Credit Card Fraud Detection System : A comparison study on various Machine Learning and Deep Learning technique(s)

Soubhik Sinha[1], Rohit K[2], Adarsh Kumar Singh[3]

[1,2,3] Department of Information Technology, VIT University, Vellore, Tamil Nadu, India

Abstract

Usage of liquid cash is getting neglected day-by-day because of the increased usage of credit card payments method. There are a few countries which are completely relying upon digital transactions , omitting paper currency. But that doesn't mean this kind of payment method are completely reliable. As the cybersecurity field is excelling at a rapid pace , accessing a bank account is not a tough job to do today. Thus , Credit Card fraud cases are also piling up in every developed and developing country. This project aims to compare all the trending efficient methods against Credit Card Frauds , by training the model using dataset (from Kaggle). The model consists of 8 modules / techniques namely – CNN (Convolutional Neural Network) , Decision Tree Algorithm , K-NN (K – Nearest Neighbour) , SVM (Support Vector Machine) , Naïve Bayes Classifier , Random Forest Algorithm , ANN (Artificial Neural Network) , Confusion Matrix. The efficiency of the model developed shall be checked by the metrics like – Accuracy , Precision , Recall (Sensitivity) , F-score. Later , the model results shall be compared with similar papers / projects with already existing and operational model to check and conclude the most efficient method (as per the metrics) among those , discussed here.

Keywords – Comparison , Metrics , Accuracy , Precision.

I. INTRODUCTION

Payments of Goods and Services are becoming more and more elephantine. Whether you go to shopping for garments , want to have / pick quick snacks at your favorite food chains OR eateries , halting at the gas station for refueling , etc. , digital payments have dominated the way we all made payments a few decades back. The first digital payment method dates back to early 1870s , termed as Electronic Fund Transfer (EFT) – a small technological move towards the omittance of Liquid Money , or in other words , Paper money (cash). But now , the world is living the 21st century , which most people often whooped "The Era of Technology". Almost Every person today has a bank account , which can be accessed via Internet. But you won't be doing an Online Banking Transaction for , say , buying a pair of Shoes at a footwear store. You will be needing a more convenient way with minimalistic GUI interface. Here comes Credit / Debit Cards. These are made of complex materials of plastic and PVC and silicon plates / coatings and coded / programmed which can be swiped through hand-held machines / devices , which directly deducts the amount from the bank account for the goods / service.

Now let us talk about the handicapped part of Credit Cards. Though people use it heavily , but they are unaware of the upcoming problems. Many stores use cloning machines which copies all the data of the card , giving

access to the person s' bank account – resulting to a huge fraud. Likewise , shopkeepers are also dupped by false credit cards.

Fraudsters never let an opportunity go to waste – whether the whole world is fighting a war , whether any country is facing economic crisis – fraudsters have only one motivation – getting the money. For the past decade , credit card fraudulent cases have rose by 400% (if counted and checked annually). People with newer accounts are duped more than the older / existing ones by 24%. In 2019 , more than 1500 data breaches occurred and millions of user records were exposed.

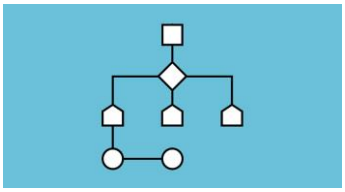
Earlier , Cards had Magnetic Strips which contained card holders' information (they are still in use). But now , Smart Card has emerged as a new hope of secured transaction. These do not contain any magnetic strips , rather they are embedded with Integrated Circuits (ICs) – which enables us to just touch / tap the card to the machine without any swiping and password / pin. Despite of so many technological advancements in terms of cybersecurity and secured banking systems , mishaps occur almost every day with someone at any moment of time.

Though the chances are good enough to get duped , there are many machine learning algorithms / techniques which can be put on field to work and subjugate this pest.

II. BACKGROUND

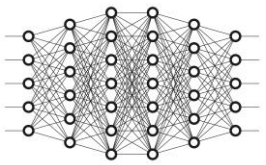
Let's discuss the concepts and key terms you may pass by in this paper.

The word Algorithm signifies "a cycle or set of rules to be continued in computations or other critical thinking tasks". Thusly Algorithm alludes to a bunch of rules/directions that bit by bit characterize how a work is to be executed upon to get the normal outcomes. The Algorithm planned are language-free, for example they are outright directions that can be carried out in any language, but the yield will be something similar, true to form.



Machine Learning (ML) is the field of study that gives PCs the capacity to learn without being expressly customized. ML is perhaps the most energizing advancements that one would have at any point gone over. As it is obvious from the name, it gives the PC that makes it more like people: The capacity to learn. ML is effectively being utilized today, maybe in a lot a bigger number of spots than one would anticipate.

Deep Learning is a subset of Machine Learning, which then again is a subset of Artificial Intelligence. Computerized reasoning is an overall term that alludes to procedures that empower PCs to copy human conduct. AI addresses a bunch of calculations prepared on information that make the entirety of this conceivable. Deep Learning, then again, is only a kind of Machine Learning, enlivened by the design of a human cerebrum (brain). Deep Learning calculations endeavor to reach comparable inferences as people would by consistently dissecting information with a given intelligent construction. To accomplish this, profound learning utilizes a diverse construction of calculations called NN (Neural Network).



Classification is the way toward foreseeing the class of given information focuses. Classes are now and then called as targets/names or classifications. Arrangement prescient demonstrating is the errand of approximating a planning capacity (f) from input factors (X) to discrete yield factors (Y). Classification has a place with the class of directed realizing where the objectives additionally furnished with the info information. There are numerous applications in arrangement in numerous areas, for example, in credit endorsement, clinical conclusion, target promoting and so on.

Random Forest Algorithm

Random Forest is a famous ML calculation that has a place with the directed learning strategy. It tends to be utilized for both Classification and Regression issues in ML. It depends on the idea of troupe realizing, which is a cycle of consolidating different classifiers to take care of an unpredictable issue and to improve the presentation of the model. As the name proposes, "Random Forest is a classifier that contains various Decision Trees on different subsets of the given dataset and takes the normal to improve the prescient exactness of that dataset." Instead of depending on one choice tree, the Random Forest takes the forecast from each tree and dependent on the greater part votes of expectations, and it predicts the last yield.

CNN – Convolutional Neural Network

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning Algorithm which can take in an information picture, allot significance (learnable loads and predispositions) to different viewpoints/objects in the picture and have the option to separate one from the other. The pre-preparing needed in a ConvNet is a lot of lower when contrasted with other order calculations. While in crude strategies channels are hand-designed, with enough preparing, ConvNets can become familiar with these channels/qualities. The design of a ConvNet is closely resembling that of the network example of Neurons in the Human Brain and was roused by the association of the Visual Cortex. Singular neurons react to improvements just in a confined district of the visual field known as the Receptive Field. An assortment of such fields covers to cover the whole visual region.

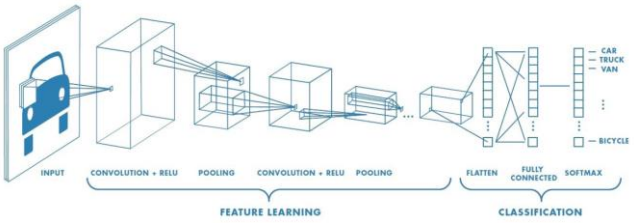


Fig. : working of CNN (source : www.towardsdatascience.com)

Decision Tree Algorithm

Decision Tree Algorithm has a place with the group of administered learning calculations. In contrast to other administered learning calculations, the choice tree calculation can be utilized for taking care of relapse and order issues as well. The objective of utilizing a Decision Tree is to make a preparation model that can use to foresee the class or worth of the objective variable by taking in basic choice principles derived from earlier data(training information). In Decision Trees, for anticipating a class name for a record we start from the base of the tree. We think about the upsides of the root property with the record's quality. Based on correlation, we follow the branch comparing to that worth and leap to the following hub.

K-NN (K – Nearest Neighbour)

The K-Nearest Neighbors (KNN) algorithm is a basic, simple to-execute managed AI calculation that can be utilized to tackle both Classification and regression issues. KNN works by discovering the distances between a question and every one of the models in the information, choosing the predetermined number models (K) nearest to the inquiry, at that point votes in favor of the most incessant mark (on account of order) or midpoints the names (on account of relapse).

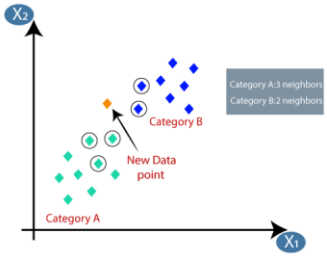


Fig : Graph after the implementation of K-NN

SVM (Support Vector Mechanism)

Support Vector Machine / Mechanism or SVM is quite possibly the most mainstream Supervised Learning algorithms, which is utilized for Classification just as Regression issues. Notwithstanding, essentially, it is utilized for Classification issues in Machine Learning. The objective of the SVM algorithm is to make the best line or choice limit that can isolate n-dimensional space into classes so we can without much of a stretch put the new information point in the right classification later on. This best choice limit is known as a hyperplane.

SVM picks the limit focuses/vectors that assistance in making the hyperplane. These limit cases are called as help vectors, and subsequently calculation is named as Support Vector Machine. Consider the beneath graph in which there are two unique classifications that are grouped utilizing a choice limit or hyperplane.

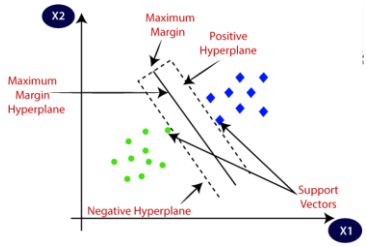


Fig. : SVM Graphical representation

Naïve Bayes Classifier

A Classifier is an ML model that is utilized to separate various items dependent on specific highlights. A Naive Bayes classifier is a probabilistic ML model that is utilized for arrangement task. The essence of the classifier depends on the Bayes hypothesis.

Bayes Hypothesis –

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Utilizing Bayes hypothesis, we can discover the likelihood of an event (A), given that B has happened. Here, B is the proof and A is the theory. The suspicion made here is that the indicators/highlights are free. That is presence of one specific component doesn't influence the other. Consequently it is called Naïve.

ANN (Artificial Neural Network)

An Artificial Neural Network (ANN) is the piece of a computing framework intended to reproduce the manner in which the human mind dissects and measures data. It is the establishment of man-made consciousness (AI) and takes care of issues that would demonstrate unthinkable or troublesome by human or measurable principles. ANNs make them learn capacities that empower them to deliver better outcomes as more information opens up.

An ANN has hundreds or thousands of artificial neurons called handling units, which are interconnected by hubs. These preparing units are comprised of information and yield units. The information units get different structures and constructions of data dependent on an interior weighting framework, and the neural network endeavors to find out about the data introduced to deliver one yield report. Actually like people need rules and rules to concoct an outcome or yield, ANNs additionally utilize a bunch of learning rules called backpropagation, a shortened form for in reverse engendering of blunder, to consummate their yield results.

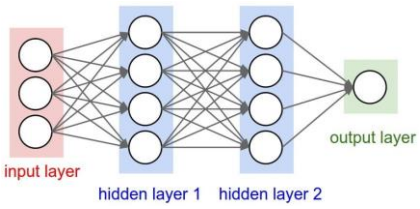


Fig.: ANN model

Confusion Matrix

Confusion Matrix is a table that is frequently used to depict the presentation of an order model (or "classifier") on a bunch of test information for which the genuine qualities are known. This itself is moderately easy to see, yet the connected phrasing can be befuddling. There are some basic terms which are included in the making of the matrix. We shall discuss this with respect to a patient having a disease or not (for convention) ---

1. True Positives (TP) : These are cases in which we predicted yes (they have the disease), and they do have the disease.

2. True Negatives (TN) : We predicted no, and they don't have the disease.

3. False Positives (FP) : We predicted yes, but they don't actually have the disease. (Also known as a "Type I error.").

4. False Negatives (FN) : We predicted no, but they actually do have the disease. (Also known as a "Type II error.").

Now let us talk about the evaluation metrics (the ones specially which are mentioned / worked with in this paper) –

Accuracy : It's the ratio of correctly predicted observation to the total observations.

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

Precision : Precision is the ratio of correctly predicted positive observations to the total predicted positive observations.

$$\text{Precision} = TP / TP+FP$$

Recall (Sensitivity) : Recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes.

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

F1 Score : F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall.

$$\text{F1 Score} = 2*(\text{Recall} * \text{Precision}) / (\text{Recall} + \text{Precision})$$

III. Literature Survey

After hunting numerous articles , the following text describes about each of the article's approach towards the modern-day digital transaction problem.

[1]. This paper talks about designing and developing a novel fraud detection method for Streaming Transaction Data, with an objective, to analyze the past transaction details of the customers and extract the behavioral patterns. Using Sliding window strategy , transactional information can be accumulated for the extraction of behavioral pattern.

[2]. In this paper, it's representing a hybrid data mining/complex network classification algorithm, able to detect illegal instances in a real card transaction data set. It is based on a recently proposed network reconstruction algorithm that allows creating representations of the deviation of one instance from a reference group.

[3]. In this survey, they have studied data-driven credit card fraud detection particularities and several machine learning methods to address each of its intricate challenges with the goal to identify fraudulent transactions that have been issued illegitimately on behalf of the rightful card owner. The approaches range from feature engineering techniques (transactions aggregations for example) to proper sequence modeling methods such as recurrent neural networks (LSTM) or graphical models (hidden markov models).

[4]. This paper investigates and checks the performance of Decision tree, Random Forest, SVM and logistic regression on highly skewed credit card fraud data. These techniques are applied on the raw and preprocessed data. The performance of the techniques is evaluated based on accuracy, sensitivity, specificity, precision.

[5]. This paper intends to illustrate the modelling of a data set using machine learning with Credit Card Fraud Detection. The Credit Card Fraud Detection Problem includes modelling past credit card transactions with the data of the ones that turned out to be fraud. This model is then used to recognize whether a new transaction is fraudulent or not. In this process, they have focused on analysing and pre-processing data sets as well as the deployment of multiple anomaly detection algorithms such as Local Outlier Factor and Isolation Forest algorithm on the PCA transformed Credit Card Transaction data.

[6]. This work , shows us a survey of financial fraud methods using machine learning and deep learning methodology, and proposed a process for accurate fraud detection based on the advantages and limitations of each research. The approach has proposed the overall process of detecting financial fraud based on machine learning and compared with artificial neural networks approach to detect fraud and process large amounts of financial data. To detect financial fraud and process large amounts of financial data, the proposed process includes feature selection, sampling, and applying supervised and unsupervised algorithms.

[7]. In this paper, after investigating difficulties of credit card fraud detection, it has been reviewed the state of the art in credit card fraud detection techniques, datasets and evaluation criteria. The advantages and disadvantages of fraud detection methods are enumerated and compared. A classification of mentioned techniques into two main fraud detection approaches, namely, misuses (supervised) and anomaly detection (unsupervised) is presented.

[8]. In this paper, it is introduced the concept of frauds related to credit cards and their various types. We have explained various techniques available for a fraud detection system such as Support Vector Machine (SVM), Artificial Neural Networks (ANN), Bayesian Network, K- Nearest Neighbor (KNN), Hidden Markov Model, Fuzzy Logic Based System and Decision Trees.

[9]. This paper, in terms of certain parameters , various credit card fraud detection technique are reviewed. As we all know , a very little amount of information is required by the attacker for conducting any fraudulent transaction in online transactions. Here , many such pest-killing techniques have been discussed and compared.

[10]. This paper describes probability of fraudulent transactions in prevalence and context of credit card usage. A conscious effort is put to bring in a conceptual distinction between fraud detection and predicting probable fraudulent opportunities on the digital space of financial transactions.

[11]. This paper aims to 1) focus on fraud cases that cannot be detected based on previous history or supervised learning, 2) create a model of deep Auto-encoder and Restricted Boltzmann machine (RBM) that can reconstruct normal transactions to find anomalies from normal patterns. The proposed deep learning based on auto-encoder (AE) is an unsupervised learning algorithm that applies backpropagation by setting the inputs equal to the outputs.

[12]. This paper showed the utilization of the supervised learning algorithms which are implemented on a dataset which was highly skewed and imbalance. The set was balanced by robust scalar to have a 51 percent non fraud cases and 49 percent fraud cases. Logistic regression, random forest, decision tree and KNN has been implemented and further learning curves are displayed which shows which algorithm has the best ability to perform.

[13]. This paper is mainly focusing on detecting credit card fraud in the real world - random forest algorithm classification method using the already evaluated data set and providing current data set. The results obtained with the use of the Random Forest Algorithm have proved much more effective.

[14]. This paper investigates and checks the performance of Random Forest Classifier, AdaBoost Classifier, XGBoost Classifier and LightGBM Classifier on highly skewed credit card fraud data. These techniques are applied on the raw and preprocessed data. The performance of the techniques is evaluated based on accuracy, sensitivity, specificity, precision.

[15]. In this paper , 14 different techniques are introduced for how data mining techniques can be successfully combined to obtain a high fraud coverage with a high or low false rate, the Advantage and The Disadvantages of every technique, and The Data Sets used in the researches by researchers.

[16]. The main objective of this paper is to review methodology of different detection methods based on credit card in terms of Parameter like Speed of detection, Accuracy and cost the comparison of mentioned approaches based on survey. This paper presents a survey of various techniques used in credit card fraud detection and prevention.

[17]. This presented paper focuses on fraud activities that cannot be detected manually by carrying out research and examine the results of logistic regression, decision tree and support vector machine.

[18]. The main aim is to detect the fraudulent transaction and to develop a method of generating test data. This algorithm is a heuristic approach used to solve high complexity computational problems. It is an optimization technique and evolutionary search based on the genetic and natural selection.

[19]. The assessment of Light Gradient Boosting Machine model which proposed in the paper depends much on datasets collected from clients' daily transactions. Besides, to prove the new model's superiority in detecting credit card fraudulence, Light Gradient Boosting Machine model is compared with Random Forest and Gradient Boosting Machine algorithm in the experiment.

[20]. In the Current Fraud Detection framework, false exchange is recognized after the transaction is completed. As opposed to the current system, the proposed system presents a methodology which facilitates the detection of fraudulent exchanges while they are being processed, this is achieved by means of Behaviour and Locational Analysis(Neural Logic) which considers a cardholder's way of managing
money and spending pattern.

[21]. As credit card becomes the most popular mode of payment for both online as well as regular purchase, cases of credit card fraud also rising. Data mining techniques could be used to detect the credit card fraud detection. The main goal of this paper is comparing the data mining techniques, such as Simple K-means, Hidden Markov Model, Bayesian Network, KNN algorithm and Outlier detection.

[22]. In this paper, they have discussed a big data analytical framework to process large volume of data and implemented various machine learning algorithms for fraud detection and observed their performance on benchmark dataset to detect frauds on real time basis there by giving low risk and high customer satisfaction.

[23]. This paper proposes an intelligent approach for detecting fraud in credit card transactions using an optimized light gradient boosting machine (OLightGBM). In the proposed approach, a Bayesian-based hyperparameter optimization algorithm is intelligently integrated to tune the parameters of a light gradient boosting machine (LightGBM).

[24]. In this work, they have provided a survey of current techniques most relevant to the problem of credit card fraud detection. In this they have focused on studies utilizing classical machine learning models, which mostly employ traditional transnational features to make fraud predictions. These models typically rely on some static physical characteristics, such as what the user knows (knowledge-based method), or what
he/she has access to (object-based method).

[25]. In this paper they have discussed that most of these transactions are processed by machine without any human interfere. In past years, data mining is applied to transaction frauds tasks. Data mining is a process to extract interesting patterns from metadata which can be fitted to machine learning models like logistic regression, Naïve Bayes and support vector machine etc.

[26]. In this paper they have research about the various detecting techniques to identify and detect the fraud through varied techniques of data mining. The aim is to first of all establish the categories of the fraud secondly, the techniques like K-nearest neighbor, Hidden Markov model, SVM, logistic regression, decision tree and neural network.

[27]. In this work, a Gradient Boosting Tree (GBT) model for the real-time detection of credit card frauds on the streaming Card-Not-Present (CNP) transactions is investigated with the use of different attributes of card transactions. In the experiments, the feature engineering strategy and the automated training set generation methodology are evaluated on the real credit card transactions.

[28]. In this paper we expand the transaction aggregation strategy, and propose to create a new set of features based on analyzing the periodic behavior of the time of a transaction using the von Mises distribution.

[29]. This article presents a few machines learning models to detect such fraud. They firstly explore and clean up the data. Then 331 expert variables are created with professional consult and selected to 30 to reduce dimensionality of our data. Multiple models, such as logistic regression and decision trees, are built and fit on the training set. Finally, we found that the random forest model performs the best in terms of fraud detection rate, achieving 54% in out-of-time test.

[30]. As a result, it tends to misrepresent a fraudulent transaction as a genuine transaction. To tackle this problem, data-level approach, where different resampling methods such as under-sampling, oversampling, and hybrid strategies, have been implemented along with an algorithmic approach where ensemble models such as bagging and boosting have been applied to a highly skewed dataset containing numerous transactions.

Also use table for literature summary(reference).

The following table(s) categorizes all the techniques and methods / methodology under Supervised and unsupervised learning algorithms.

