Accuracy:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

However, there are problems with accuracy. It assumes equal costs for both kinds of errors. A 99% accuracy can be excellent, good, mediocre, poor or terrible depending upon the problem.



## 9.3 Precision:

Precision means the percentage of your results which are relevant. And Precision is the Proportion of true result over all positive results. It is also called Positive predictive Value.

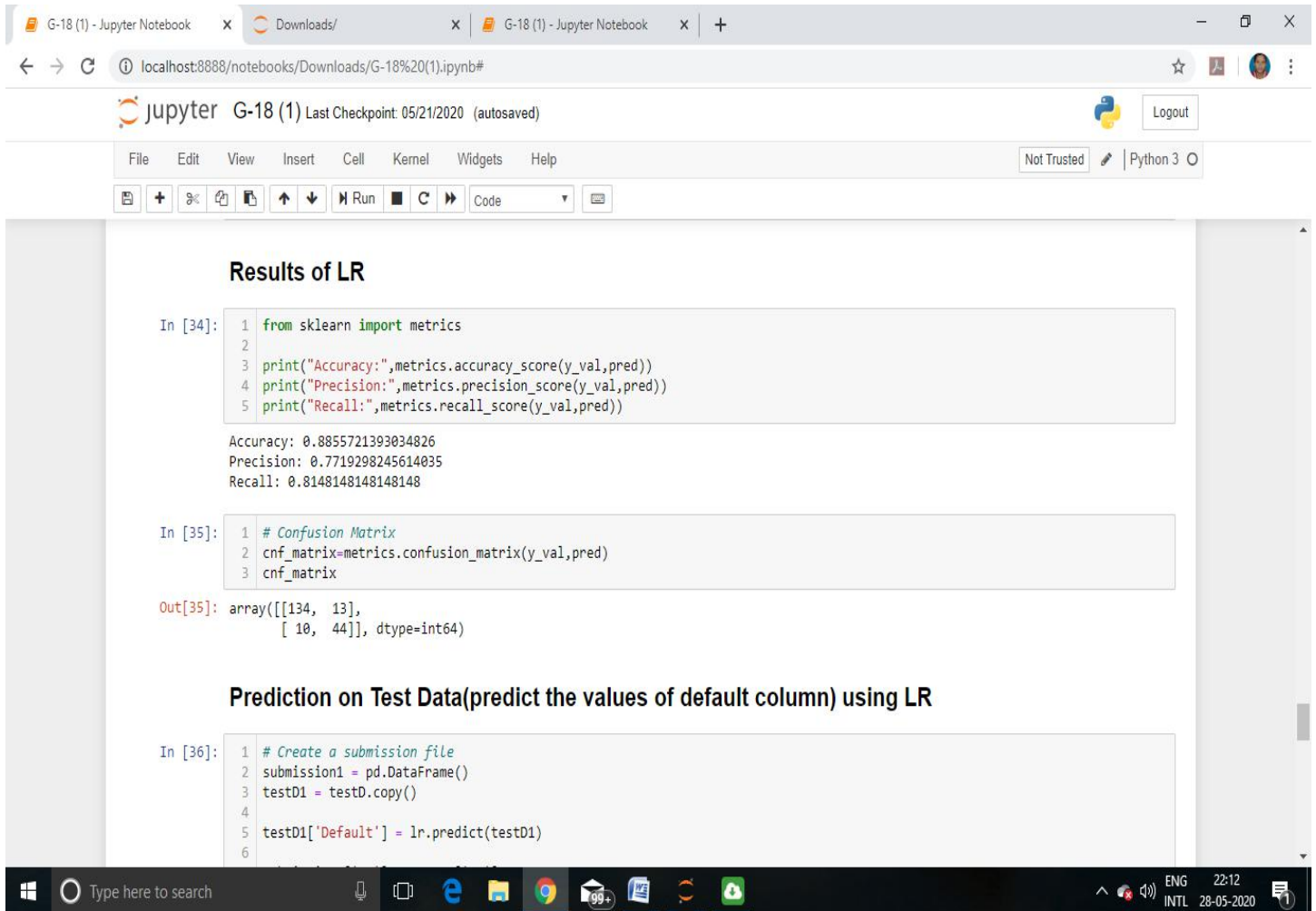
It is fraction of relevant instance.

- Precision = 
$$\frac{\text{True positive}}{\text{True positive} + \text{False Positive}}$$

### 9.4 Recall:

Recall is the fraction of all correct result returned by the model. It is also called Sensitivity. It is fraction of the total amount of relevant instance that were actually retrieved .