ARTICLES / STUDIES BASED ON SUPERVISED LEARNING ALGORITHMS

Authors	Year	Methodology and Techniques used	Advantages	Issues	Metrics Used
Vaishnavi Nath Dornadula Geetha S	2019	Sliding-Window Method	Lead the system to adapt to new cardholder's transaction behaviours timely	To design and develop a novel fraud detection method for Streaming Transaction Data, with an objective, to analyse the past transaction details of the customers and extract the behavioural patterns.	Accuracy Precision Mathews Correlational Coeff. (MCC)
Massimiliano Zanin Miguel Romance Santiago Moral Regino Criado	2018	Parenclitic networks Artificial Neural Networks --- (ANN) Multi-Layer Perceptrons (MLP)	Increased efficiencyin detecting frauds in some niches of operations, like medium-sized and on-line transactions.	To detect illegal instances in a real card transaction dataset.	True Positive Rate (TPR) Receiver Operating Characteristic (ROC) False Positive Ratio (FTR)

Dr . Yvan Lucas Dr . Johannes Jurgovsky	2020	Feature engineering techniques Recurrent Neural Networks **(LSTM) – RNN** Graphical Models (Random fore)	Making a strong difference for fraudulent transactions , whichare much more rarethan genuine transactions. Allowing Fraud detection systems to obtain good performances	To identify fraudulent transactions that have been issue illegitimately on behalf of the rightful card owner.	Precision True Positive Rate (TPR) Accuracy F1 Score Mathews Correlational Coeff. **(MCC)**
Navanshu Khare Saad Yunus Sait	2018	Decision Tree Random Forest (SUP) SVM Model -- Support Vector Machine Logistic Regression	It's known that Random Forest algorithm will perform better with a larger number of training data.	To check the performance of Decision Tree , Random Forest , SVM and Logistic Regression on highly skewed Credit Card Fraud Data.	Accuracy Sensitivity Specificity Precision
Dahee Choi Kyunghoo Lee	2018	Machine Learning Deep Learning Feature Selection Sampling Supervised Algorithms Unsupervised Algorithms HMM -- Hidden Markov Model	ML based method has higherdetection than neural. Networks at various ratios Neural Networks gets accuracy as high as 95%.	Proposing a process for accurate fraud detection -- the overall process of detecting financial fraud based on ML andcomparing it with ANN approach to detect fraud and process large amount of financial data.	Accuracy F- measure

Samaneh Sorournejad Zahra Zojaji Reza Ebrahimi Atani Amir Hassan Monadjemi	2016	Misuse detection ANN - Artificial Neural Network Supervised Techniques Unsupervised Techniques Negative Selection Algorithm Genetic Algorithm HMM -- Hidden Markov Model SVM	Efficient Accuracy and Maintainability , Improved Capability of Pattern Recognition Improved Working with Noisy Data.	To review the state of the art in credit card fraud detection techniques, datasets and evaluation criteria.	Accuracy Precision / Hit Rate True Positive Rate / Sensitivity False Positive Rate ROC Cost F1 measure
		Bayesian Network Fuzzy Neural Network			
Yashvi Jain NamrataTiwari ShripriyaDubey Sarika Jain	2019	Support VectorMachine (SVM) Artificial Neural Networks (ANN) Bayesian Network K- Nearest Neighbour (KNN) Hidden Markov Model Fuzzy Logic Based System Decision Trees	Creation of hybrid of various techniques that arealready used in fraud detection to cancel out them limitations and get enhanced performance.	To introduce the concept of fraud related to credit card(s) and them various types and their solutions - valid against modern tricks and Hacks.	Accuracy Detection Rate False Alarm Rate Sensitivity Specificity Cost

Sonal Mehndiratta Kamal Gupta	2019	Data Mining Machine Learning Classification ANN - Artificial Neural Network Genetic Algorithm (GA) HMM -- Hidden Markov Model K- Nearest Neighbour (KNN) Naive Bayes	Majority of voting methods achieve good accuracy rates in order to detect thefraud in the credit cards. Proposed random forests provide good results on the small dataset. There is a large moving window, higher number of attributes and number of link types available which can be searched by CD and SD algorithms.	To avoid the leakage of information to the attacker , which if fails willlead to huge amounts of loss to the Credit card company.	Precision Sensitivity Accuracy Balanced Classification Rate

Kaithekuzhical Leena Kurien Dr . Ajeet Chikkamannur	2019	Machine Learning Artificial Intelligence (AI) Deep Learning	Efficient Over Sampling , Under-sampling of data for accuracy. Thorough Analysis of the features and sub-sample ratios for imbalanced Datasets.	Probability of fraudulent transactions in prevalence and context of credit card usage.	Feature Co-Relation Quartile True positive(TP) True Negative(TN) False positive(FP) False negative(FN) Precision F1 Score Recall

		Support Vector Machine Genetic Algorithm Bayesian Network	By using KNN, it acts as a better classifier for credit card detection.	Solution to the fraud.	Specificity Sensitivity

Apapan Pumsirirat Liu Yan	2018	Creating an Auto Encoder using Deep Learning TensorFlow (google library)	Accurately achieve credit carddetection with a large dataset. Guarantee thatAE and RBM can make more accurate AUC for receiver operator characteristics.	To focus on fraud cases that cannot be detected basedon previous history or supervised learning. To create a model of deep Auto-encoder and Restricted Boltzmann Machine (RBM) that can reconstruct normal transactions to find anomalies from normal patterns.	MSE RMSE
Daniyal Baig	2020	Artificial Neural Network	Gives the highest accuracy ascompared to other logistic regressions.	To identify the Credit card fraudand provide a reasonable	Accuracy Precision

S. Abinayaa H. Sangeetha R.A. Karthikeyan K. Saran Sriram D. Piyush	2020	Random Forest Algorithm Machine Learning	By using machine Learning alongside Random Forest Algorithm we can get a better result in the detection of fraud.	To find the most common methods of fraud alongside their detection methods and algorithms.	Sensitivity Precision Accuracy
Nishant Sharma	2019	Random Forest Algorithm AdaBoost Classifier XGBoost Classifier LightGBM Classifier	By this test we can conclude that XGBoost Classifier provides the highest accuracy on credit card fraud detection.	To investigate and check the performance of mentioned techniques onhighly skewed credit card fraud data.	Accuracy Performance
Wael Khalifa Mohamed Ismail RoushdyAbdel-Badeeh M. Salem Hossam Eldin Hossam Eldin Mohammed Abd El-Hamid	2019	Machine Learning Data Mining Support Vector Machine Bayesian Network	By this research survey, we can figure out which technique is the best to detect credit card fraud and which technique can be inour system.	The goal of this research is to survey procedure of various discoverytechniques dependent on Visa.	Accuracy Speed Precision Exactness Cost
Surbhi Gupta Nitima Malsa Vimal Gupta	2017	Artificial Immune System (AIS) Neural Network Genetic Algorithms Decision Tree	By this survey,you can find the differences between each technique mentioned. New classifierscan be formed by combining the	The goal is to compare different techniques by either testing them separately or by combining and forming new classifiers which	Speed of detection Accuracy Cost
		Support Vector Machine (SVM)	different techniques.	can be used to improve the detection of credit card fraud.	
Shiv Shankar Singh	2019	Data Mining Logistic RegressionDecision tree Support Vector Machine (SVM)	This survey makes sure that more businesses are aware of the different techniques available to catch credit card fraud.	This presented paper focuses onfraud activities that cannot be detected manually by carrying out research and examine the results of logisticregression, decision tree and support vector machine	Precision Cost Exact outcome Rapidly train machines

Ishu Trivedi Monika Mrigya Mridushi	2016	Artificial Neural network Genetic Algorithms Neural Network Bayesian Network	This method proves accurate infinding out the fraudulent transactions and minimizing the number of false alerts. probability of detecting fraud in avery short span of time after the transactions has been made.	The main aim isto detect the fraudulent transaction andto develop a method of generating test data.	Time Speed Accuracy Rate of Error
Yong Fang Yunyun Zhang Cheng Huang	2019	Light-GBM model Imbalanced data Random Forest Algorithm Gradient Boosting Machine Algorithm (GBM)	The result indicatesthat the new modelis fast, has a lesser error rate and is more accurate.	The goal is to compare the older system with the new system and see their differences	Accuracy Efficiency Recall Rate Speed
Aman Gulat Prakash Dubey Md Fuzail C Jasmine Norman Mangayarkarasi R	2017	Behaviour and Locational Analysis (Neural Logic) Neural Network Geolocation	Has an accuracyof up to 80% in detecting credit card fraud. The geolocation of the fraudster canbe pinpointed	The goal here is to create a Credit card frauddetection systemusing the techniques of neural networks and geolocation.	Accuracy Rate of Detection
N. Geetha T. Kavipriya	2017	Neural Network Bayesian Network Genetic Algorithm Support Vector Machine Decision Tree Fuzzy Logic Based System	Our goal is to analyze different data mining techniques in a waythat they help us to detect and predict the credit card fraud. Analysis presented by different researcher's shows that different data mining techniques. Along with these techniques "Hidden Markov Model" is optimize the best solution for the fraud detection.	The main goal of this paper is comparing the data mining techniques, suchas Simple K-means, Hidden Markov Model, Bayesian Network, KNN algorithm and Outlier detection.	Precision Accuracy Minimum Risk

Suraj Patil Varsha Nemade Piyush Kumar Son	2018	Logistic regression Designing analytical model for Fraud prediction Decision Tree Random Forest	In this paper we have proposed a robust framework to process large volume of data, thefunctionality of framework can be extended to extractreal time data fromdifferent desperatesources.	The main challenge for today's CCFD system is how to improve fraud detection accuracy with growing numberof transactions done by user persecond.	Precision FDR FOR LR Sensitivity Miss Rate
			These analytical models are run on credit card dataset and accuracy of analytical model is evaluated with helpof confusion matrix".		Fallout Specificity F1 Score
ALTYEB ALTAHER TAHASHARAF JAMEEL MALEBARY	2020	Light-GBM algorithm K-fold cross-validation (CV) Gradient-based oneside Sampling (GOSS) Decision tree	The results reveal that the proposed algorithmis superior to other classifiers. The results also highlight the importance and value of adopting an efficient parameter optimization strategy for enhancing the predictive performance of the proposed approach.	To demonstrate the effectivenessof our proposed Light-GBM for detecting fraud in credit card transactions, experiments were performed using two real- world public credit card transaction data sets consisting fraudulent transactions and legitimate ones.	Accuracy Precision Recall F1-score
Niloofar Yousefi Marie Alaghband Ivan Garibay	2019	Machine learning Supervised learningK- means algorithm ANN Decision Tree (DT) Support-vector machine(SVM)	During this survey, we noticed that supervised learning techniqueshave been used more frequently than unsupervised methods. To be more specific, the most commonly used fraud detection techniques are LR,ANN, DT, SVM andNB.	To drive the future research agenda for the community in order to develop more accurate, reliable and scalable models of credit card fraud detection.	Accuracy Precision Stable

Shimin LEI Ke XU YiZhe HUANG Xinye SHA	2020	CNN XGboost based model	This Survey presents an XGboost-based financial system to detect transaction fraud.	To introduce a new set of features based on analyzing the periodic behavior of the time of a transaction usingthe von Mises distribution.	Accuracy Auc-Roc score True positive rate (TPR) False positive rate (FPR)
Mehak Mahajan Sandeep Sharma	2019	Supervised learning SVM Logistic Regression Naive Bayes Neural NetworkK- NN Decision tree	In this paper wehave discussed various techniquesof data mining through which we can detect the fraud. Various techniques like Hidden Markov Model, K-mean clustering algorithm, K-nearest neighbor, Decision Tree, Fusion approach due using dumpsterShafer, Bayesian Network, Neural Network, SVM and Logistic Regression are used.	In this paper we have research about the various detecting techniques to identify and detect the fraud through varied techniques of data mining.	Accuracy Efficiency Precision
Ali Yeşilkanat Barış Bayram Bilge Köroğlu Seçil Arslan	2020	Gradient Boosting Tree (GBT) XGBoost	In this work, for the real-time detection of credit card fraud,a new approach is proposed for training dataset construction to make usable and valuable of different types of attributes of a	In this survey, new strategy for training dataset generation employing the sliding window approach in a given time frameto adapt to the changes on the trends of	False-Positive Rate (FPR) Recall Precision Area Under Curve (AUC)
			transaction by combining numerical, hand- crafted numerical,categorical and textual features.	fraudulent transactions.	

Alejandro Correa Bahnsen Djamila Aouada Aleksandar Stojanovic Björn Ottersten	2016	Neural networks Bayesian learning Association rules Hybrid models Support vector machines Peer group analysis Random Forest	In this paper, we address the cost-sensitivity and the features pre- processing to achieve improved fraud detection and savings. In this paper, we proposed a new cost-based measureto evaluate credit card fraud detection models, taking into account the different financial costs incurred by the fraud detection process.	In this paper we expand the transaction aggregation strategy, and propose to create a new set of features based on analyzing the periodic behavior of the time of a transaction usingthe von Mises distribution.	Accuracy Recall Precision F1-Score
Yaodong Han Shun Yao Tie Wen Zhenyu Tian Changyu WangZheyuan Gu	2020	Logistic Regression Decision Tree Random Forest Naïve Bayes Boosted Tree AdaBoost Neural Network Support vector machine (SVM)	In this article, we build machine learning models on a synthetically generated dataset about credit card applications and evaluated their performance. As for building more models, we can also try ensemble model, such as voting or stacking classifiers, to further improve performance.	Traditional approaches, such as expert system, suffers from the incapability to handle complex problems and tremendous amount of data,while the recentdevelopment ofvarious machine learning techniques brings new solutions.	Accuracy Precision
		K-Nearest Neighbor			
Ronish Shakya	2018	Bagging Boosting Logistic regression Random Forest XGBoost	We implemented the algorithmic approaches such asbagging and boosting to tackle the class imbalance problem. Besides these models, we chose logistic regression model to compare with other models. Then, we analyzed all threemodels with and without using resampling techniques.	In this survey to tackle this credit card fraud problem, data- level approach, where different resampling methods such as under-sampling, oversampling, and hybrid strategies, have been implemented along with an algorithmic approach where ensemble models such as bagging and boosting have been applied to a highly skewed dataset containing 284807 transactions.	True Positive (TP) False Positive (FP) True Negative (TN) False Negative (FN)

ARTICLES / STUDIES BASED ON UNSUPERVISED LEARNING ALGORITHMS

Authors	Year	Methodology and Techniques used	Advantages	Issues	Metrics Used
Vaishnavi Nath Dornadula Geetha S	2019	Clustering method SMOTE (Synthetic Minority Over- Sampling Technique)operation.	Lead the system to adapt to new cardholder's transaction behaviours timely	To design and develop a novel fraud detection method for Streaming Transaction Data, with an objective, to analyse the past transaction details of the customers and extract the behavioural patterns.	Accuracy Precision Mathews Correlational Coeff. (MCC)
Massimiliano Zanin Miguel Romance Santiago Moral Regino Criado	2018	Parenclitic networks	Increased efficiencyin detecting frauds in some niches of operations, like medium-sized and on-line transactions.	To detect illegal instances in a real card transaction dataset.	True Positive Rate (TPR) Receiver Operating Characteristic (ROC) False Positive Ratio (FTR)
Dr . Yvan Lucas Dr . Johannes Jurgovsky	2020	**SMOTE** (Synthetic Minority Over- Sampling Technique)operation. Feature engineering techniques Recurrent Neural Networks **(LSTM) – RNN**	Making a strong difference for fraudulent transactions , whichare much more rarethan genuine transactions. Allowing Fraud detection systemsto obtain good performances	To identify fraudulent transactions that have been issue illegitimately on behalf of the rightful card owner.	Precision True Positive Rate (TPR) Accuracy F1 Score Mathews Correlational Coeff. **(MCC)**

		Graphical Models (hidden markov models)			
S P Maniraj Aditya Saini Shadab Ahmed Swarna Deep Sarkar	2019	Machine Learning Algorithm Local Outlier Factor Isolation Forest Algorithm	The algorithm used reaches approximately 100% accuracy , with Precision risingby 5% if dataset size is increased by a factor of 10.	To detect 100% of the fraudulent Transactions. To recognize whether a new transaction is fraudulent or not.	Accuracy Precision Recall F1 Score Support Macro average Weighted Average
Dahee Choi Kyunghoo Lee	2018	Machine Learning Deep Learning Feature Selection Sampling Supervised Algorithms Unsupervised Algorithms HMM -- Hidden Markov Model	ML based method has higherdetection than neural. Networks at various ratios Neural Networks gets accuracy as high as 95%.	Proposing a process for accurate fraud detection -- the overall process of detecting financial fraud based on ML andcomparing it with ANN approach to detect fraud and process large amount of financial data.	Accuracy F- measure
Samaneh Sorournejad Zahra Zojaji Reza Ebrahimi Atani Amir Hassan Monadjemi	2016	Anomaly Detection ANN - Artificial Neural Network Supervised Techniques Unsupervised Techniques	Efficient Accuracy and Maintainability , Improved Capability of Pattern Recognition Improved Working with Noisy Data.	To review the state of the art in credit card fraud detection techniques, datasets and evaluation criteria.	Accuracy Precision / Hit Rate True Positive Rate / Sensitivity False Positive Rate ROC

		HMM -- Hidden Markov Model Bayesian Network Fuzzy Neural Network			Cost F1 measure
Yashvi Jain NamrataTiwari ShripriyaDubey Sarika Jain	2019	Artificial Neural Networks (ANN) Bayesian Network K- Nearest Neighbour (KNN) Hidden Markov Model Fuzzy Logic Based System	Creation of hybrid of various techniques that arealready used in fraud detection to cancel out them limitations and get enhanced performance.	To introduce the concept of fraud related to credit card(s) and them various types and their solutions - valid against modern tricks and Hacks.	Accuracy Detection Rate False Alarm Rate Sensitivity Specificity Cost
Sonal Mehndiratta Kamal Gupta	2019	Data Mining Machine Learning Anomaly Detection ANN - Artificial Neural Network HMM -- Hidden Markov Model K- Nearest Neighbour (KNN)	□ Majority of voting methods achieve good accuracy rates in order to detect thefraud in the credit cards. □ Proposed random forests provide good results on the small dataset. □ There is a large moving window, higher number of attributes and number of link types available which can be searched by CD andSD algorithms	To avoid the leakage of information to the attacker , which if fails willlead to huge amounts of loss to the Credit card company.	Precision Sensitivity Accuracy Balanced Classification Rate

Kaithekuzhical Leena Kurien Dr . Ajeet Chikkamannur	2019	Machine Learning Artificial Intelligence (AI) Deep Learning Single linked hierarchical clustering SMOTE Random Under sampling (RUS)	Efficient Over Sampling , Under-sampling of data for accuracy. Thorough Analysis of the features and sub-sample ratios for imbalanced Datasets.	Probability of fraudulent transactions in prevalence and context of credit card usage.	Feature Co-Relation Quartile True positive(TP) True Negative(TN) False positive(FP) False negative(FN) Precision F1 Score Recall
Apapan Pumsirirat Liu Yan	2018	Creating an Auto Encoder using Deep Learning Restricted Boltzmann Machine TensorFlow (google library)	Accurately achieve credit carddetection with a large dataset. Guarantee thatAE and RBM can make more accurate AUC for receiver operator characteristics.	To focus on fraud cases that cannot be detected basedon previous history or supervised learning. To create a model of deep Auto-encoder and Restricted Boltzmann Machine (RBM) that can reconstruct normal transactions to find anomalies from normal patterns.	MSE RMSE
Daniyal Baig	2020	Artificial Neural Network	Gives the highest accuracy ascompared to other logistic regressions.	To identify the Credit card fraudand provide a reasonable	Accuracy Precision

		Bayesian Network K- Nearest Neighbour (KNN)	By using KNN, it acts as a better classifier for credit card detection.	Solution to the fraud.	Specificity Sensitivity
S. Abinayaa H. Sangeetha R.A. Karthikeyan K. Saran Sriram D. Piyush	2020	Machine Learning	By using machine Learning alongside Random Forest Algorithm we can get a better result in the detection of fraud.	To find the most common methods of fraud alongside their detection methods and algorithms.	Sensitivity Precision Accuracy
Wael Khalifa Mohamed Ismail Roushdy Abdel-Badeeh M. Salem Hossam Eldin Hossam Eldin Mohammed Abd El-Hamid	2019	Machine Learning Data Mining K-Nearest Neighbor Algorithm Bayesian Network	By this research survey, we can figure out which technique is the best to detect credit card fraud and which technique can be inour system.	The goal of this research is to survey procedure of various discoverytechniques dependent on Visa.	Accuracy Speed Precision Exactness Cost
Surbhi Gupta Nitima Malsa Vimal Gupta	2017	Artificial Immune System (AIS) Hidden Markov Model (HMM) Neural Network	By this survey,you can find the differences between each technique mentioned. New classifierscan be formed by combining the different techniques.	The goal is to compare different techniques by either testing them separately or by combining and forming new classifiers which can be used to improve the detection of credit card fraud.	Speed of detection Accuracy Cost
Shiv Shankar Singh	2019	Data Mining	This survey makes sure that more businesses are	This presented paper focuses on fraud activities	Precision Cost
			aware of the different techniques available to catch credit card fraud.	that cannot be detected manually by carrying out research and examine the results of logisticregression, decision tree and support vector machine	Exact outcome Rapidly train machines

Ishu Trivedi Monika Mrigya Mridushi	2016	Artificial Neural network Neural Network Hidden Markov Method	This method proves accurate infinding out the fraudulent transactions and minimizing the number of false alerts. probability of detecting fraud in avery short span of time after the transactions has been made.	The main aim isto detect the fraudulent transaction andto develop a method of generating test data.	Time Speed Accuracy Rate of Error
Yong Fang Yunyun Zhang Cheng Huang	2019	Light-GBM model SMOTE algorithm Imbalanced data	The result indicatesthat the new modelis fast, has a lesser error rate and is more accurate.	The goal is to compare the older system with the new system and see their differences	Accuracy Efficiency Recall Rate Speed
Aman Gulat Prakash Dubey Md Fuzail C Jasmine Norman	2017	Behaviour and Locational Analysis (Neural Logic) Neural Network Geolocation	Has an accuracyof up to 80% in detecting credit card fraud. The geolocation of the fraudster canbe pinpointed	The goal here is to create a Credit card frauddetection systemusing the techniques of neural networks and geolocation.	Accuracy Rate of Detection
Mangayarkarasi R					
N. Geetha T. Kavipriya	2017	Hidden Markov Model Neural Network Bayesian Network K- nearest neighbour algorithm Fuzzy Logic Based System	Our goal is to analyze different data mining techniques in a waythat they help us to detect and predict the credit card fraud. Analysis presented by different researcher's shows that different data mining techniques. Along with these techniques "Hidden Markov Model" is optimize the best solution for the fraud detection.	The main goal of this paper is comparing the data mining techniques, suchas Simple K-means, Hidden Markov Model, Bayesian Network, KNN algorithm and Outlier detection.	Precision Accuracy Minimum Risk

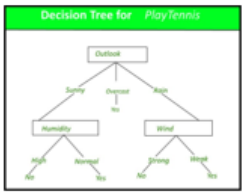
Suraj Patil Varsha Nemade Piyush Kumar Son	2018	Designing analytical model for Fraud prediction	In this paper we have proposed a robust framework to process large volume of data, thefunctionality of framework can be extended to extractreal time data fromdifferent desperatesources. These analytical models are run on credit card dataset and accuracy of analytical model is evaluated with helpof confusion matrix“.	The main challenge for today’s CCFD system is how to improve fraud detection accuracy with growing numberof transactions done by user persecond.	Precision FDR FOR LR Sensitivity Miss Rate Fallout Specificity F1 Score
ALTYEB ALTAHER TAHA SHARAF JAMEEL MALEBARY	2020	Light-GBM algorithm K-fold cross- validation (CV) Gradient-based one side Sampling (GOSS)	The results reveal that the proposed algorithm is superior to other classifiers. The results also highlight the importance and value of adopting an efficient parameter optimization strategy for enhancing the predictive performance of the proposed approach.	To demonstrate the effectiveness of our proposed Light-GBM for detecting fraud in credit card transactions, experiments were performed using two real- world public credit card transaction data sets consisting fraudulent transactions and legitimate ones.	Accuracy Precision Recall F1-score
Niloofar Yousefi Marie Alaghband Ivan Garibay	2019	Machine learning Supervised learning K-means algorithm ANN	During this survey, we noticed that supervised learning techniques have been used more frequently than unsupervised methods. To be more specific, the most commonly used fraud detection techniques are LR, ANN, DT, SVM and NB.	To drive the future research agenda for the community in order to develop more accurate, reliable and scalable models of credit card fraud detection.	Accuracy Precision Stable

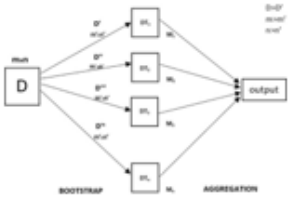
Shimin LEI Ke XU YiZhe HUANG Xinye SHA	2020	Hidden Markov Model CNN	This Survey presents an Xgboost-based financial system to detect transaction fraud.	To introduce a new set of features based on analyzing the periodic behavior of the time of a transaction using the von Mises distribution.	Accuracy Auc-Roc score True positive rate (TPR) False positive rate (FPR)
Mehak Mahajan Sandeep Sharma	2019	Neural Network K-NN	In this paper we have discussed various techniques of data mining through which we can detect the fraud. Various techniques like Hidden Markov Model, K-mean clustering algorithm, K-nearest neighbor, Decision Tree, Fusion approach due using dumpster Shafer, Bayesian Network, Neural Network, SVM and Logistic Regression are used.	In this paper we have research about the various detecting techniques to identify and detect the fraud through varied techniques of data mining.	Accuracy Efficiency Precision
Ali Yeşilkanat Barış Bayram Bilge Köroğlu Seçil Arslan	2020	Gradient Boosting Tree (GBT)	In this work, for the real-time detection of credit card fraud, a new approach is proposed for training dataset construction to make usable and valuable of different types of attributes of a transaction by combining numerical, hand- crafted numerical, categorical and textual features.	In this survey, new strategy for training dataset generation employing the sliding window approach in a given time frame to adapt to the changes on the trends of fraudulent transactions.	False- Positive Rate (FPR) Recall Precision Area Under Curve (AUC)

Alejandro Correa Bahnsen	2016	Neural networks	In this paper, we address the cost-sensitivity and the	In this paper we expand the transaction	Accuracy Recall
Djamila Aouada Aleksandar Stojanovic Björn Ottersten		Bayesian learning Association rules Hybrid models Peer group analysis	features pre- processing to achieve improved fraud detection and savings. In this paper, we proposed a new cost-based measure to evaluate credit card fraud detection models, taking into account the different financial costs incurred by the fraud detection process.	aggregation strategy, and propose to create a new set of features based on analyzing the periodic behavior of the time of a transaction using the von Mises distribution.	Precision F1-Score
Yaodong Han Shun Yao Tie Wen Zhenyu Tian Changyu Wang Zheyuan Gu	2020	Boosted Tree AdaBoost Neural Network K-Nearest Neighbor	In this article, we build machine learning models on a synthetically generated dataset about credit card applications and evaluated their performance. As for building more models, we can also try ensemble model, such as voting or stacking classifiers, to further improve performance.	Traditional approaches, such as expert system, suffers from the incapability to handle complex problems and tremendous amount of data, while the recent development of various machine learning techniques brings new solutions.	Accuracy Precision
Ronish Shakya	2018	Synthetic Minority Over-sampling Technique (SMOTE) Bagging Boosting	We implemented the algorithmic approaches such as bagging and boosting to tackle the class imbalance problem.	In this survey to tackle this credit card fraud problem, data- level approach, where different resampling methods such as under-sampling,	True Positive (TP) False Positive (FP) True Negative (TN) False Negative (FN)

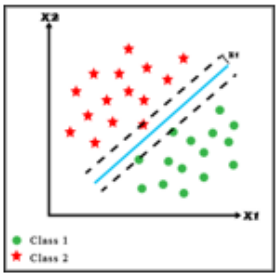
			□ Besides these models, we chose logistic regression model to compare with other models. □ Then, we analyzed all threemodels with and without using resampling techniques.	oversampling, and hybrid strategies, have been implemented along with an algorithmic approach where ensemble models such as bagging and boosting have been applied to a highly skewed dataset containing 284807 transactions.	

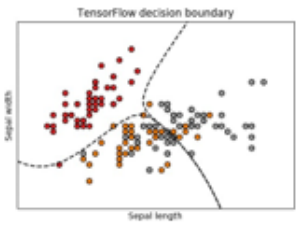
ARTICLES BASED ON SOME OF THE MENTIONED TECHNIQUES

S. no	Category	Algorithm / Technique used	Definition (with / without pictures)	Advantages	Disadvantages	Citation(s)
1	Machine Learning	Decision Tree Algorithm	It's a **Supervised Learning Algorithm**. A Decision tree is a flowchart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminalnode) holds a class label. It's mostly is used here to sort the information received and sorts and classifies the data into different groups.  	**1.** Compared to other algorithms, decision tree's less effortive for data preparation during pe-processing. **2.** A decision tree does not require normalization of data. **3.** A decision tree does not require scaling of data as well. **4.** Missing values in the data do not affect the process of building a decision tree to any extent. **5.** A decision tree model is very intuitive and easy to explain to technical teams as well as stakeholders.	**1.** A small change in the data can cause a large change in the structure of the decision tree, causing instability. **2.** For a decision tree, sometimes, calculation can go for mare complex, compared to other algorithms. **3.** Decision tree often involves higher time to train the model. **4.** Decision tree training is relatively expensive as the complexity and time taken are more. **5.** The decision tree algorithm is inadequate for applying regression and predicting continuous values.	**[4]** Navanshu Khare , Saad Yunus Sait (2018) , Credit Card Fraud Detection Using Machine Learning Models and Collating Machine Learning Models. **[8]** Yashvi Jain , NamrataTiwari , Shripriya Dubey , Sarika Jain (2019) , A Comparative Analysis of Various Credit Card Fraud Detection Techniques. **[16]** Surbhi Gupta , Mrs. Nitima Malsa , Mr. Vimal Gupta (2017) , Credit Card Fraud Detection & Prevention – A Survey. **[21]** N.Geetha , T.Kavipriya (2017) , Study on Credit Card Fraud Detection Using Data Mining Techniques

2	Machine Learning	Random Forest Algorithm	It's a Supervised Learning Algorithm.A Random Forest is an ensemble technique capable of performing both regression and classification tasks with the useof multiple decision trees and a technique called	**1.** It reduces overfitting in decision trees and helps to improve the accuracy. **2.** It is flexible to both classification and regression problems.	**1.** It requires much computational power as well as resources as it builds numerous trees to combine their outputs.	**[4]** Navanshu Khare , Saad Yunus Sait (2018) , Credit Card Fraud Detection Using Machine Learning Models and Collating Machine Learning Models
			Bootstrap and Aggregation, commonly known as **bagging**. The basic idea behind this is to combine multiple decision trees in determiningthe final output rather than relying on individual decision trees.The random forest algorithm takes the sorted data from the decision tree algorithm and finds out how accurate the data matches with the dataset given by the databank. 	**3.** It works well with both categorical and continuous values. **4.** It automates missing values present in the data. **5.** Normalizing of data is not required as it uses a rule-based approach.	**2.** It also requires much time for training as it combines a lot of decision trees to determine the class. **3.** Due to the ensemble of decision trees, it also suffers interpretability and fails to determine the significance of each variable	**[13]** S. Abinayaa , H. Sangeetha, R. A. Karthikeyan , K. Saran Sriram, D. Piyush (2020) , Credit Card Fraud Detection and Prevention using Machine Learning. **[14]** Nishant Sharma (2019) , CREDIT CARD FRAUD DETECTION PREDICTIVE MODELING. **[19]** Yong Fang , Yunyun , Zhang Cheng Huang (2019) , Credit Card Fraud Detection Based on Machine Learning

3	Machine Learning	K – Nearest Neighbour	K-Nearest Neighbor is one of the most basic yet essential classification algorithms in Machine Learning. It belongs to the **supervised learning** domain and finds intense application in pattern recognition, data mining and intrusion detection. It is widely disposable in real-life scenarios since it is non-parametric, meaning, it does not make any underlying assumptions about the distribution of data (as opposed to other algorithms such as GMM, which assume a Gaussian distribution of the given data) 	**1.** Quick calculation time **2.** Simple algorithm – to interpret. **3.** Versatile – useful for regression and classification. **4.** High accuracy – you do not need to compare with better supervised learning models. **5.** No assumptions about data – no need to make additional assumptions , tune several parameters , or build a model. This makes it crucial in non-linear data case.	**1.** Accuracy depends on the quality of the data. **2.** With large data , the prediction stage might be slow. **3.** Sensitive to the scale of the data and irrelevant features. **4.** Require high memory – need to store all of the training data. **5.** Given that it stores all of the training , it can be computationally expensive.	**[8]** Yashvi Jain , NamrataTiwari , Shripriya Dubey , Sarika Jain (2019) , A Comparative Analysis of Various Credit Card Fraud Detection Techniques. **[9]** Sonal Mehndiratta , Mr. Kamal Gupta (2019) , Credit Card Fraud Detection Techniques: A Review **[26]** Mehak Mahajan , Sandeep Sharma (2019) , Detect Frauds in Credit Card using Data Mining Techniques

4	Machine Learning	Support Vector Machine (SVM)	A **Support Vector** Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labelled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. A support vector machine takes these data points and outputs the hyperplane (which in two dimensions it's simply a line) that best separates the tags. This line is the decision boundary: anything that falls to one side of it we will classify as blue, and anything that falls to the other as red. 	**1.** SVM works relatively well when there is a clear margin of separation between classes. **2.** SVM is more effective in high dimensional spaces. **3.** SVM is effective in cases where the number of dimensions is greater than the number of samples. **4.** SVM is relatively memory efficient.	**1.** SVM algorithm is not suitable for large data sets. **2**. SVM does not perform very well when the data set has more noise , i.e. , target classes are overlapping. **3.** In cases where the number of features for each data point exceeds the number of training data samples , the SVM will underperform. **4.** As the Support Vector Classifier works By putting data points , above and below the classifying hyper-lane there is no probabilistic explanation for the classification.	**[4]** Navanshu Khare , Saad Yunus Sait (2018) , Credit Card Fraud Detection Using Machine Learning Models and Collating Machine Learning Models. **[7]** Samaneh Sorournejad , Zahra Zojaji , Reza Ebrahimi Atani , Amir Hassan Monadjemi (2016) , A Survey of Credit Card Fraud Detection Techniques: Data and Technique Oriented Perspective **[8]** Yashvi Jain , NamrataTiwari , Shripriya Dubey , Sarika Jain (2019) , A Comparative Analysis of Various Credit Card Fraud Detection Techniques

5	Machine Learning	Naïve Bayes Algorithm	It's a Supervised Learning Algorithm. In statistics, naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong independence assumptions between the features. They are among the simplest Bayesian network models, but coupled with kernel density estimation, they can achieve higher accuracy levels.	**1.** This algorithm works quickly and can save a lot of time. **2.** Naïve Bayes is suitable for solving multi-class prediction problems. **3.** If its assumption of the independence of features holds true , it can perform than other models and requires much less training data. **4.** Naïve Bayes is better suited for categorical input	**1.** Naïve Bayes assumes that all predictors (OR features) are independent , rarely happening in real life. This limits the applicability of this algorithm in real-world use cases. **2.** This algorithm faces the 'ZERO-frequency problem' where it assigns zero probability to a categorical variable whose category in the	**[26]** Mehak Mahajan , Sandeep Sharma (2019) , Detect Frauds in Credit Card using Data Mining Techniques **[29]** Yaodong Han , Shun YaoTie Wen , Zhenyu Tian , Changyu Wang , Zheyuan Gu (2020) , Detection and Analysis of Credit Card Application Fraud Using Machine Learning Algorithms.
				variables than numerical variables.	test data wasn't available in the training dataset. It would be best if you use a smoothing technique to overcome this issue. **3.** Its estimates can be wrong in some cases , so you shouldn't take its probability outputs very seriously.	**[9]** Sonal Mehndiratta , Mr. Kamal Gupta (2019) , Credit Card Fraud Detection Techniques: A Review

IV. PROPOSED ALGORITHM

After search numerous articles / studies / projects , analysing the various architectures proposed / created in huge numbers , we thus, present our own proposed architecture using numerous modules.

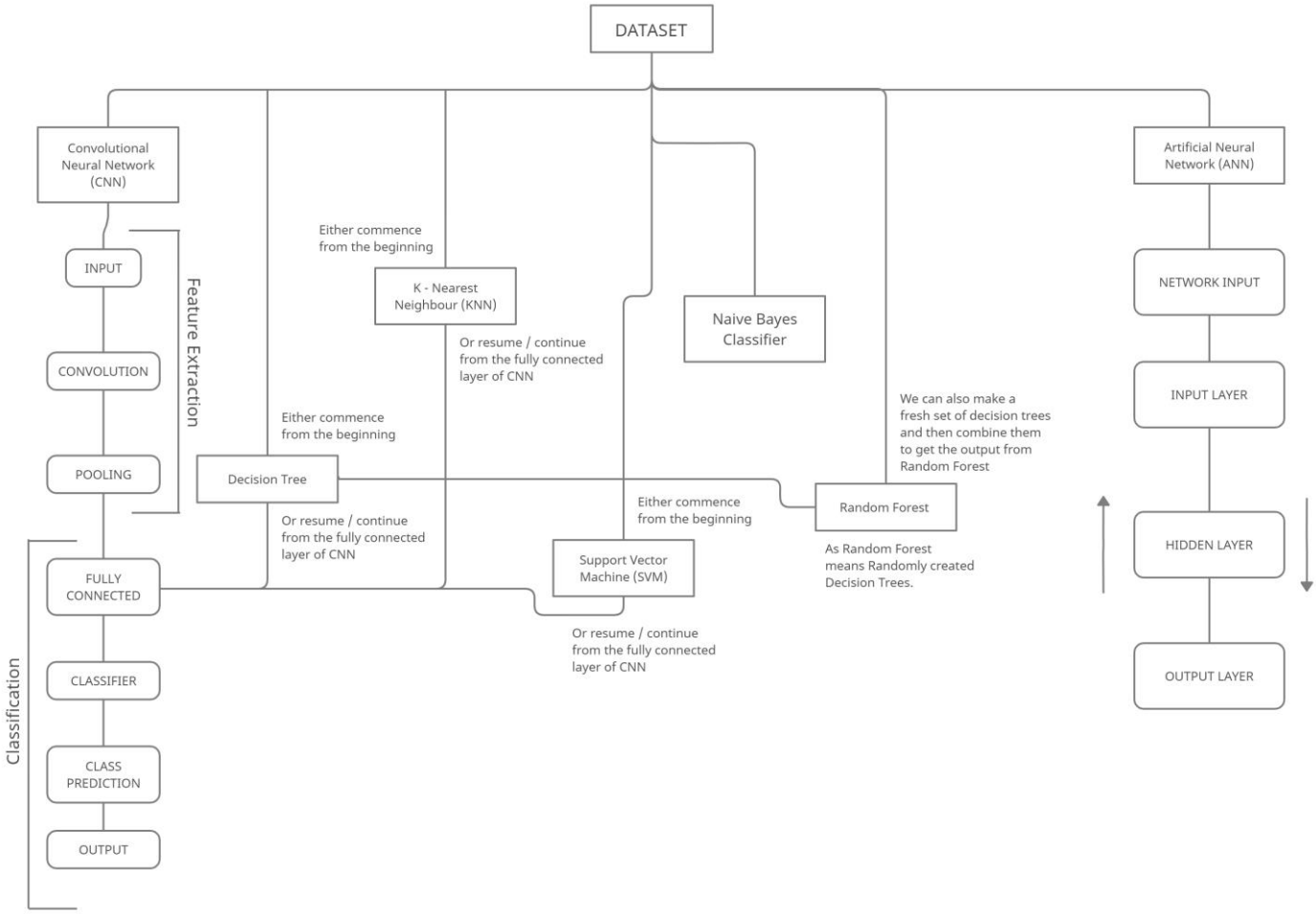


Fig. : Proposed Architecture for modules mentioned

As we can see , we have included 7 different modules in our architecture. We shall now discuss each one of those with their own algorithms and the reason for expanding their own flow with other module.

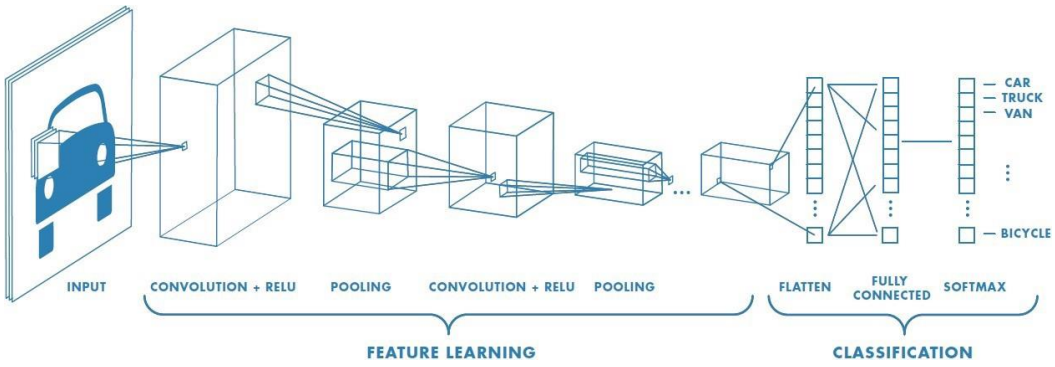
1. CONVOLUTIONAL NEURAL NETWORK :

Being a Deep Learning Algorithm , it can take input image, assign importance to various aspects / objects in the image and be able to differentiate one from the other. The architecture of a CNN (also called ConVet) , is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex (The visual cortex of the brain is the area of the cerebral cortex that processes visual information).

We propose to get familiar with a Decision tree, which explains the particular justification every forecast made by the CNN at the semantic level. i.e., the Decision tree expands feature representations in high conv-layers of the CNN into Primary Concepts of item parts. Thusly, the Decision Tree tells individuals which article parts enact which channels for the forecast and the amount they add to the metrics' score.

The CNN-KNN model uses advantages of both methods (CNN and KNN). The advantages of CNN are sparse connectivity among the neurons between successive layers and weights sharing between layers. The KNN classifies the nearest data samples as a class based on similar measures. This CNN - KNN model extracted the salient features automatically and reduce the laborious and time consumption.

Studies shows that using a CNN - SVM model increases the accuracy of the obtained results , rather than interpreting from simple / traditional CNN models. Thus , SVM is used along with CNN to increase Accuracy.

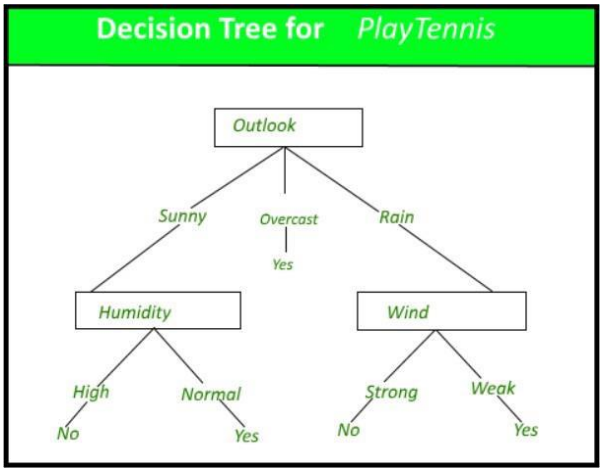


Algorithm / Pseudocode :

- Flatten the input dimensions to 1D (width pixels X height pixels)
- Normalize the image pixel values (divide by 255)
- One-hot encode the categorical column
- Build a model architecture (sequential) with dense layers
- Train the model and make predictions

2. DECISION TREE ALGORITHM :

It's a Supervised Learning Algorithm. A Decision tree is a flowchart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label. It's mostly is used here to sort the information received and sorts and classifies the data into different groups.



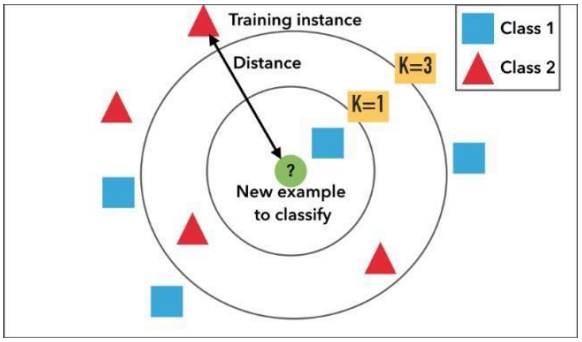
Algorithm / Pseudocode :

- Place the best attribute of the dataset at the root of the tree.
- Split the training set into subsets. Subsets should be made in such a way that each subset contains data with the same value for an attribute.
- Repeat step 1 and step 2 on each subset until you find leaf nodes in all the branches of the tree.

3. K-NEAREST NEIGHBOUR :

K-Nearest Neighbor is one of the most basic yet essential classification algorithms in Machine Learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining and intrusion detection.

It is widely disposable in real-life scenarios since it is non- parametric, meaning, it does not make any underlying assumptions about the distribution of data (as opposed to other algorithms such as GMM, which assume a Gaussian distribution of the given data).



Algorithm / Pseudocode :

Let (X_i, C_i) where i = 1, 2……., n be data points. X_i denotes feature values & C_i denotes labels for X_i for each i.

Assuming the number of classes as 'c'

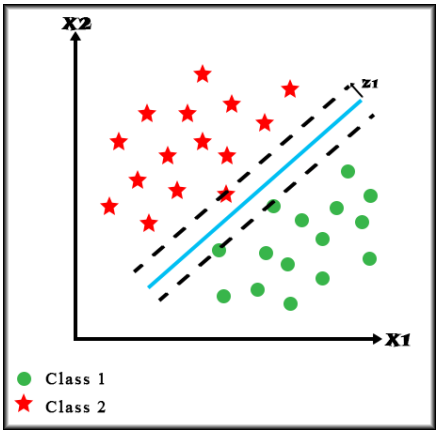
$C_i \in \{1, 2, 3, ……, c\}$ for all values of i

- Calculate "$d(x, x_i)$" i =1, 2, ….., n; where d denotes the Euclidean distance between the points.
- Arrange the calculated n Euclidean distances in non-decreasing order.
- Let k be a +ve integer, take the first k distances from this sorted list.
- Find those k-points corresponding to these k-distances.
- Let k_i denotes the number of points belonging to the i^{th} class among k points i.e., $k \geq 0$
- If $k_i > k_j \; \forall \; i \neq j$ then put x in class i.