- Recall = 
$$\frac{\text{True positive}}{\text{True positive} + \text{False Positive}}$$



**Results of LR**

```
In [34]: 1 from sklearn import metrics
2         print("Accuracy:",metrics.accuracy_score(y_val,pred))
3         print("Precision:",metrics.precision_score(y_val,pred))
4         print("Recall:",metrics.recall_score(y_val,pred))

Accuracy: 0.8855721393034826
Precision: 0.7719298245614035
Recall: 0.8148148148148148
```

```
In [35]: 1 # Confusion Matrix
2         cnf_matrix=metrics.confusion_matrix(y_val,pred)
3         cnf_matrix

Out[35]: array([[134, 13],
                [ 10, 44]], dtype=int64)
```

**Prediction on Test Data(predict the values of default column) using LR**

```
In [36]: 1 # Create a submission file
2         submission1 = pd.DataFrame()
3         testD1 = testD.copy()
4
5         testD1['Default'] = lr.predict(testD1)
6
```

The screenshot shows a Jupyter Notebook with the following code cells:

```
In [33]: 1 pred = lr.predict(X_val)
```

```
1 ## Results of LR
```

```
In [34]: 1 from sklearn import metrics
2
3 print("Accuracy:", metrics.accuracy_score(y_val, pred))
4 print("Precision:", metrics.precision_score(y_val, pred))
5 print("Recall:", metrics.recall_score(y_val, pred))
```

Accuracy: 0.8855721393034826  
Precision: 0.7719298245614035  
Recall: 0.8148148148148148

```
In [35]: 1 # Confusion Matrix
2 cnf_matrix=metrics.confusion_matrix(y_val, pred)
3 cnf_matrix
```

```
Out[35]: array([[134, 13],
               [ 10, 44]], dtype=int64)
```

```
1 ## Prediction on Test Data(predict the values of default column) using LR
```

```
In [36]: 1 # Create a submission file
2 submission1 = pd.DataFrame()
3 testD1 = testD.copy()
4
5 testD1['Default'] = lr.predict(testD1)
6
```

## Logistic Model:

Let us try to understand logistic regression by considering a logistic model with given parameters, then seeing how the coefficients can be estimated from data. Consider a model With two prediction,  $x_1$  and  $x_2$ , and one binary response variables  $Y$ , which we denote  $p = P(Y=1)$ . We assume a linear relationship between the predictor variable and the log-odds of the events that  $Y=1$ . This linear relationship can be written in the following mathematical form (where  $\ell$  is the log-odds,  $b$  is the of the logarithm, and  $\beta_i$  are parameters of the model):

$$\ell = \log \frac{p}{1-p} = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

We can recover the odds by exponentiating the log-odds:

$$\frac{p}{1-p} = b^{\beta_0 + \beta_1 x_1 + \beta_2 x_2}$$

By simple algebraic manipulation, the probability that  $Y=1$  is

$$p = \frac{b^{\beta_0 + \beta_1 x_1 + \beta_2 x_2}}{b^{\beta_0 + \beta_1 x_1 + \beta_2 x_2} + 1} = \frac{1}{1 + b^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2)}}$$

The above formula shows that once  $\beta_i$  are fixed, we can easily compute either the log-odds that  $Y=1$  for a given observation, or the probability that  $Y=0$  for a given observation. The main use-case of a logistic model is to be given an observation  $(x_1, x_2)$ , and estimate the probability  $p$  that  $Y=1$ . In most applications, the base  $b$  of the logarithm is usually taken to be  $e$ . However in some cases it can be easier to communicate results by working in base 2, or base 10.

## 1. Logistic Regression

```
In [31]: #import classes
from sklearn.linear_model import LogisticRegression
model1 = LogisticRegression()
```

### Train the model

```
In [32]: lr = model1.fit(X_train,y_train)

E:\Anaconda\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning: Default so
lver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

### predict on validation data to check accuracy

```
In [33]: pred = lr.predict(X_val)
```

### Results of LR

```
In [34]: from sklearn import metrics

print("Accuracy:",metrics.accuracy_score(y_val,pred))
print("Precision:",metrics.precision_score(y_val,pred))
print("Recall:",metrics.recall_score(y_val,pred))
```

```
Accuracy: 0.8855721393034826
Precision: 0.7719298245614035
Recall: 0.8148148148148148
```

```
In [35]: # Confusion Matrix
cnf_matrix=metrics.confusion_matrix(y_val,pred)
cnf_matrix
```

```
Out[35]: array([[134,  13],
               [ 10,  44]], dtype=int64)
```

### Prediction on Test Data(predict the values of default column) using LR

```
In [36]: # Create a submission file
submission1 = pd.DataFrame()
testD1 = testD.copy()

testD1['Default'] = lr.predict(testD1)

submission1['ID'] = testD1['ID']
submission1['Default'] = testD1['Default']

# Submission file to csv
submission1.to_csv('LR_Predictions.csv', index=False)
submission1.head()
```

Out[36]:

	ID	Default
0	1001	1
1	1002	1
2	1003	0
3	1004	1
4	1005	0

### 10. Conclusion:

The future of machine learning in the banking and financial industry is well recognised, and it is expected that the field of risk management will also seek to apply machine learning techniques to enhance their capabilities. One could attribute this to the fact that credit risks is considered the most significant risk to a banking organisation. More specifically from a methodology perspective, credit risk management problems researched have been around credit scoring; it would go a long way to research how machine learning can be applied to quantitative areas for better computations of credit risk exposure by predicting probabilities of default, loss given default given the many complexities and the varied factors that are involved. while there has been research on the application of machine learning in risk management over the years, it still falls short and is not on par across the various areas of risk management or risk methodologies.

There still remain a large number of areas as highlighted above in bank risk management that could significantly benefit from study on how machine learning can be applied to address specific problems.