4. SUPPORT VECTOR MECHANISM :

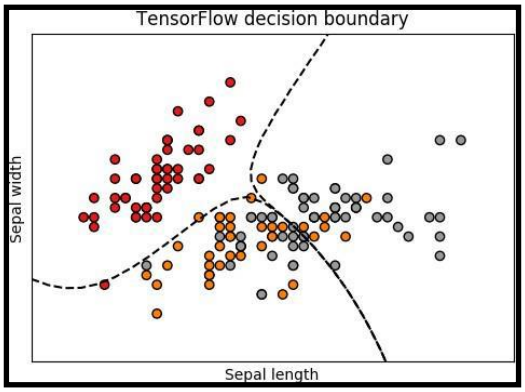
Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labelled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples.

A support vector machine takes these data points and outputs the hyperplane (which in two dimensions it's simply a line) that best separates the tags. This line is the decision boundary: anything that falls to one side of it we will classify as blue, and anything that falls to the other as red.



5. NAÏVE BAYES CLASSIFIER :

It's a Supervised Learning Algorithm. In statistics, naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong independence assumptions between the features. They are among the simplest Bayesian network models, but coupled with kernel density estimation, they can achieve higher accuracy levels.



Algorithm / Pseudocode :

Input :
Training dataset T ,
F = (f_1, f_2, f_3, …, f_n) // value of the predictor variable in testing the dataset.

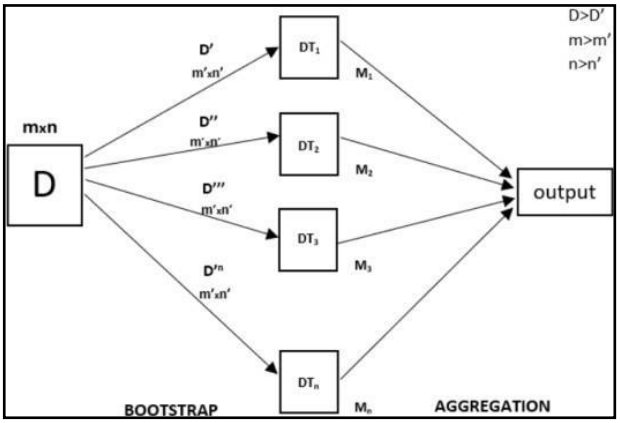
Output :
A class of testing dataset.

Steps Covered :

- Read the training dataset T
- Calculate the mean and standard deviation of the predictor variables in each class
- Repeat
- Calculate the probability of f using the gauss density equation in each class
- Until the probability of all predictor variables (f_1, f_2, f_3, …, f_n) has been calculated
- Calculate the likelihood for each class
- Get the greatest likelihood

6. RANDOM FOREST ALGORITHM :

It's a Supervised Learning Algorithm. A Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called Bootstrap and Aggregation, commonly known as bagging. The basic idea behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision trees. The random forest algorithm takes the sorted data from the decision tree algorithm and finds out how accurately the data matches with the dataset given by the databank.



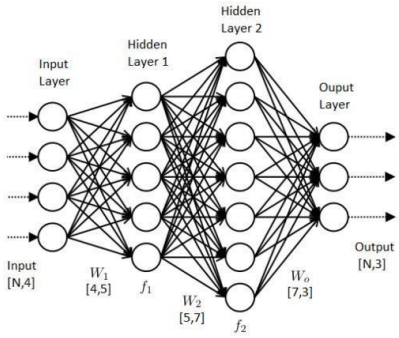
Algorithm / Pseudocode :

- Randomly select "k" features from total "m" features.
- Where k << m
- Among the "k" features, calculate the node "d" using the best split point.
- Split the node into daughter nodes using the best split.
- Repeat 1 to 3 steps until "l" number of nodes has been reached.
- Build forest by repeating steps 1 to 4 for "n" number times to create "n" number of trees.

7. ARTIFICIAL NEURAL NETWORK :

This can be Supervised and Unsupervised both , depending on the intended outcome and the outputs. Artificial neural networks, usually simply called neural networks, are computing systems vaguely inspired by the biological neural networks that constitute animal brains. An ANN is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a

biological brain. The Artificial Neural Network (ANN) is mainly used to increase the accuracy of the random forest algorithm to almost 100%.



The input layer is the first layer of an ANN that receives the input information in the form of various texts, numbers, audio files, image pixels, etc. In the middle of the ANN model are the hidden layers. There can be a single hidden layer, as in the case of a perceptron or multiple hidden layers. These hidden layers perform various types of mathematical computation on the input data and recognize the patterns that are part of. In the output layer, we obtain the result that we obtain through rigorous computations performed by the middle layer.

8. CONFUSION 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. The confusion matrix itself is relatively simple to understand, but the related terminology can be confusing. A confusion Matrix is used to graph and classify the different cases of fraudulent transactions detected in our system

n=165	**Predicted: NO**	**Predicted: YES**	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Algorithm / Pseudocode :

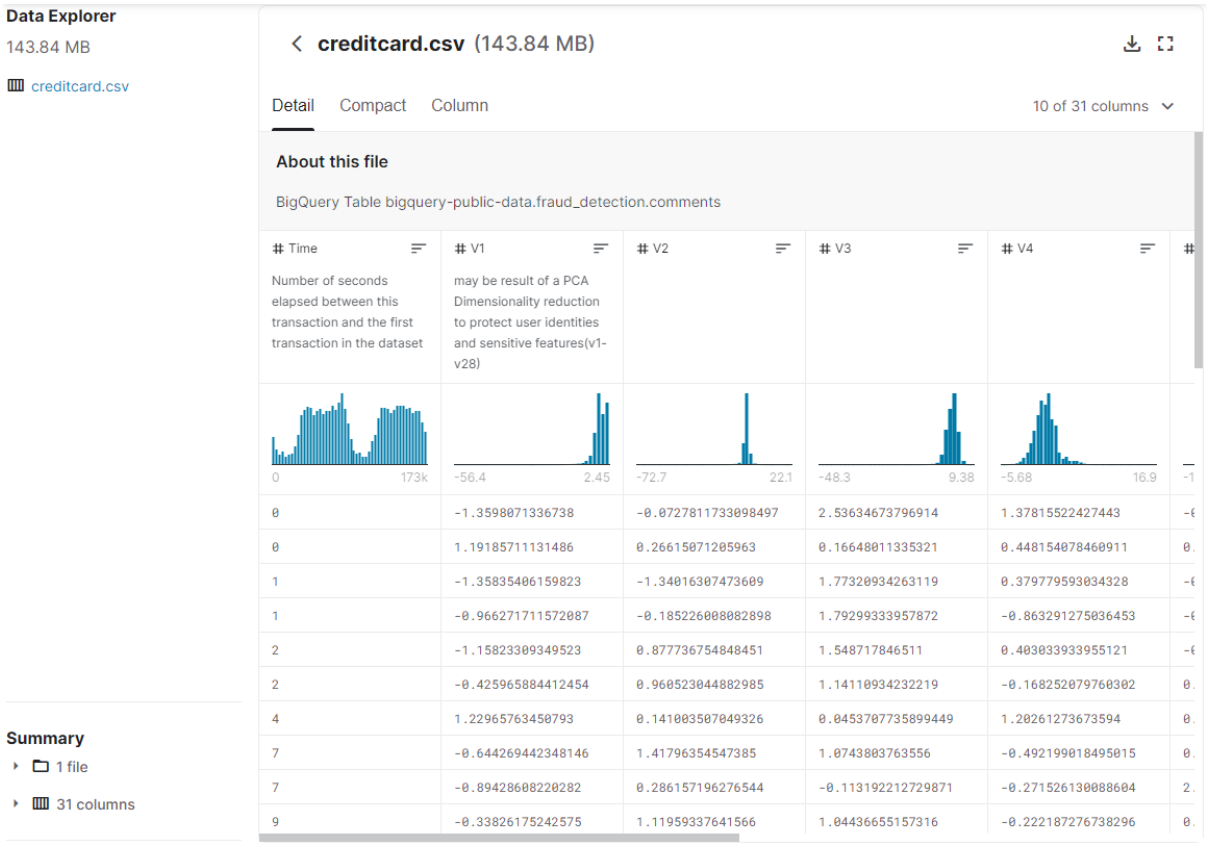
- You need a test dataset or a validation dataset with expected outcome values.
- Make a prediction for each row in your test dataset.
- From the expected outcomes and predictions count -
  The number of correct predictions for each class.
- The number of incorrect predictions for each class, organized by the class that was predicted.

V. EXPERIMENTS - RESULTS

Before providing the snaps / implementation codes – let us discuss about the dataset that we will be using. The dataset used for this very project has been taken from Kaggle – by Google (LINK : https://www.kaggle.com/mlg-ulb/creditcardfraud) . Kaggle is a huge community of Data Scientists who provide datasets and many other resources for analysis and learning purpose.
Coming back to the dataset explanation – The dataset shows us the information of around a hundred – thousand transactions in the European Countries (Altogether). It's given that 0.17% of all the transaction carried out are fraudulent , and the rest are successful / Valid Transactions. That means , out of the given 284,807 transactions , only 492 frauds happened – quite a small number !
All the features and numerical based , i.e. , only numerical values are stored in the dataset (for analysis , numbers make the job easy). The feature 'Amount' is the transaction Amount, this feature can be used for example-dependant cost-sensitive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise. There are total 31 columns in the dataset.

The whole implementation has been carried out in Python 3 environment (Python 3.9.4) in VS Code Python Jupyter Notebook along with the virtual environment by Anaconda.

Now let us have a look to the dataset selected for this project ---



We can see that the columns / attributes have somewhat unrelated values to what we need for this project. Actually , these values are (cell values written under the column names V1 , V2 , …..V28) results of the PCA transformation.
Principal component analysis (PCA) is a technique for dimensionality reduction, which is the process of reducing the number of predictor variables in a dataset. PCA initially hails from the field of straight variable Linear Algebra. It is a change strategy that makes (weighted linear) blends of the first factors in an informational index, with the goal that the new mixes will catch as much fluctuation (i.e., the partition between focuses) in the dataset as could really be expected while killing relationships (i.e., redundancy). PCA makes the new factors by changing the first (mean-centred) perceptions (records) in a dataset to another arrangement of factors (measurements) utilizing the eigenvectors and eigenvalues determined from a covariance grid of the unique factors.

In Linear Algebra, an eigenvector or trademark vector of a straight change is a nonzero vector that changes all things considered by a scalar factor when that direct change is applied to it. The relating eigenvalue, regularly signified by 'Lambda', is the factor by which the eigenvector is scaled.

Now let us jump into the implementation portion. Almost every important piece of code has some information provided in the comments (along with the evaluation metrics) ---

IMPLEMENTATION OF CNN (CONVOLUTIONAL NEURAL NETWORKS) –

```
#SOFT COMPUTING - J COMPONENT PROJECT

'''
TEAM ---

SOUBHIK SINHA (19BIT0303)
ROHIT K (19BIT0318)
ADARSH KUMAR SINGH (19BIT0253)
'''

#TOPIC : CREDIT CARD FRAUD DETECTION
'''
COMPONENTS / MODULES --
CONVOLUTIONAL NEURAL NETWORK (CNN)
'''
```

```
'\nCOMPONENTS / MODULES -- \nCONVOLUTIONAL NEURAL NETWORK\n'
```

```
#Importing packages

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Flatten, Dense, Dropout, BatchNormalization
from tensorflow.keras.layers import Conv1D, MaxPool1D
from tensorflow.keras.optimizers import Adam
print(tf.__version__) #Version is important because many libraries get updatded very
                      #frequently , resulting to the depreciation of some commands.
```

```
2.5.0
```

```
import pandas as ps
import numpy as ny
import  seaborn as sbn
import matplotlib.pyplot as pylt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

```
#Lets import / load the dataset

cred = ps.read_csv("D:\COLLEGE_4th_SEM\E2_Soft_Computing_(J)\creditcard.csv")
cred.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431

5 rows × 31 columns

```
cred.shape
```

```
(284807, 31)
```

```
#Checking NULL values

cred.isnull().sum()
```

```
Time      0
V1        0
V2        0
V3        0
V4        0
V5        0
V6        0
V7        0
V8        0
V9        0
V10       0
V11       0
V12       0
V13       0
V14       0
V15       0
V16       0
V17       0
V18       0
V19       0
V20       0
V21       0
V22       0
V23       0
V24       0
V25       0
V26       0
V27       0
V28       0
Amount    0
Class     0
dtype: int64
```

```
cred.info()
```

```
cred.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
```

```
cred['Class'].value_counts()
```

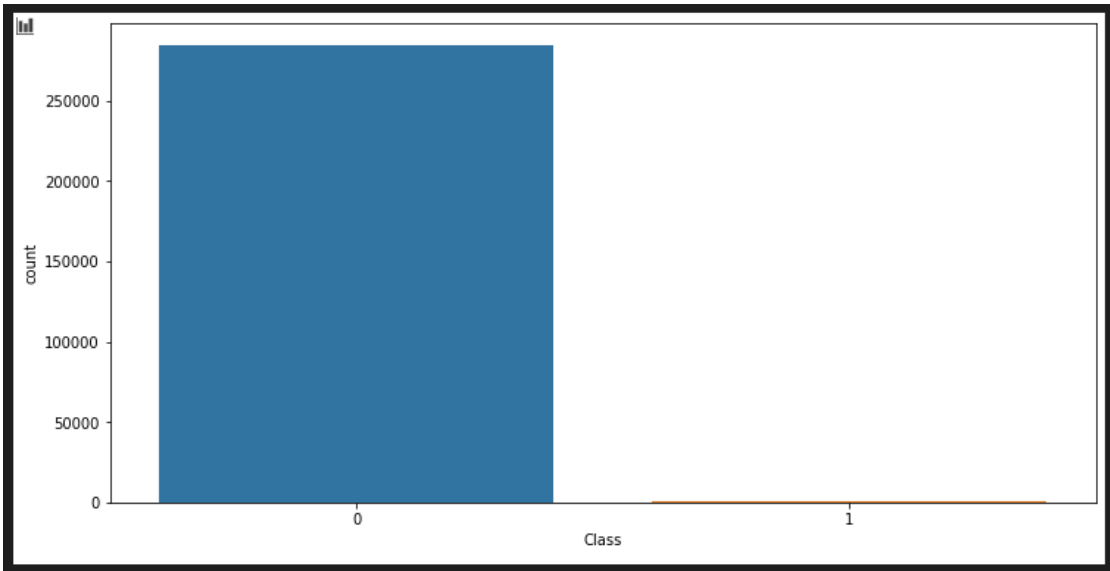
```
0    284315
1       492
Name: Class, dtype: int64
```

```
'''
0 means valid transaction --> 284315 in numbers
1 means fraudulent transaction --> 492 in numbers
'''
```

```
'\n0 means valid transaction --> 284315 in numbers\n1 means fraudulent transaction --> 492 in numbers\n'
```

```
import warnings as wrng
wrng.filterwarnings("ignore")
pylt.figure(figsize=(12,6))
sbn.countplot(cred["Class"])
```

```
<AxesSubplot:xlabel='Class', ylabel='count'>
```




```
#if you see closely , there is some graph / block above the feature 1 - constituting the
#no. of fraud cases (492 out of 284807).
```

```
#Dataset balancing

valid = cred[cred['Class']==0]
fraud = cred[cred['Class']==1]
```

```
valid.shape, fraud.shape
```

```
((284315, 31), (492, 31))
```

```
valid = valid.sample(fraud.shape[0])
valid.shape
```

```
(492, 31)
```

```
cred = fraud.append(valid, ignore_index=True)
cred
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	
0	406.0	-2.312227	1.951992	-1.609851	3.997906	-0.522188	-1.426545	-2.537387	1.391657	-2.770089	...	0.5
1	472.0	-3.043541	-3.157307	1.088463	2.288644	1.359805	-1.064823	0.325574	-0.067794	-0.270953	...	0.6
2	4462.0	-2.303350	1.759247	-0.359745	2.330243	-0.821628	-0.075788	0.562320	-0.399147	-0.238253	...	-0.2
3	6986.0	-4.397974	1.358367	-2.592844	2.679787	-1.128131	-1.706536	-3.496197	-0.248778	-0.247768	...	0.5
4	7519.0	1.234235	3.019740	-4.304597	4.732795	3.624201	-1.357746	1.713445	-0.496358	-1.282858	...	-0.3
...	...	...	...	...	...	...	...	...	...	...	...	
979	51849.0	1.241534	-0.652551	0.618187	-0.748399	-1.037570	-0.359124	-0.701672	0.085478	-1.004227	...	0.0
980	87209.0	-0.089293	0.532420	0.124387	-0.311731	0.343017	-1.115875	1.082966	-0.242925	0.021415	...	0.4
981	134811.0	-2.415369	1.742191	0.578007	0.408996	-0.995119	1.858211	-0.445551	0.743951	0.661218	...	-0.0
982	154918.0	0.080918	0.956493	-0.334867	-0.632220	0.935900	-0.516355	0.827996	0.069705	-0.278242	...	-0.2
983	114361.0	-0.476196	1.101944	-1.324560	-0.867280	1.912673	-1.135177	1.458408	-1.260725	-0.650718	...	0.9

984 rows × 31 columns

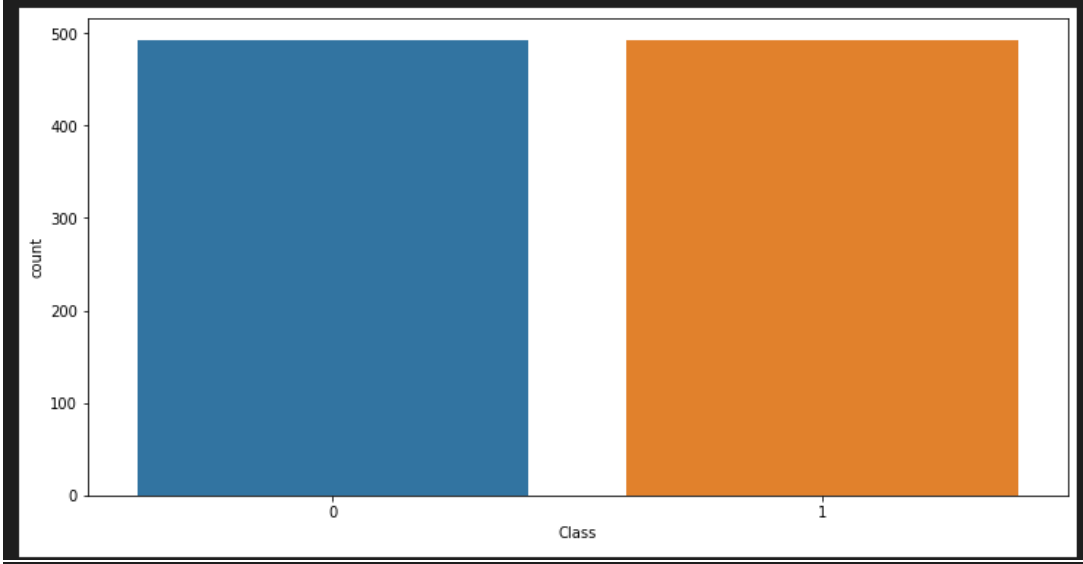
```
cred['Class'].value_counts()
```

```
0    492
1    492
Name: Class, dtype: int64
```

```
X = cred.drop('Class', axis = 1)
Y = cred['Class']
```

```
pylt.figure(figsize=(12,6))
sbn.countplot(Y)
```

```
<AxesSubplot:xlabel='Class', ylabel='count'>
```




```
#Now it seems to be a balnaced dataset.
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2,
                                                    random_state = 0, stratify = Y)
```

```
X_train.shape, X_test.shape
```

```
((787, 30), (197, 30))
```

```
#Thus , we have splitted the balaved data into test set and train set of data for model.
```

```
#Using Standard Scaler to Scale the data

slr = StandardScaler()
X_train = slr.fit_transform(X_train)
X_test = slr.transform(X_test)
```

```
Y_train = ny.array(Y_train)
Y_test=ny.array(Y_test)
```

```
X_train.shape
```

```
(787, 30)
```

```
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], 1)
```

```
X_train.shape, X_test.shape
```

```
((787, 30, 1), (197, 30, 1))
```

```
#Lets build the model of CNN

epochs = 20
cnn_model = Sequential()
cnn_model.add(Conv1D(32, 2, activation='relu', input_shape = X_train[0].shape))
cnn_model.add(BatchNormalization())
cnn_model.add(Dropout(0.2))

cnn_model.add(Conv1D(64, 2, activation='relu'))
cnn_model.add(BatchNormalization())
cnn_model.add(Dropout(0.5))

cnn_model.add(Flatten())
cnn_model.add(Dense(64, activation='relu'))
cnn_model.add(Dropout(0.5))

cnn_model.add(Dense(1, activation='sigmoid'))
```

```
cnn_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, 29, 32)            96
_________________________________________________________________
batch_normalization (BatchNo (None, 29, 32)            128
_________________________________________________________________
dropout (Dropout)            (None, 29, 32)            0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 28, 64)            4160
_________________________________________________________________
batch_normalization_1 (Batch (None, 28, 64)            256
_________________________________________________________________
dropout_1 (Dropout)          (None, 28, 64)            0
_________________________________________________________________
flatten (Flatten)            (None, 1792)              0
_________________________________________________________________
dense (Dense)                (None, 64)                114752
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 65
_________________________________________________________________
```

```
cnn_model.compile(optimizer=Adam(lr=0.0001), loss = 'binary_crossentropy', metrics=['accuracy']
)
```

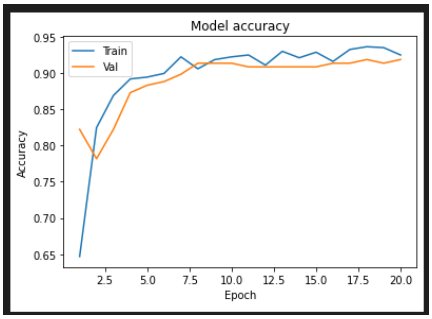
```
history = cnn_model.fit(X_train, Y_train, epochs=epochs, validation_data=(X_test, Y_test),
verbose=1)
```

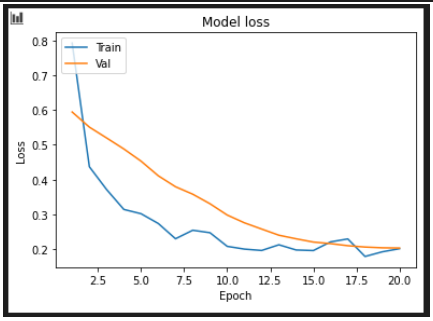
```
Epoch 1/20
25/25 [==============================] - 1s 12ms/step - loss: 0.7920 - accuracy: 0.6468 - val_loss: 0.5
942 - val_accuracy: 0.8223
Epoch 2/20
25/25 [==============================] - 0s 5ms/step - loss: 0.4369 - accuracy: 0.8247 - val_loss: 0.55
07 - val_accuracy: 0.7817
Epoch 3/20
25/25 [==============================] - 0s 5ms/step - loss: 0.3714 - accuracy: 0.8691 - val_loss: 0.51
96 - val_accuracy: 0.8223
Epoch 4/20
25/25 [==============================] - 0s 4ms/step - loss: 0.3143 - accuracy: 0.8920 - val_loss: 0.48
80 - val_accuracy: 0.8731
Epoch 5/20
25/25 [==============================] - 0s 4ms/step - loss: 0.3020 - accuracy: 0.8945 - val_loss: 0.45
30 - val_accuracy: 0.8832
Epoch 6/20
25/25 [==============================] - 0s 5ms/step - loss: 0.2739 - accuracy: 0.8996 - val_loss: 0.41
10 - val_accuracy: 0.8883
Epoch 7/20
25/25 [==============================] - 0s 5ms/step - loss: 0.2301 - accuracy: 0.9225 - val_loss: 0.37
95 - val_accuracy: 0.8985
Epoch 8/20
25/25 [==============================] - 0s 5ms/step - loss: 0.2545 - accuracy: 0.9060 - val_loss: 0.35
81 - val_accuracy: 0.9137
```

```
#Here , we can see that the accuracy is changing after every epoch , that is
#Each time the dataset is passed (for 20 times (epoch = 20)) , we get a new accuracy.
#Highest Accuracy obtained = 93.65%
#Lowest Accuracy obtained = 64.68%
```

```
def plot_learningCurve(history, epoch):
  # Plot training & validation accuracy values
  epoch_range = range(1, epoch+1)
  pylt.plot(epoch_range, history.history['accuracy'])
  pylt.plot(epoch_range, history.history['val_accuracy'])
  pylt.title('Model accuracy')
  pylt.ylabel('Accuracy')
  pylt.xlabel('Epoch')
  pylt.legend(['Train', 'Val'], loc='upper left')
  pylt.show()

  pylt.plot(epoch_range, history.history['loss'])
  pylt.plot(epoch_range, history.history['val_loss'])
  pylt.title('Model loss')
  pylt.ylabel('Loss')
  pylt.xlabel('Epoch')
  pylt.legend(['Train', 'Val'], loc='upper left')
  pylt.show()
```

```
plot_learningCurve(history, epochs)
```





```
#Lets add MaxPool

epochs = 50
cnn_model = Sequential()
cnn_model.add(Conv1D(32, 2, activation='relu', input_shape = X_train[0].shape))
cnn_model.add(BatchNormalization())
cnn_model.add(MaxPool1D(2))
cnn_model.add(Dropout(0.2))

cnn_model.add(Conv1D(64, 2, activation='relu'))
cnn_model.add(BatchNormalization())
cnn_model.add(MaxPool1D(2))
cnn_model.add(Dropout(0.5))

cnn_model.add(Flatten())
cnn_model.add(Dense(64, activation='relu'))
cnn_model.add(Dropout(0.5))

cnn_model.add(Dense(1, activation='sigmoid'))

cnn_model.compile(optimizer=Adam(lr=0.0001), loss = 'binary_crossentropy', metrics=['accuracy']
)
history = cnn_model.fit(X_train, Y_train, epochs=epochs, validation_data=(X_test, Y_test),
verbose=1)
plot_learningCurve(history, epochs)
```

```
Epoch 1/50
25/25 [==============================] - 1s 16ms/step - loss: 1.0922 - accuracy: 0.5565 - val_loss: 0.6
643 - val_accuracy: 0.7056
Epoch 2/50
25/25 [==============================] - 0s 5ms/step - loss: 0.8918 - accuracy: 0.6442 - val_loss: 0.63
82 - val_accuracy: 0.8173
Epoch 3/50
25/25 [==============================] - 0s 4ms/step - loss: 0.7253 - accuracy: 0.7052 - val_loss: 0.60
79 - val_accuracy: 0.8274
Epoch 4/50
25/25 [==============================] - 0s 5ms/step - loss: 0.6080 - accuracy: 0.7421 - val_loss: 0.57
38 - val_accuracy: 0.8274
Epoch 5/50
25/25 [==============================] - 0s 4ms/step - loss: 0.6036 - accuracy: 0.7497 - val_loss: 0.54
02 - val_accuracy: 0.8223
Epoch 6/50
25/25 [==============================] - 0s 4ms/step - loss: 0.5193 - accuracy: 0.7789 - val_loss: 0.50
73 - val_accuracy: 0.8325
Epoch 7/50
25/25 [==============================] - 0s 4ms/step - loss: 0.4454 - accuracy: 0.8323 - val_loss: 0.47
65 - val_accuracy: 0.8426
Epoch 8/50
25/25 [==============================] - 0s 4ms/step - loss: 0.4667 - accuracy: 0.8259 - val_loss: 0.44
```

```
#Thus , here I conlude the model of CNN.
```

Evalution Metric(s) if applied on CNN model ---
Accuracy : 93.65%

IMPLEMENTATION OF DECISION TREE ALGORITHM

```
#SOFT COMPUTING - J COMPONENT PROJECT

'''
TEAM ---

SOUBHIK SINHA (19BIT0303)
ROHIT K (19BIT0318)
ADARSH KUMAR SINGH (19BIT0253)
'''

#TOPIC : CREDIT CARD FRAUD DETECTION
'''
COMPONENTS / MODULES --
DECISION TREE
'''
```

```
'\nCOMPONENTS / MODULES -- \nDECISION TREE\n'
```

```
#Lets import the packages and load data

import pandas as ps

cred = ps.read_csv("D:\COLLEGE_4th_SEM\E2_Soft_Computing_(J)\creditcard.csv")
```

```
print(f"Dataset Shape : \n {cred.shape}")
```

```
Dataset Shape :
 (284807, 31)
```

```
print(cred.describe())
```

```
                Time            V1            V2            V3            V4  \
count  284807.000000  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean    94813.859575  1.759061e-12 -8.251130e-13 -9.654937e-13  8.321385e-13
std     47488.145955  1.958696e+00  1.651309e+00  1.516255e+00  1.415869e+00
min         0.000000 -5.640751e+01 -7.271573e+01 -4.832559e+01 -5.683171e+00
25%     54201.500000 -9.203734e-01 -5.985499e-01 -8.903648e-01 -8.486401e-01
50%     84692.000000  1.810880e-02  6.548556e-02  1.798463e-01 -1.984653e-02
75%    139320.500000  1.315642e+00  8.037239e-01  1.027196e+00  7.433413e-01
max    172792.000000  2.454930e+00  2.205773e+01  9.382558e+00  1.687534e+01
```

…..(continued)….

```
                V25           V26           V27           V28         Amount  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05  284807.000000
mean  -6.987102e-13 -5.617874e-13  3.332082e-12 -3.518874e-12      88.349619
std    5.212781e-01  4.822270e-01  4.036325e-01  3.300833e-01     250.120109
min   -1.029540e+01 -2.604551e+00 -2.256568e+01 -1.543008e+01       0.000000
25%   -3.171451e-01 -3.269839e-01 -7.083953e-02 -5.295979e-02       5.600000
50%    1.659350e-02 -5.213911e-02  1.342146e-03  1.124383e-02      22.000000
75%    3.507156e-01  2.409522e-01  9.104512e-02  7.827995e-02      77.165000
max    7.519589e+00  3.517346e+00  3.161220e+01  3.384781e+01   25691.160000

               Class
count  284807.000000
mean        0.001727
std         0.041527
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max         1.000000

[8 rows x 31 columns]
```

```
cred.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431

5 rows × 31 columns

```
cred
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...
...	...	...	...	...	...	...	...	...	...	...	...
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7.305334	1.914428	...
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.294869	0.584800	...
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.708417	0.432454	...
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.679145	0.392087	...
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.414650	0.486180	...

284807 rows × 31 columns

```
print(f"Columns / Feature / Variable names : \n {cred.columns}")
```

```
Columns / Feature / Variable names :
 Index(['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',
       'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20',
       'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'Amount',
       'Class'],
      dtype='object')
```

```
print(f"Unique values of target variable / Attribute : \n {cred['Class'].unique()}")
```

```
Unique values of target variable / Attribute :
 [0 1]
```

```
# 0 means valid , 1 means fraudulent
```

```
print(f"Number of samples under each target value (0 and 1) : \n {cred['Class'].value_counts()}
")
```

```
Number of samples under each target value (0 and 1) :
 0    284315
1       492
Name: Class, dtype: int64
```

```
#Remove useless features

cred = cred.drop(['Time'], axis=1)
print(f"Features left after removal of 'Time' feature : \n{cred.columns}")
```

```
Features left after removal of 'Time' feature :
Index(['V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10', 'V11',
       'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20', 'V21',
       'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'Amount', 'Class'],
      dtype='object')
```

```
#Checking for NULL or NaN values

print(f"Dataset information : \n {cred.info()}")
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 30 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   V1      284807 non-null  float64
 1   V2      284807 non-null  float64
 2   V3      284807 non-null  float64
 3   V4      284807 non-null  float64
 4   V5      284807 non-null  float64
 5   V6      284807 non-null  float64
 6   V7      284807 non-null  float64
 7   V8      284807 non-null  float64
 8   V9      284807 non-null  float64
 9   V10     284807 non-null  float64
 10  V11     284807 non-null  float64
 11  V12     284807 non-null  float64
 12  V13     284807 non-null  float64
 13  V14     284807 non-null  float64
 14  V15     284807 non-null  float64
 15  V16     284807 non-null  float64
 16  V17     284807 non-null  float64
 17  V18     284807 non-null  float64
 18  V19     284807 non-null  float64
```

```
#Data Transformation

print(f"Some Amount Column / Feature values : \n {cred['Amount'][0:4]}")
```

```
Some Amout Column / Feature values :
 0    149.62
1      2.69
2    378.66
3    123.50
Name: Amount, dtype: float64
```

```
#Data Preprocessing

from sklearn.preprocessing import StandardScaler
cred['norm_amount'] = StandardScaler().fit_transform(
cred['Amount'].values.reshape(-1,1))
cred = cred.drop(['Amount'], axis=1)
print(f"Some Amount Couln values - After application of StandardScalar : \n {cred
['norm_amount'][0:4]}")
```

```
Some Amount Couln values - After application of StandardScalar :
 0    0.244964
1   -0.342475
2    1.160686
3    0.140534
Name: norm_amount, dtype: float64
```

```
#Creation of features and target

X = cred.drop(['Class'], axis=1)
Y = cred[['Class']]
```

```
#Splitting the dataset for training and testing

from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=0)
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
```

```
(199364, 29)
(85443, 29)
(199364, 1)
(85443, 1)
```

```
#Lets create the model for Decision Tree Algo.

from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

def decision_tree_classification(X_train, Y_train, X_test, Y_test):
    dec_tree_classf = DecisionTreeClassifier()
    print("START MODEL TRAINING ...")
    dec_tree_classf.fit(X_train, Y_train.values.ravel())
    print("COMPLETTION OF MODEL TRAINING")
    accuracy = dec_tree_classf.score(X_test, Y_test)
    print(f'Accuracy : {accuracy}')
    Y_predict = dec_tree_classf.predict(X_test)

    #CONFUSION MATRIX
    print(f"Confusion Matrix : \n {confusion_matrix(Y_test, Y_predict)}")

    # Classification report for F1_score
    print(f"Classification Report :- \n {classification_report(Y_test, Y_predict)}")

# DECISION TREE CLASSIFICATION MODEL CALLED
decision_tree_classification(X_train, Y_train, X_test, Y_test)
```

```
START MODEL TRAINING ...
COMPLETTION OF MODEL TRAINING
Accuracy : 0.9992743700478681
Confusion Matrix :
 [[85270    26]
 [   36   111]]
Classification Report :-
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     85296
           1       0.81      0.76      0.78       147

    accuracy                           1.00     85443
   macro avg       0.90      0.88      0.89     85443
weighted avg       1.00      1.00      1.00     85443
```

```
'''
If converted the scores into percentages to approximate values ---

Accuracy = 99.92%
Precision = 81%
Recall = 76%
F1_Score = 78%
'''

#Thus , I conclude the model of Decision Tree Classification and the scores of metrics.
```

Evaluation Metrics applied to Decision Tree Algorithm Model –

Accuracy : 99.92%
Precision : 81%
Recall : 76%
F1 Score : 78%

IMPLEMENTATION OF K-NN (K – NEAREST NEIGHBOUR)

```
#SOFT COMPUTING - J COMPONENT PROJECT

'''
TEAM ---

SOUBHIK SINHA (19BIT0303)
ROHIT K (19BIT0318)
ADARSH KUMAR SINGH (19BIT0253)
'''

#TOPIC : CREDIT CARD FRAUD DETECTION
'''
COMPONENTS / MODULES --
SVM (SUPPORT VECTOR MACHINE)
'''
```

```
'\nCOMPONENTS / MODULES -- \nSVM (SUPPORT VECTOR MACHINE)\n'
```

```
#Import the packages

import numpy as ny
import pandas as ps
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import normalize
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
import time
```

```
tic=time.time()
```

```
#Importing dataset

cred=ps.read_csv("D:\COLLEGE_4th_SEM\E2_Soft_Computing_(J)\creditcard.csv")

cred.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431

5 rows x 31 columns

```
#Lets Randomize the whole dataset

cred=cred.sample(frac=1)
```

```
#We don't need Time and Class feature as they are unused - saw in other algos. models

features=cred.drop(["Time","Class"],axis=1)
```

```
labels=ps.DataFrame(cred[["Class"]])
```

```
features_array=features.values
```

```
labels_array=labels.values
```

```
train_feat,test_feat,train_lab,test_lab=train_test_split(features_array,labels_array,
train_size=0.90)
```

```
train_feat=normalize(train_feat)
test_feat=normalize(test_feat)
```

```
#KNN (K-NEAREST NEIGHBOUR) CLASSIFICATION

KNN=KNeighborsClassifier(n_neighbors=5,algorithm="kd_tree",n_jobs=-1)
KNN.fit(train_feat,train_lab.ravel())
KNN_predicted_test_lab=KNN.predict(test_feat)
```

```
#Confusion matrix for KNN

trueNeg,falsePos,falseNeg,truePos=confusion_matrix(test_lab,KNN_predicted_test_lab).ravel()
```

```
#Evaluation Metrics for KNN
```

```
accuracy = accuracy_score(test_lab,KNN_predicted_test_lab)
precison = precision_score(test_lab,KNN_predicted_test_lab)
recall = recall_score(test_lab,KNN_predicted_test_lab)
f1_score = f1_score(test_lab,KNN_predicted_test_lab)
```

```
#Let us print all the results at once

print("Confusion Matrix of KNN")
print("True Negative = ",trueNeg," || False Positive = ",falsePos)
print("False Negative = ",falseNeg," || True Positive = ",truePos)
print(" ")
print("SCORES VIA METRICS --")
print("Accuracy ==>",accuracy)
print("Precison ==>",precison)
print("Recall ==>",recall)
print("F1_Score ==>",f1_score)
```

```
Confusion Matrix of KNN
True Negative =  28421  || False Positive =  7
False Negative =  12  || True Positive =  41

SCORES VIA METRICS --
Accuracy ==> 0.9993328885923949
Precison ==> 0.8541666666666666
Recall ==> 0.7735849056603774
F1_Score ==> 0.811881188118812
```

```
'''
If converted into percentages , then the scores ia metrics would be (approximately) ---

Accuracy = 99.93%
Precision = 85.42%
ReCall = 77.36%
F1_Score = 81.19%
'''
```

```
'\nIf converted into percentages , then the scores ia metrics would be (approximately) ---\n\nAccuracy =
99.93%\nPrecision = 85.42%\nReCall = 77.36%\nF1_Score = 81.19%\n'
```

```
#Thus , I conclude the model of KNN (Classification)
```

Evaluation Metrics applied to K-NN model –

Accuracy : 99.93%
Precision : 85.42%
Recall : 77.36%
F1 Score : 81.19%

IMPLEMENTATION OF SVM (SUPPORT VECTOR MODEL)

```
#SOFT COMPUTING - J COMPONENT PROJECT

'''
TEAM ---

SOUBHIK SINHA (19BIT0303)
ROHIT K (19BIT0318)
ADARSH KUMAR SINGH (19BIT0253)
'''

#TOPIC : CREDIT CARD FRAUD DETECTION
'''
COMPONENTS / MODULES --
SVM (SUPPORT VECTOR MACHINE)
'''
```

```
'\nCOMPONENTS / MODULES -- \nSVM (SUPPORT VECORE MACHINE)\n'
```

```
#Importing all the required packages

import numpy as ny
import pandas as ps
import matplotlib.pyplot as pypt
import seaborn as sbn
import warnings
warnings.filterwarnings("ignore")

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

from IPython.core.display import HTML
HTML("""
<style>
.output_png {
    display: table-cell;
    text-align: center;
    vertical-align: middle;
}
</style>
""")
```

```
# Importing / loading dataset

cred = ps.read_csv("D:\COLLEGE_4th_SEM\E2_Soft_Computing_(J)\creditcard.csv")
cred.head(5)
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431

5 rows × 31 columns

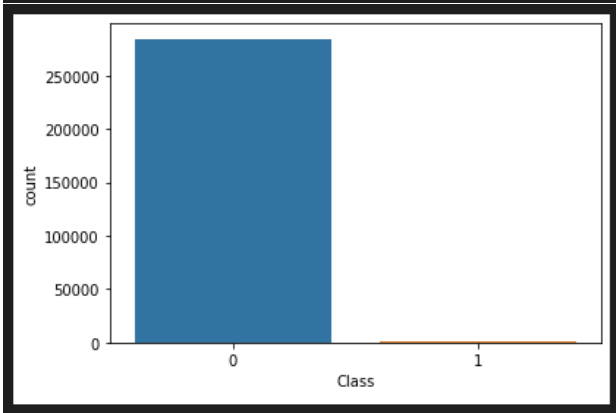
```
#Lets remove time here - reason - we already saw that in ANN

cred = cred.drop("Time", axis=1)
```

```
#Lets Standardize Amount

from sklearn import preprocessing
scaler = preprocessing.StandardScaler()
```

```
cred['std_Amount'] = scaler.fit_transform(cred['Amount'].values.reshape (-1,1))

cred = cred.drop("Amount", axis=1)
```

```
sbn.countplot(x="Class" , data = cred)
```

```
<AxesSubplot:xlabel='Class', ylabel='count'>
```




```
import imblearn
from imblearn.under_sampling import RandomUnderSampler

und_sam = RandomUnderSampler(sampling_strategy=0.5)
```

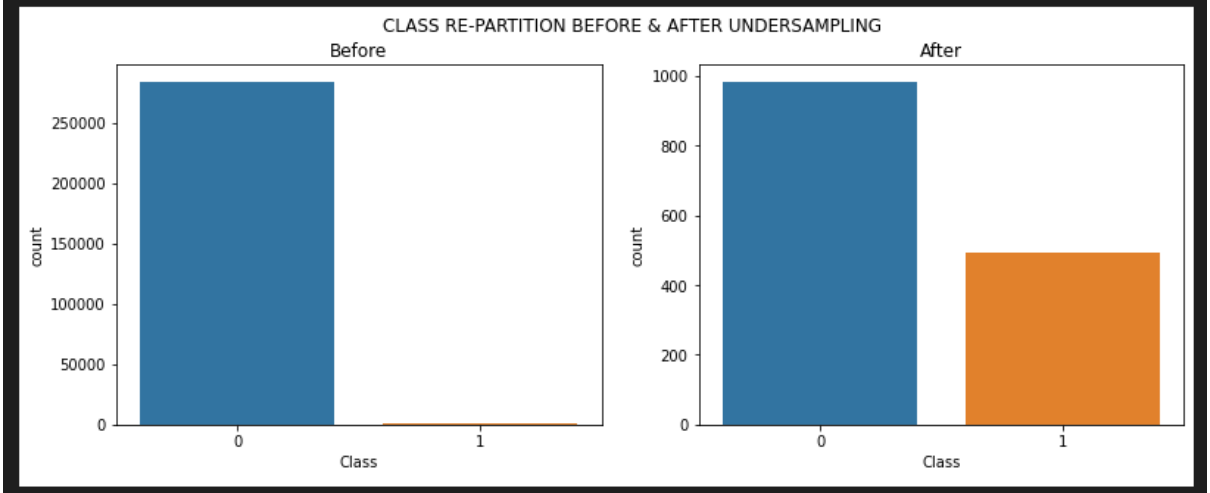
```
cols = cred.columns.tolist()
cols = [c for c in cols if c not in ["Class"]]
target = "Class"
```

```
X = cred[cols]
Y = cred[target]

X_under, Y_under = und_sam.fit_resample(X, Y)
```

```
from pandas import DataFrame
test = ps.DataFrame(Y_under, columns = ['Class'])
```

```
fig, axs = pypt.subplots(ncols=2, figsize=(13,4.5))
sbn.countplot(x="Class", data=cred, ax=axs[0])
sbn.countplot(x="Class", data=test, ax=axs[1])

fig.suptitle("CLASS RE-PARTITION BEFORE & AFTER UNDERSAMPLING")
a1=fig.axes[0]
a1.set_title("Before")
a2=fig.axes[1]
a2.set_title("After")
```

```
Text(0.5, 1.0, 'After')
```




```
#Now at this very point , I shall say that the dataset is balanced
```

```
#Let us split the data for training and testing purpose

from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(X_under, Y_under, test_size=0.2,
random_state=1)
```

```
#FROM HERE ONWARDS WE SHALL DO MODELLING ! I MEAN CREATING THE MODEL OF SVM

#Importing packages

from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.neural_network import MLPClassifier
```

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from keras.layers import Dropout
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import BatchNormalization

from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from sklearn.metrics import auc
from sklearn.metrics import precision_recall_curve
```

```
#Lets train the model for SVM

train_model = SVC(probability=True, random_state=2)
SVC_model = train_model.fit(X_train, Y_train)
```

```
#Lets predict

Y_predict_SVM = train_model.predict(X_test)
```

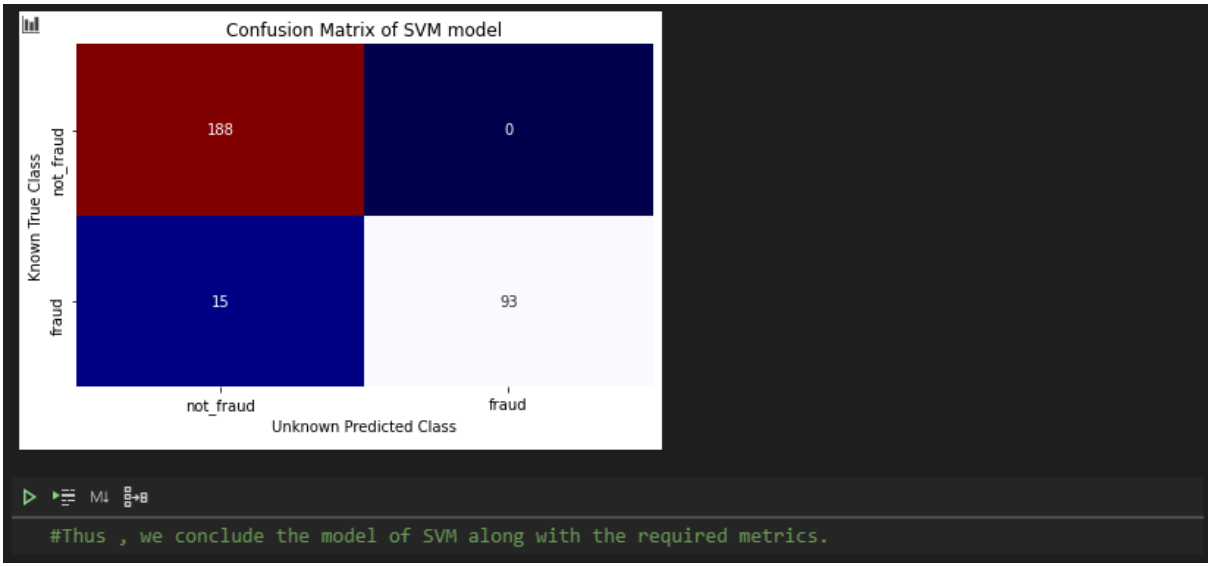
```
#EVALUATION MERICS

print("Accuracy of the SVM model : ",metrics.accuracy_score(Y_test, Y_predict_SVM))
print("Precision of the SVM model : ",metrics.precision_score(Y_test, Y_predict_SVM))
print("Recall of the SVM model : ",metrics.recall_score(Y_test, Y_predict_SVM))
print("F1_Score of the SVM model : ",metrics.f1_score(Y_test, Y_predict_SVM))
```

```
Accuracy SVM: 0.9493243243243243
Precision SVM: 1.0
Recall SVM: 0.8611111111111112
F1 Score SVM: 0.9253731343283582
```

```
'''
Thus , if converted to percentage (approximately) ---
Accuracy =  94.93%
Precision = 100.0%
Recall = 86.11%
F1_Score = 92.54%
'''
```

```
'\nThus , if converted to percentage (approximately) ---\nAccuracy =  94.93%\nPrecision = 100.0%\nRecall
= 86.11%\nF1_Score = 92.54%\n'
```

```
#Lets build the confusion matrix

mat_svm = confusion_matrix(Y_test, Y_predict_SVM)
cm_svm = ps.DataFrame(mat_svm, index=['not_fraud', 'fraud'], columns=['not_fraud', 'fraud'])

sbn.heatmap(cm_svm, annot=True, cbar=None, cmap="seismic", fmt = 'g')
pypt.title("Confusion Matrix of SVM model"), pypt.tight_layout()
pypt.ylabel("Known True Class"), pypt.xlabel("Unknown Predicted Class")
pypt.show()
```




```
#Thus , we conclude the model of SVM along with the required metrics.
```

Evaluation Metrics applied to SVM model ---

Accuracy : 94.93%
Precision : 100.0%
Recall : 86.11%
F1 Score : 92.54%

IMPLEMENTATION OF NAÏVE BAYES CLASSIFIER ---

```
#SOFT COMPUTING - J COMPONENT PROJECT

'''
TEAM ---

SOUBHIK SINHA (19BIT0303)
ROHIT K (19BIT0318)
ADARSH KUMAR SINGH (19BIT0253)
'''

#TOPIC : CREDIT CARD FRAUD DETECTION
'''
COMPONENTS / MODULES --
NAIVE BAYES CLASSIFIER
'''
```

```
'\nCOMPONENTS / MODULES -- \nNAIVE BAYES CLASSIFIER\n'
```

```
#This is a Gaussian Naive Bayes Classification


#Lets start with importing packages

import numpy as ny
import pandas as ps
import matplotlib.pyplot as pylt
from sklearn.metrics import confusion_matrix,auc,roc_auc_score
from sklearn.metrics import recall_score, precision_score, accuracy_score, f1_score
```

```
# Loading .CSV file
cred = ps.read_csv("D:\COLLEGE_4th_SEM\E2_Soft_Computing_(J)\creditcard.csv")
```

```
# Lets get the shape of the dataset
print(cred.shape)
```

```
(284807, 31)
```

```
# Obatining initial rows
print(cred.head())
```

```
   Time        V1        V2        V3        V4        V5        V6        V7  \
0   0.0 -1.359807 -0.072781  2.536347  1.378155 -0.338321  0.462388  0.239599
1   0.0  1.191857  0.266151  0.166480  0.448154  0.060018 -0.082361 -0.078803
2   1.0 -1.358354 -1.340163  1.773209  0.379780 -0.503198  1.800499  0.791461
3   1.0 -0.966272 -0.185226  1.792993 -0.863291 -0.010309  1.247203  0.237609
4   2.0 -1.158233  0.877737  1.548718  0.403034 -0.407193  0.095921  0.592941

         V8        V9  ...       V21       V22       V23       V24       V25  \
0  0.098698  0.363787  ... -0.018307  0.277838 -0.110474  0.066928  0.128539
1  0.085102 -0.255425  ... -0.225775 -0.638672  0.101288 -0.339846  0.167170
2  0.247676 -1.514654  ...  0.247998  0.771679  0.909412 -0.689281 -0.327642
3  0.377436 -1.387024  ... -0.108300  0.005274 -0.190321 -1.175575  0.647376
4 -0.270533  0.817739  ... -0.009431  0.798278 -0.137458  0.141267 -0.206010

        V26       V27       V28  Amount  Class
0 -0.189115  0.133558 -0.021053  149.62      0
1  0.125895 -0.008983  0.014724    2.69      0
2 -0.139097 -0.055353 -0.059752  378.66      0
3 -0.221929  0.062723  0.061458  123.50      0
4  0.502292  0.219422  0.215153   69.99      0

[5 rows x 31 columns]
```

```
#Describing dataset
print(cred.describe())
```

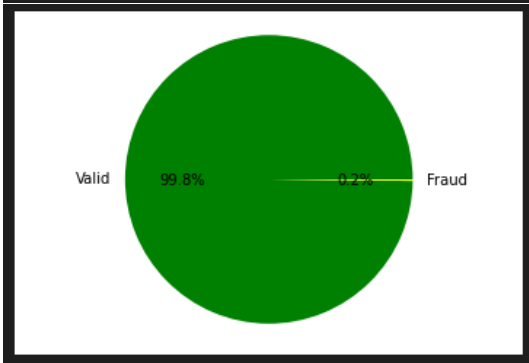
```
                Time            V1            V2            V3            V4  \
count  284807.000000  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean    94813.859575  1.759061e-12 -8.251130e-13 -9.654937e-13  8.321385e-13
std     47488.145955  1.958696e+00  1.651309e+00  1.516255e+00  1.415869e+00
min         0.000000 -5.640751e+01 -7.271573e+01 -4.832559e+01 -5.683171e+00
25%     54201.500000 -9.203734e-01 -5.985499e-01 -8.903648e-01 -8.486401e-01
50%     84692.000000  1.810880e-02  6.548556e-02  1.798463e-01 -1.984653e-02
75%    139320.500000  1.315642e+00  8.037239e-01  1.027196e+00  7.433413e-01
max    172792.000000  2.454930e+00  2.205773e+01  9.382558e+00  1.687534e+01

                 V5            V6            V7            V8            V9  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean   1.649999e-13  4.248366e-13 -3.054600e-13  8.777971e-14 -1.179749e-12
std    1.380247e+00  1.332271e+00  1.237094e+00  1.194353e+00  1.098632e+00
min   -1.137433e+02 -2.616051e+01 -4.355724e+01 -7.321672e+01 -1.343407e+01
25%   -6.915971e-01 -7.682956e-01 -5.540759e-01 -2.086297e-01 -6.430976e-01
50%   -5.433583e-02 -2.741871e-01  4.010308e-02  2.235804e-02 -5.142873e-02
75%    6.119264e-01  3.985649e-01  5.704361e-01  3.273459e-01  5.971390e-01
max    3.480167e+01  7.330163e+01  1.205895e+02  2.000721e+01  1.559499e+01

       ...           V21           V22           V23           V24  \
count  ...  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean   ... -3.405756e-13 -5.723197e-13 -9.725856e-13  1.464150e-12
std    ...  7.345240e-01  7.257016e-01  6.244603e-01  6.056471e-01
```

```
#Dataset Information
print(cred.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
```

```
#Lets check for 0 and 1 , for valid and fraudulant transactions , respectively

print("PIE CHART OF THE CLASS")
fig, ax = pylt.subplots(1, 1)
ax.pie(cred.Class.value_counts(),autopct='%1.1f%%', labels=['Valid','Fraud'], colors=['green',
'yellow'])
pylt.axis('equal')
pylt.ylabel('')
```

```
PIE CHART OF THE CLASS

Text(0, 0.5, '')
```

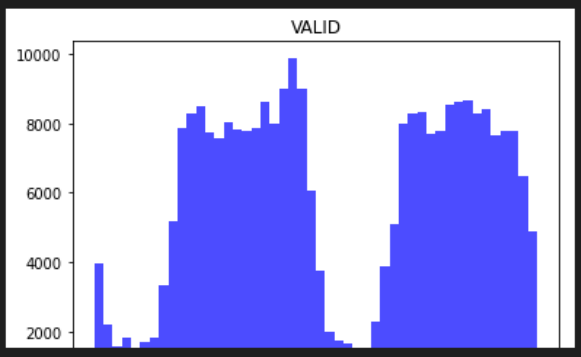



```
#Here the percentage of frauds is 0.2% which is the approximate result of 0.17%
#obtained from other models' code
```

```
#Lets check whether the feature Time has anything to do wtih the frauds.

print("TIME FEATURE")
cred["HOURS"] = cred["Time"]/3600 # convert to hours
print(cred["HOURS"].tail(5))

fig, (ax1, ax2) = pylt.subplots(2, 1, sharex = True, figsize=(6,10))
ax1.hist(cred.HOURS[cred.Class==0],bins=48,color='blue',alpha=0.7)
ax1.set_title('VALID')

ax2.hist(cred.HOURS[cred.Class==1],bins=48,color='orange',alpha=0.7)
ax2.set_title('FRAUD')
```

```
   pylt.xlabel('TIME IN HOURS')
   pylt.ylabel('NUMBER OF TRANSACTTIONS')
```

```
TIME FEATURE
284802    47.996111
284803    47.996389
284804    47.996667
284805    47.996667
284806    47.997778
Name: HOURS, dtype: float64

Text(0, 0.5, 'NUMBER OF TRANSACTTIONS')
```

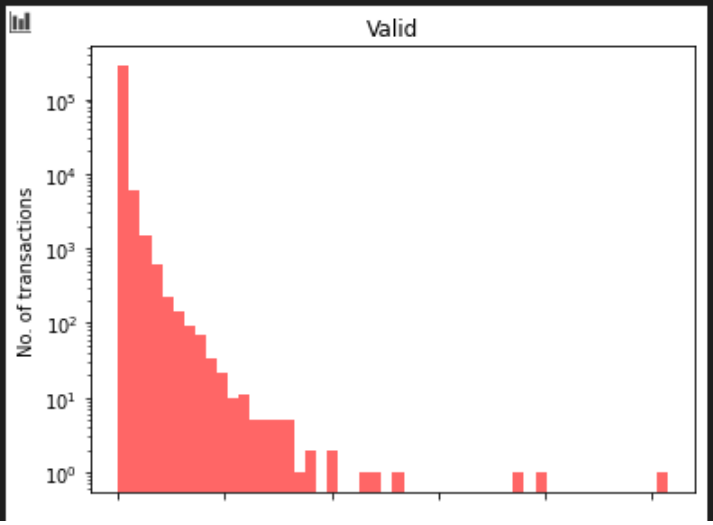


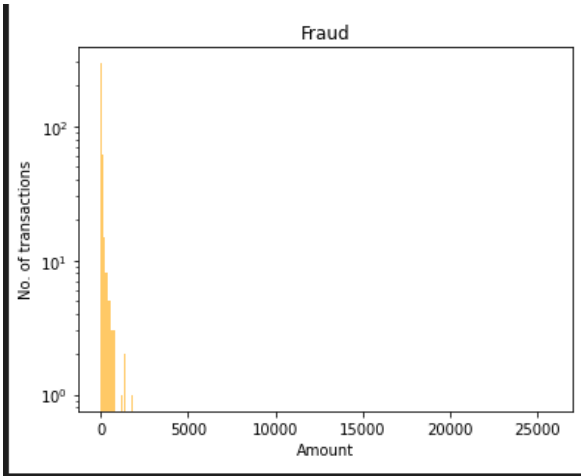

```
   #So we saw that Time has no effect on the frauds. Thus , we will drop it.

   cred = cred.drop(['Time'],axis = 1)
```

```
   #Lets check this time, whether the feature Smount has anything to do wtih frauds.

   fig, (ax3,ax4) = pylt.subplots(2,1, figsize = (6,10), sharex = True)
   ax3.hist(cred.Amount[cred.Class==0],bins=50,color='red',alpha=0.6)
   ax3.set_yscale('log')
   ax3.set_title('Valid')
   ax3.set_ylabel('No. of transactions')
```

```
ax4.hist(cred.Amount[cred.Class==1],bins=50,color='orange',alpha=0.6)
ax4.set_yscale('log')
ax4.set_title('Fraud')
ax4.set_xlabel('Amount')
ax4.set_ylabel('No. of transactions')
```

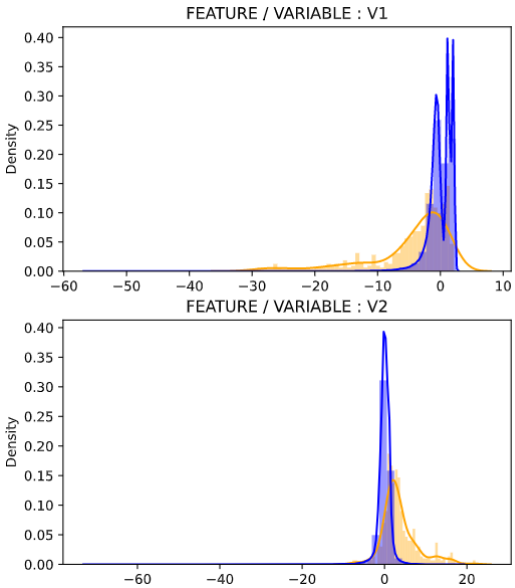
```
Text(0, 0.5, 'No. of transactions')
```

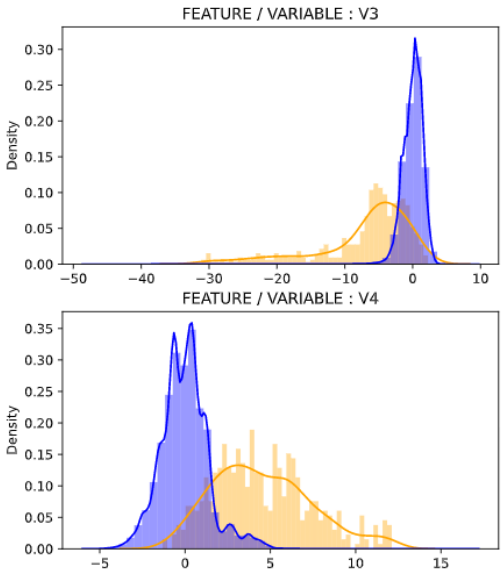


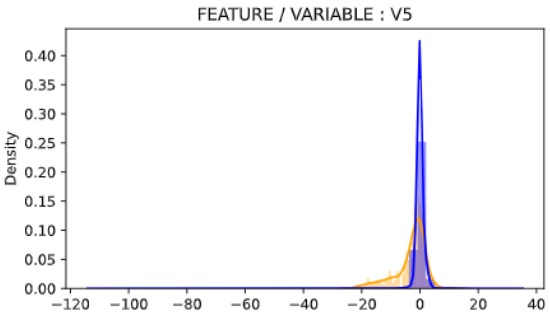


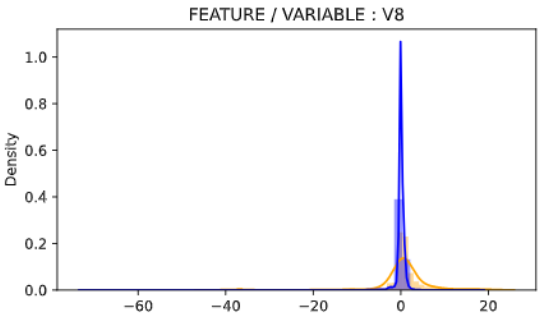

```
#Lets just Standardize the Amount feature.

from sklearn.preprocessing import StandardScaler
cred['SCALED_AMT'] = StandardScaler().fit_transform(cred['Amount'].values.reshape(-1,1))
cred = cred.drop(['Amount'],axis=1)
```

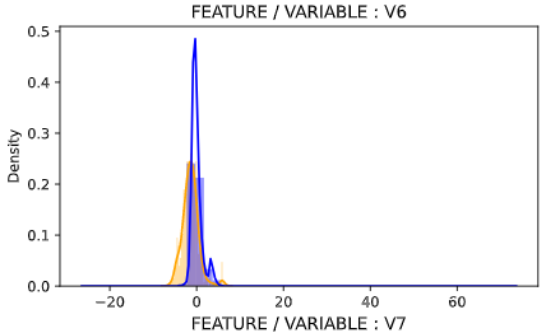
```
#Lets check the correlations among the features - the results obtained from PCA transformation.
#Those are - V1,V2,.....,V28 (total 28 columns).

import seaborn as sbn
import matplotlib.gridspec as gridspec
gs = gridspec.GridSpec(28, 1)
pylt.figure(figsize=(6,28*4))
for i, col in enumerate(cred[cred.iloc[:,0:28].columns]):
    ax5 = pylt.subplot(gs[i])
    sbn.distplot(cred[col][cred.Class == 1], bins=50, color='orange')
    sbn.distplot(cred[col][cred.Class == 0], bins=50, color='blue')
    ax5.set_xlabel('')
    ax5.set_title('FEATURE / VARIABLE : ' + str(col))
pylt.show()
```

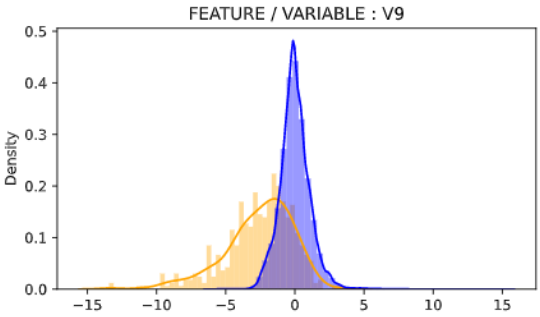


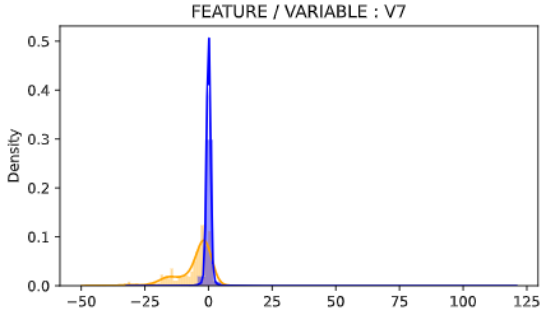


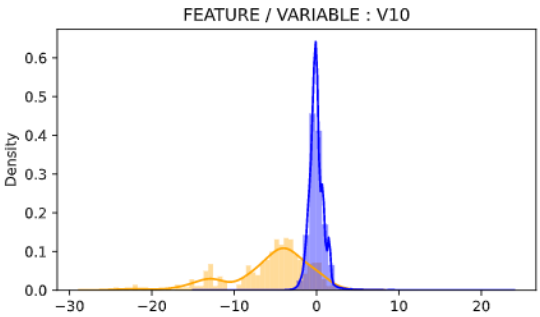

FEATURE / VARIABLE : V5
Density

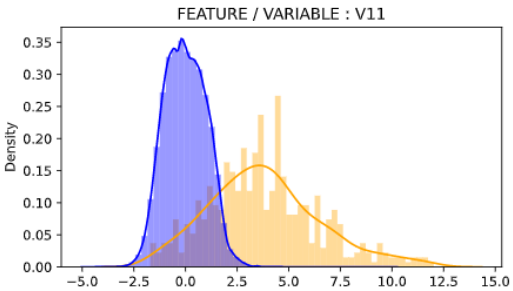

FEATURE / VARIABLE : V8
Density

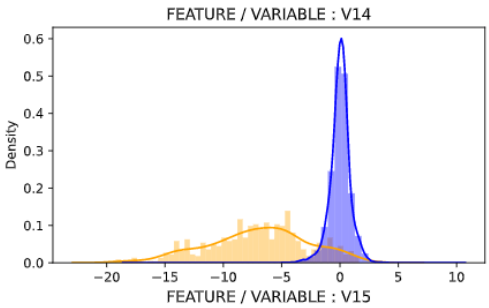

FEATURE / VARIABLE : V6
Density

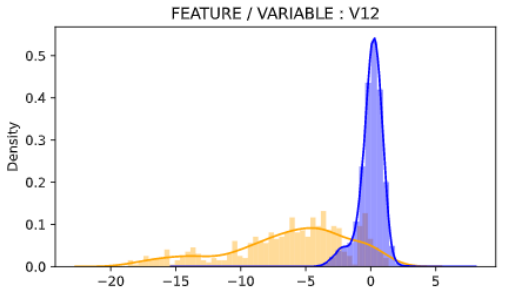

FEATURE / VARIABLE : V9
Density

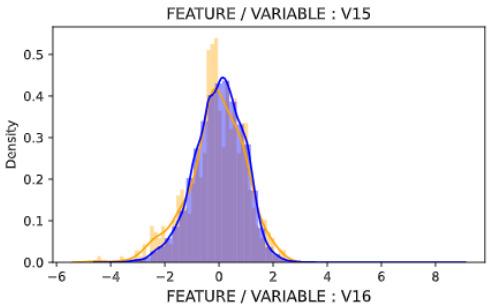

FEATURE / VARIABLE : V7
Density

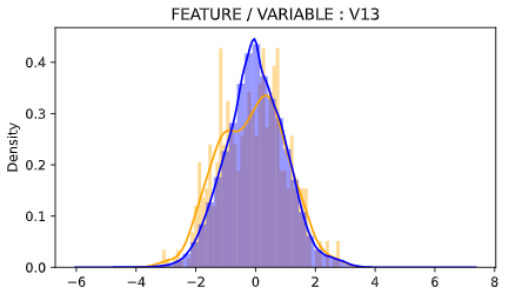

FEATURE / VARIABLE : V10
Density

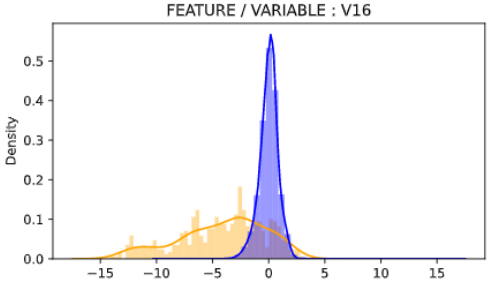

FEATURE / VARIABLE : V11
Density

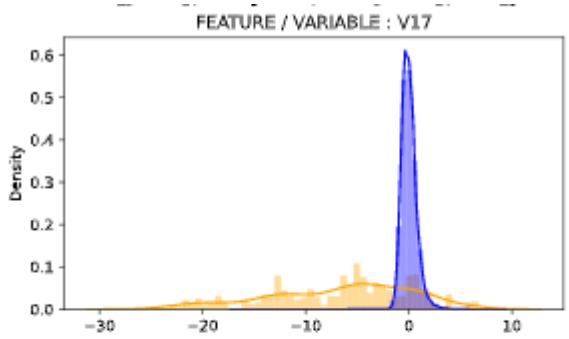

FEATURE / VARIABLE : V14
Density

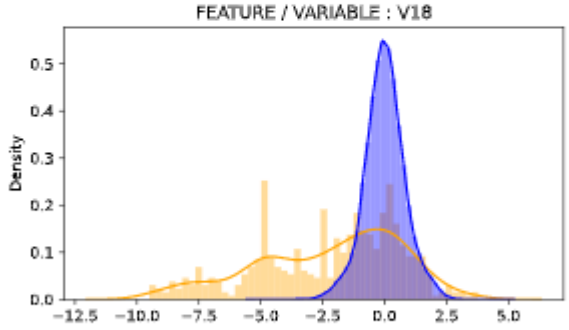

FEATURE / VARIABLE : V12
Density

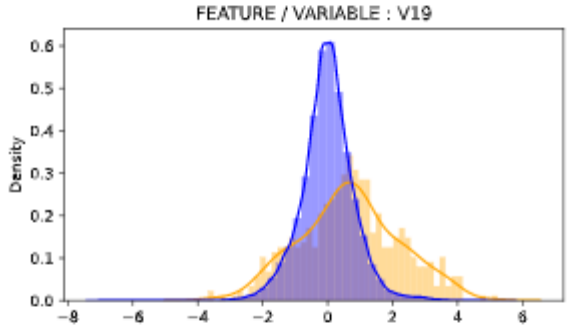

FEATURE / VARIABLE : V15
Density

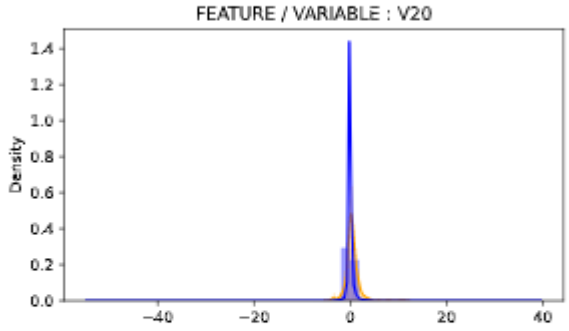

FEATURE / VARIABLE : V13
Density

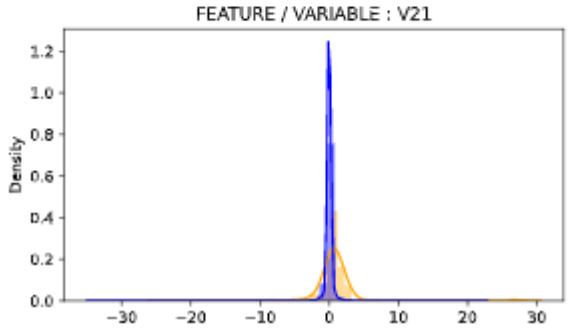

FEATURE / VARIABLE : V16
Density

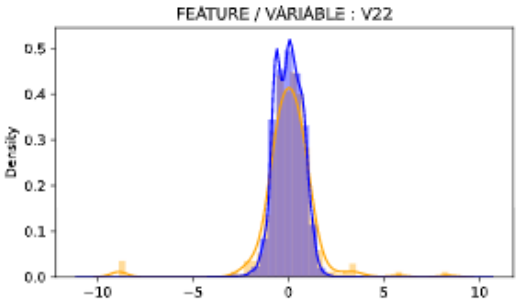

FEATURE / VARIABLE : V17
Density

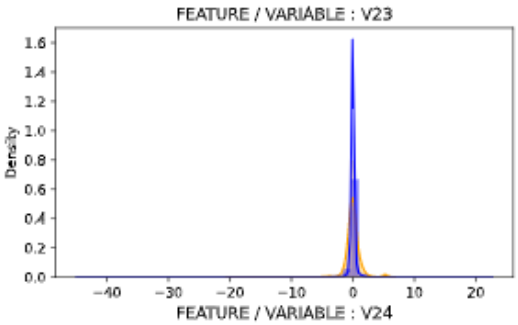

FEATURE / VARIABLE : V18
Density

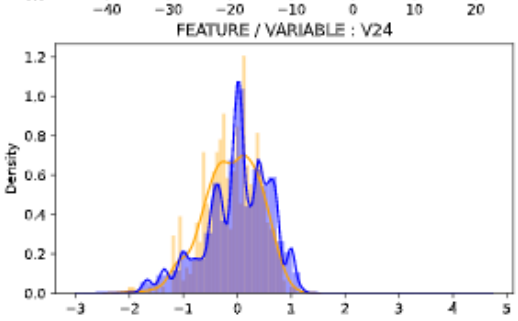

FEATURE / VARIABLE : V19
Density

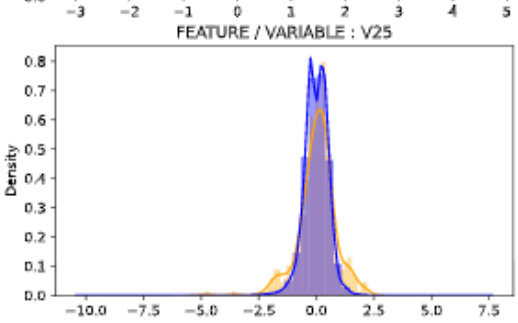

FEATURE / VARIABLE : V20
Density

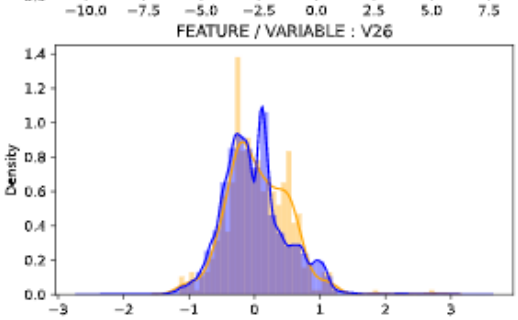

FEATURE / VARIABLE : V21
Density

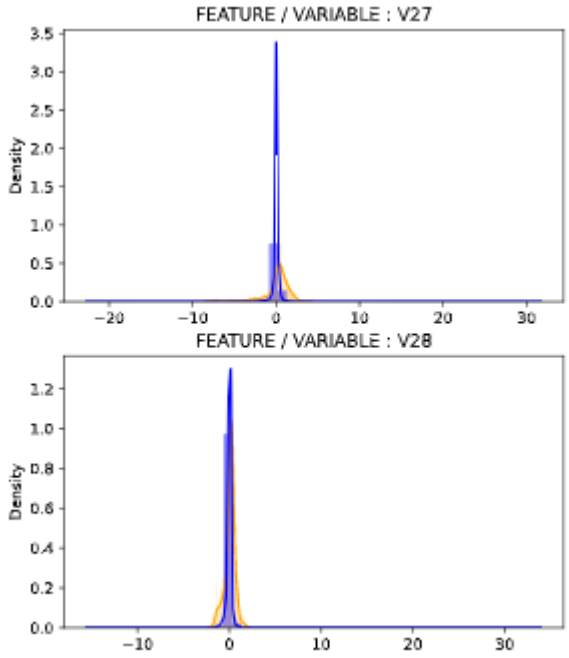

FEATURE / VARIABLE : V22
Density


FEATURE / VARIABLE : V23
Density


FEATURE / VARIABLE : V24
Density


FEATURE / VARIABLE : V25
Density


FEATURE / VARIABLE : V26
Density




```
#Now , lets just split the data in the dataset into training data and testing data. The ratio
#we are taking here will be 4:1 for testing and training data ,respectively.

def split_data(cred, drop_list):
    cred = cred.drop(drop_list,axis=1)
    print(cred.columns)

    #test-train data splitting
    from sklearn.model_selection import train_test_split
    Y = cred['Class'].values
    X = cred.drop(['Class'],axis=1).values
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,
                                                        random_state=42, stratify=Y)

    print("training-set-data size : ", len(Y_train),
      "\ntesting-set_data size : ", len(Y_test))
    print("fraud cases in test-set-data obtained : ", sum(Y_test))
    return X_train, X_test, Y_train, Y_test
```

```
#Creating Classifier

def get_predictions(classf, X_train, y_train, X_test):
    classf = classf
    classf.fit(X_train,Y_train)
    Y_predict = classf.predict(X_test)
    Y_predict_prob = classf.predict_proba(X_test)
    train_predict = classf.predict(X_train)

    print('training-set-data confusion matrix:\n', confusion_matrix(Y_train,train_predict))
    return Y_predict, Y_predict_prob
```

```
#Lets obtain the scores via metrics

def print_scores(Y_test,Y_predict,Y_predict_prob):
    print('testing-set-data confusion matrix : \n', confusion_matrix(Y_test,Y_predict))
    print("recall : ", recall_score(Y_test,Y_predict))
    print("precision : ", precision_score(Y_test,Y_predict))
    print("F1_score : ", f1_score(Y_test,Y_predict))
    print("accuracy : ", accuracy_score(Y_test,Y_predict))
```

```
#Let us see what will happen if some features are dropped one by one to check
#the classifier sensitivity. And the we shall see the metrics' scores

from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression

# CASE - 1 : DROP NOTHING
drop_list = []
X_train, X_test, Y_train, Y_test = split_data(cred, drop_list)
Y_predict, Y_predict_prob = get_predictions(GaussianNB(), X_train, Y_train, X_test)
print_scores(Y_test,Y_predict,Y_predict_prob)
```

```
Index(['V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10', 'V11',
       'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20', 'V21',
       'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'Class', 'HOURS',
       'SCALED_AMT'],
      dtype='object')
training-set-data size :  227845
testing-set_data size :  56962
fraud cases in test-set-data obtained :  98
training-set-data confusion matrix:
 [[222480   4971]
 [    69    325]]
testing-set-data confusion matrix :
 [[55535  1329]
 [   15    83]]
recall :  0.8469387755102041
precision :  0.058781869688385266
F1_score :  0.10993377483443707
accuracy :  0.9764053228468101
```

```
#Here , if scores are converted to percentages (approximately) --

'''
Accuracy = 97.64%
Precision = 5.88%
ReCall = 84.69%
F1_Score = 10.99%
'''
```

```
'\nAccuracy = 97.64%\nPrecision = 5.88%\nReCall = 84.69%\nF1_Score = 10.99%\n'
```

```
# CASE - 2 : DROP MAIN COMPONENTS WITH SAME DISTROs

drop_list = ['V28','V27','V26','V25','V24','V23','V22','V20','V15','V13','V8']
X_train, X_test, Y_train, Y_test = split_data(cred, drop_list)
Y_predict, Y_predict_prob = get_predictions(GaussianNB(), X_train, Y_train, X_test)
print_scores(Y_test,Y_predict,Y_predict_prob)
```

```
Index(['V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V9', 'V10', 'V11', 'V12',
       'V14', 'V16', 'V17', 'V18', 'V19', 'V21', 'Class', 'HOURS',
       'SCALED_AMT'],
      dtype='object')
training-set-data size :  227845
testing-set_data size :  56962
fraud cases in test-set-data obtained :  98
training-set-data confusion matrix:
 [[223967   3484]
 [    61    333]]
testing-set-data confusion matrix :
 [[55935   929]
 [   12    86]]
recall :  0.8775510204081632
precision :  0.08472906403940887
F1_score :  0.15453728661275834
accuracy :  0.9834802148800955
```

```
#Here , if scores are converted to percentages (approximately) --

'''
Accuracy = 98.34%
Precision = 8.47%
ReCall = 87.76%
F1_Score = 15.45%
'''
```

```
'\nAccuracy = 97.64%\nPrecision = 5.88%\nReCall = 84.69%\nF1_Score = 10.99%\n'
```

```
# CASE - 3 : DROP MAIN COMPONENTS AND TIME

drop_list = ['HOURS','V28','V27','V26','V25','V24','V23','V22','V20','V15','V13','V8']
X_train, X_test, Y_train, Y_test = split_data(cred, drop_list)
Y_predict, Y_predict_prob = get_predictions(GaussianNB(), X_train, Y_train, X_test)
print_scores(Y_test,Y_predict,Y_predict_prob)
```

```
Index(['V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V9', 'V10', 'V11', 'V12',
       'V14', 'V16', 'V17', 'V18', 'V19', 'V21', 'Class', 'SCALED_AMT'],
      dtype='object')
training-set-data size :  227845
testing-set_data size :  56962
fraud cases in test-set-data obtained :  98
training-set-data confusion matrix:
 [[223964   3487]
 [    60    334]]
testing-set-data confusion matrix :
 [[55936   928]
 [   12    86]]
recall :  0.8775510204081632
precision :  0.08481262327416174
F1_score :  0.15467625899280577
accuracy :  0.9834977704434535
```

```
#Here , if scores are converted to percentages (approximately) --

'''
Accuracy = 98.34%
Precision = 8.48%
ReCall = 87.76%
F1_Score = 15.47%
'''

#Thus , we can safely remove Time feature as nothing much has changed - it's all the same !
```

```
'\nAccuracy = 97.64%\nPrecision = 5.88%\nReCall = 84.69%\nF1_Score = 10.99%\n'
```

```
#Thus , we can safely remove Time feature as nothing much has changed - it's all the same !
```

```
'\nAccuracy = 97.64%\nPrecision = 5.88%\nReCall = 84.69%\nF1_Score = 10.99%\n'
```

```
# CASE - 4 : DROP MAIN COMPONENTS AND TIME AND SCALED_AMT (AMOUNT)

drop_list = ['SCALED_AMT','HOURS','V28','V27','V26','V25','V24','V23','V22','V20','V15','V13',
'V8']
X_train, X_test, Y_train, Y_test = split_data(cred, drop_list)
Y_predict, Y_predict_prob = get_predictions(GaussianNB(), X_train, Y_train, Y_test)
print_scores(Y_test,Y_predict,Y_predict_prob)
```

```
Index(['V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V9', 'V10', 'V11', 'V12',
       'V14', 'V16', 'V17', 'V18', 'V19', 'V21', 'Class'],
      dtype='object')
training-set-data size :  227845
testing-set_data size :  56962
fraud cases in test-set-data obtained :  98
```

```
cred = cred.drop(drop_list , axis = 1)
print(cred.columns)
```

```
Index(['V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V9', 'V10', 'V11', 'V12',
       'V14', 'V16', 'V17', 'V18', 'V19', 'V21', 'Class'],
      dtype='object')
```

```
#Thus , after going through some cases , I conclude the Naive Bayes Classifier Model
#Along with the scores via metrics.
```

Evaluation Metrics applied to Naïve Bayes Classifier –

Accuracy : 98.34%
Precision : 98.48%
Recall : 87.76%
F1 Score : 95.47%

IMPLEMENTATION OF RANDOM FOREST ALGORITHM AND **CONFUSION MATRIX ---**

```
#SOFT COMPUTING - J COMPONENT PROJECT

'''
TEAM ---

SOUBHIK SINHA (19BIT0303)
ROHIT K (19BIT0318)
ADARSH KUMAR SINGH (19BIT0253)
'''


#TOPIC : CREDIT CARD FRAUD DETECTION
'''
COMPONENTS / MODULES --
1. RANDOM FOREST
2. CONFUSION MATRIX
'''
```

```
'\nTEAM ---\n\nSOUBHIK SINHA (19BIT0303)\nROHIT K (19BIT0318)\nADARSH KUMAR SINGH (19BIT0253)\n'
```

```
#Lets import all the Important / required libraries. If you can't
#run a particular libray , that means it's not installed. Simply
#go to cmd and run --> pip install your_package_name. For this
#pip should be installed.
import pandas as ps
import numpy as ny
import matplotlib.pyplot as pypt
from matplotlib import gridspec
import seaborn as sbn
```

```
#Lets load the dataset. The dataset shall be in the form of .CSV file.
'''
Dataset taken from kaggle ---
Link : https://www.kaggle.com/mlg-ulb/creditcardfraud
'''

cred = ps.read_csv("D:\COLLEGE_4th_SEM\E2_Soft_Computing_(J)\creditcard.csv")
```

```
#Lets see what's inside the dataset
cred.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431

5 rows × 31 columns

```
#Data in the dataset should be given some shape
print(cred.shape)
print(cred.describe())
```


```
#Data in the dataset should be given some shape
print(cred.shape)
print(cred.describe())
```


```
(284807, 31)
                Time            V1            V2            V3            V4  \
count  284807.000000  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean    94813.859575  1.759061e-12 -8.251130e-13 -9.654937e-13  8.321385e-13
std     47488.145955  1.958696e+00  1.651309e+00  1.516255e+00  1.415869e+00
min         0.000000 -5.640751e+01 -7.271573e+01 -4.832559e+01 -5.683171e+00
25%     54201.500000 -9.203734e-01 -5.985499e-01 -8.903648e-01 -8.486401e-01
50%     84692.000000  1.810880e-02  6.548556e-02  1.798463e-01 -1.984653e-02
75%    139320.500000  1.315642e+00  8.037239e-01  1.027196e+00  7.433413e-01
max    172792.000000  2.454930e+00  2.205773e+01  9.382558e+00  1.687534e+01

                 V5            V6            V7            V8            V9  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean   1.649999e-13  4.248366e-13 -3.054600e-13  8.777971e-14 -1.179749e-12
std    1.380247e+00  1.332271e+00  1.237094e+00  1.194353e+00  1.098632e+00
min   -1.137433e+02 -2.616051e+01 -4.355724e+01 -7.321672e+01 -1.343407e+01
25%   -6.915971e-01 -7.682956e-01 -5.540759e-01 -2.086297e-01 -6.430976e-01
50%   -5.433583e-02 -2.741871e-01  4.010308e-02  2.235804e-02 -5.142873e-02
75%    6.119264e-01  3.985649e-01  5.704361e-01  3.273459e-01  5.971390e-01
max    3.480167e+01  7.330163e+01  1.205895e+02  2.000721e+01  1.559499e+01

       ...           V21           V22           V23           V24  \
count  ...  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean   ...  3.405756e-13  5.723197e-13 -9.725856e-13  1.464150e-12
```

```
#Now let us check how many fraud cases have dominated the loaded dataset
valid_trans = cred[cred['Class'] == 0]
fraud_trans = cred[cred['Class'] == 1]
ol_frac = len(fraud_trans)/float(len(valid_trans))
print(ol_frac)
print('No. of Valid Transactions : {}'.format(len(cred[cred['Class'] == 1])))
print('No. of Fraudulent Transactions : {}'.format(len(cred[cred['Class'] == 0])))

#adding {} in the above sentences mean the result will be displayed there. (For the above code)
```

```
0.0017304750013189597
No. of Valid Transactions : 492
No. of Fraudulent Transactions : 284315
```

```
# So if we take out the percentage of the fraudulent cases among the total transactions ,
#we shall find that --
```

```
print((492/284315)*100)
```

```
0.17304750013189596
```

```
#Approximately 0.17% of all the transactions are Fraudulent. But , we have a problem here -
#The data in the dataset provided is highly unbalanced. Lets just implement without balancing
#it. Later we shall think over it.

print("FRAUDULENT TRANSACTION AMOUNT DETAILS ARE AS SHOWN BELOW ---")
fraud_trans.Amount.describe()
```

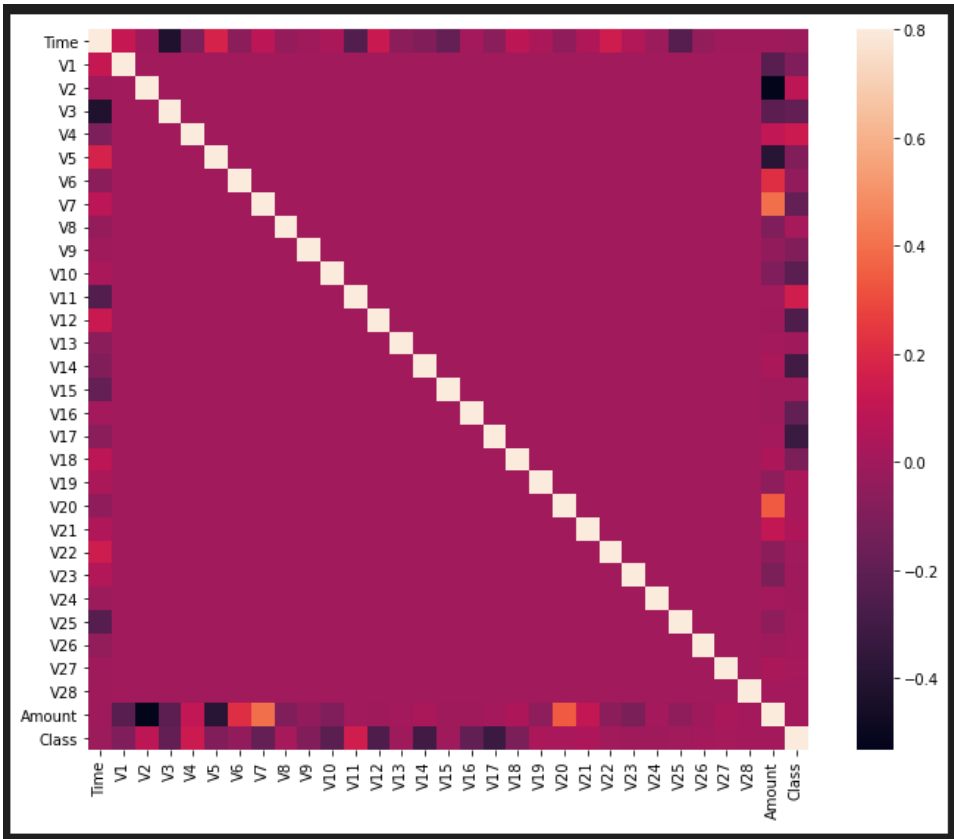
```
FRAUDULENT TRANSACTION AMOUNT DETAILS ARE AS SHOWN BELOW ---

count     492.000000
mean      122.211321
std       256.683288
min         0.000000
25%         1.000000
50%         9.250000
75%       105.890000
max      2125.870000
Name: Amount, dtype: float64
```

```
#Lets print for the Normal transaction (The valid ones)

print("VALID TRANSACTION AMOUNT DETAILS ---")
valid_trans.Amount.describe()
```

```
VALID TRANSACTION AMOUNT DETAILS ---

count    284315.000000
mean         88.291022
std         250.105092
min           0.000000
25%           5.650000
50%          22.000000
75%          77.050000
max       25691.160000
Name: Amount, dtype: float64
```

```
#Have a look over the mean pelf transaction for the Fraudulet cases - they are much higher
#than the valid ones.
```

```
#Lets create a Correlation Matrix.

'''
What is a Correlation Matrix ?

A Correlation matrix is a table showing correlation coefficients between sets of variables.
'''

cormatx = cred.corr()
figure = pypt.figure(figsize = (12,9))
sbn.heatmap(cormatx,vmax = 0.8 , square = True)
pypt.show()
```




```
#If you see the two consecuive coloured blocks of every feature (from V1,V2,V3....to amount
and class) , you will see there is not much correlation among the features. For example : just
#have a look at V2 & V5 - they are highly negatively corrlated. Also check V20 and Amount. But
#most of the features are not correlated. (See the blank pink space inside and the
#color-depiction-graph on the right)
```

```
#Lets separate teh values of X-axis and Y-axis from the given / loaded dataset
Y = cred["Class"]
X = cred.drop(['Class'],axis = 1)
print(Y.shape)
print(X.shape)

#Let us retrieve the values
y_Data = Y.values
x_Data = X.values
```

```
(284807,)
(284807, 30)
```

```
#As the dataset is quite big , why not we divide this in two parts - one for training the model
#and one for testing the model (that is trained).

from sklearn.model_selection import train_test_split

#from here , we shall divide the dataset for the above mentioned
x_Train, x_Test, y_Train, y_Test = train_test_split(
        x_Data, y_Data, test_size = 0.2, random_state = 42)
```

```
from sklearn.ensemble import RandomForestClassifier
ran_for_classfr = RandomForestClassifier()
ran_for_classfr.fit(x_Train, y_Train)

#Predicting the possible results
y_Predict = ran_for_classfr.predict(x_Test)
```

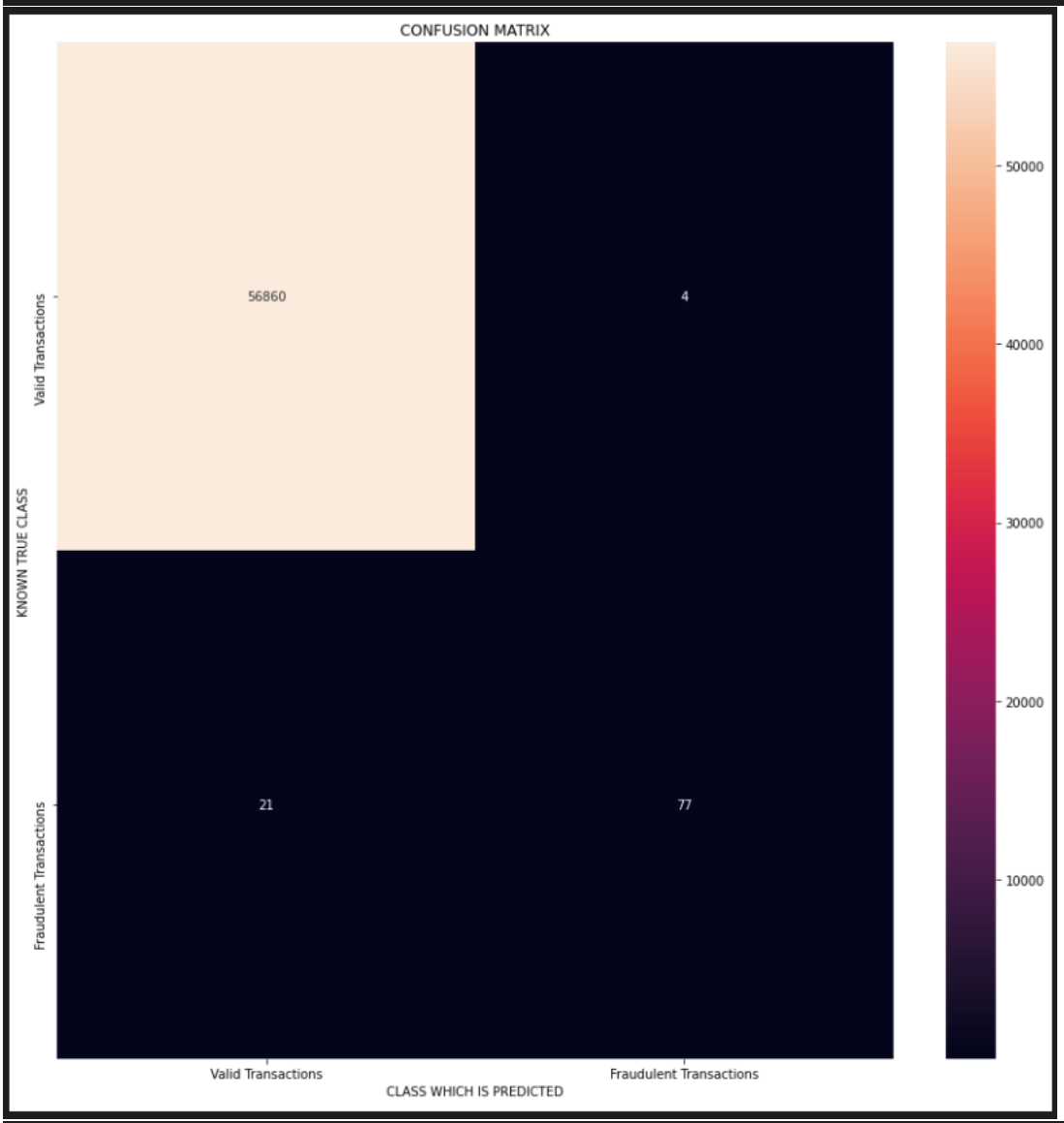
```
#Lets apply the metrics for evluation
from sklearn.metrics import f1_score, matthews_corrcoef
from sklearn.metrics import classification_report, accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_score, recall_score

precision = precision_score(y_Test, y_Predict)
print("The precision is : {}".format(precision))

accuracy = accuracy_score(y_Test, y_Predict)
print("The accuracy is : {}".format(accuracy))
```

```
f1_score = f1_score(y_Test, y_Predict)
print("The F1-Score is : {}".format(f1_score))

mat_corf = matthews_corrcoef(y_Test, y_Predict)
print("The Matthews correlation coefficient is : {}".format(mat_corf))

recall = recall_score(y_Test, y_Predict)  # Recall is also known as Sensitivity
print("The recall is : {}".format(recall))
```

```
The precision is : 0.9506172839506173
The accuracy is : 0.9995611109160493
The F1-Score is : 0.8603351955307262
The Matthews correlation coefficient is : 0.8640351019464072
The recall is : 0.7857142857142857
```

```
'''
If Converted the above taken out metrics values in percentage ---

Precision : 95.1%
Accuracy : 99.9% (That's almost 100% !!!)
F1_Score : 86.0%
Matthews Correlation Coefficient : 86.4%
Recall (OR Sensitivity) : 78.6%
'''
```

```
#Lets have a look in Confusion Matrix
LABELS = ['Valid Transactions', 'Fraudulent Transactions']
confusion_matrix = confusion_matrix(y_Test, y_Predict)
pypt.figure(figsize =(15, 15))
sbn.heatmap(confusion_matrix, xticklabels = LABELS,
            yticklabels = LABELS, annot = True, fmt ="d");
pypt.title("CONFUSION MATRIX")
pypt.ylabel('KNOWN TRUE CLASS')
pypt.xlabel('CLASS WHICH IS PREDICTED')
pypt.show()
```




```
#Thus , we have the confusion matrix of testing and predicting values
```

Evaluation Matrix applied to Random Forest Algorithm model ---

Precision : 95.1%
Accuracy : 99.9%
F1 Score : 86.0%
Recall (OR Sensitivity) : 78.6%

IMPLEMENTATION OF ANN (ARTIFICIAL NEURAL NETWORK)

```
#SOFT COMPUTING - J COMPONENT PROJECT

'''
TEAM ---

SOUBHIK SINHA (19BIT0303)
ROHIT K (19BIT0318)
ADARSH KUMAR SINGH (19BIT0253)
'''

#TOPIC : CREDIT CARD FRAUD DETECTION
'''
COMPONENTS / MODULES --
ARTIFICIAL NEURAL NETWORK (ANN)
'''
```

```
'\nCOMPONENTS / MODULES -- \nANN (ARTIFICIAL NEURAL NETWORK)\n'
```

```
#Importing Necessary Libraries

import pandas as ps
import numpy as ny
import matplotlib.pyplot as pylt
import seaborn as sbn

%matplotlib inline
sbn.set_style("whitegrid")
```

```
#Loading dataset from local

cred = ps.read_csv("D:\COLLEGE_4th_SEM\E2_Soft_Computing_(J)\creditcard.csv")
cred.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431

5 rows × 31 columns

```
#Lets Analyze the data from the dataset loaded

cred.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
 18  V18     284807 non-null  float64
```

```
ps.set_option("display.float","{:.2f}".format)
cred.describe()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...
count	284807.00	284807.00	284807.00	284807.00	284807.00	284807.00	284807.00	284807.00	284807.00	284807.00	... 2
mean	94813.86	0.00	-0.00	-0.00	0.00	0.00	0.00	-0.00	0.00	-0.00	...
std	47488.15	1.96	1.65	1.52	1.42	1.38	1.33	1.24	1.19	1.10	...
min	0.00	-56.41	-72.72	-48.33	-5.68	-113.74	-26.16	-43.56	-73.22	-13.43	...
25%	54201.50	-0.92	-0.60	-0.89	-0.85	-0.69	-0.77	-0.55	-0.21	-0.64	...
50%	84692.00	0.02	0.07	0.18	-0.02	-0.05	-0.27	0.04	0.02	-0.05	...
75%	139320.50	1.32	0.80	1.03	0.74	0.61	0.40	0.57	0.33	0.60	...
max	172792.00	2.45	22.06	9.38	16.88	34.80	73.30	120.59	20.01	15.59	...

8 rows × 31 columns

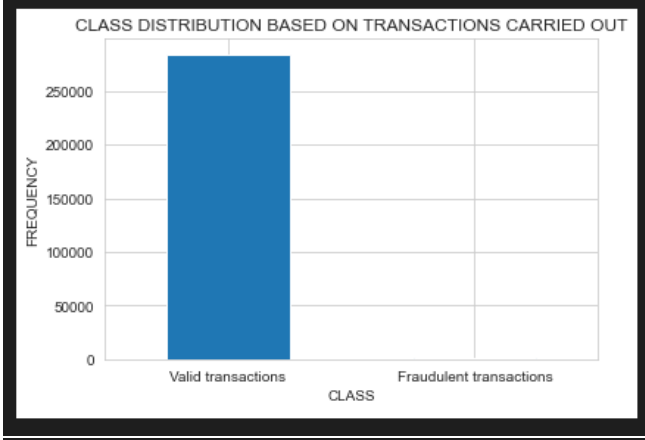
```
#Lets hunt for the missing values in the dataset. (NULL VALUES)
#Missing values create a lot of problem in analysis.

cred.isnull().sum().sum()
```

```
0
```

```
cred.columns()
```

```
#Transformed Variables -- Time , Amount , Class (0 OR 1)

LABELS = ["Valid transactions", "Fraudulent transactions"]

count_classes = ps.value_counts(cred['Class'], sort = True)
count_classes.plot(kind = 'bar', rot=0)
pylt.title("CLASS DISTRIBUTION BASED ON TRANSACTIONS CARRIED OUT")
pylt.xticks(range(2), LABELS)
pylt.xlabel("CLASS")
pylt.ylabel("FREQUENCY")
```

```
Text(0, 0.5, 'FREQUENCY')
```




```
cred.Class.value_counts()
```

```
0    284315
1       492
Name: Class, dtype: int64
```

```
# In the above execution , 0 means valid transactions , 1 means invalid OR fraudulent
# transaction. Thus we can see there are 284315 valid transactions
# and 492 fraudulent transactions , which is accounting 0.17% of the total transactions
```

```
# As the number of valid transactions are significantly high , the model shall assume
# that theere are no frauds. But assumption for / by a model is dangerous.
```

```
#Lets again obtain the valid and fraudulent transactions
fraud = cred[cred['Class'] == 1]
valid = cred[cred['Class'] == 0]

print(f"Fraudulent Transactions Shape : {fraud.shape}")
print(f"Valid Transactions Shape : {valid.shape}")
```

```
Fraudulent Transactions Shape : (492, 31)
Valid Transactions Shape : (284315, 31)
```

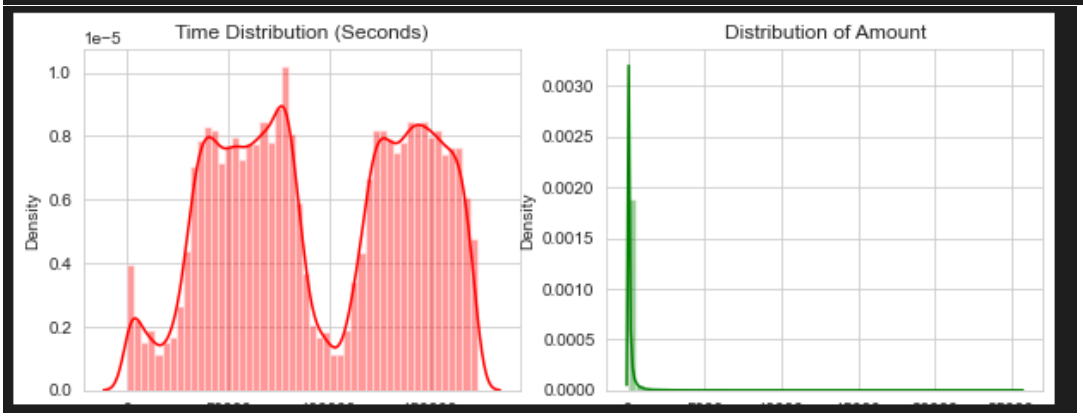
```
#Amount of money used in different transaction classes (valid and fraud)

ps.concat([fraud.Amount.describe() , valid.Amount.describe()] , axis = 1)
```

	Amount	Amount
count	492.000000	284315.000000
mean	122.211321	88.291022
std	256.683288	250.105092
min	0.000000	0.000000
25%	1.000000	5.650000
50%	9.250000	22.000000
75%	105.890000	77.050000
max	2125.870000	25691.160000

```
#Checking time frame of fraudulent transactions

ps.concat([fraud.Time.describe(), valid.Time.describe()], axis=1)
```

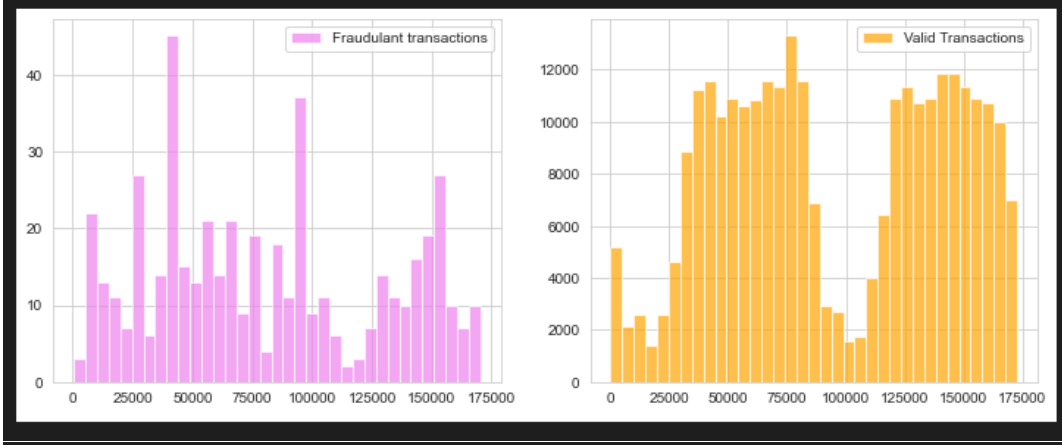
	Time	Time
count	492.000000	284315.000000
mean	80746.806911	94838.202258
std	47835.365138	47484.015786
min	406.000000	0.000000
25%	41241.500000	54230.000000
50%	75568.500000	84711.000000
75%	128483.000000	139333.000000
max	170348.000000	172792.000000

```
#Lets Plot the above execution , i.e., the Time feature / Variable / Atrribute

pylt.figure(figsize=(10,8))

pylt.subplot(2, 2, 1)
pylt.title('Time Distribution (Seconds)')
sbn.distplot(cred['Time'], color='red');


#Lets plot the amount feature

pylt.subplot(2, 2, 2)
pylt.title('Distribution of Amount')
sbn.distplot(cred['Amount'],color='green');
```



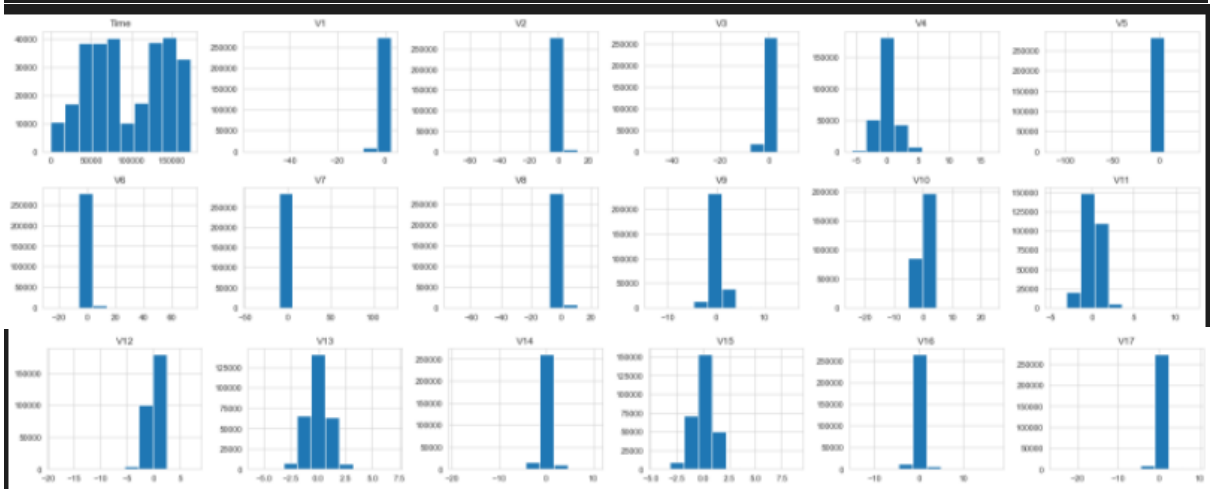
```
# data[data.Class == 0].Time.hist(bins=35, color='blue', alpha=0.6)
pylt.figure(figsize=(12, 10))

pylt.subplot(2, 2, 1)
cred[cred.Class == 1].Time.hist(bins=35, color='violet', alpha=0.7, label="Fraudulant
transactions")

pylt.legend()

pylt.subplot(2, 2, 2)
cred[cred.Class == 0].Time.hist(bins=35, color='orange', alpha=0.7, label="Valid Transactions")

pylt.legend()
```

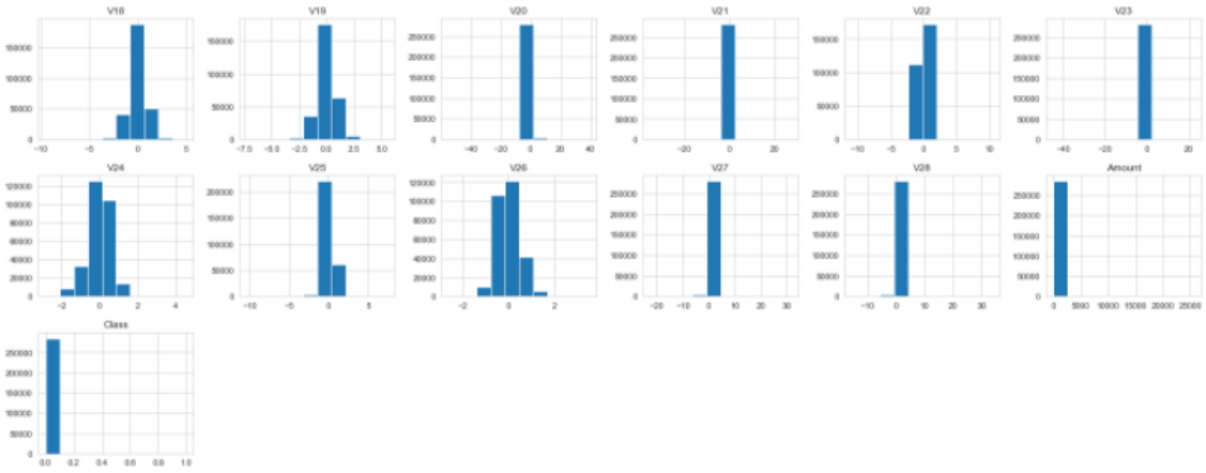
```
<matplotlib.legend.Legend at 0x1b467399a30>
```



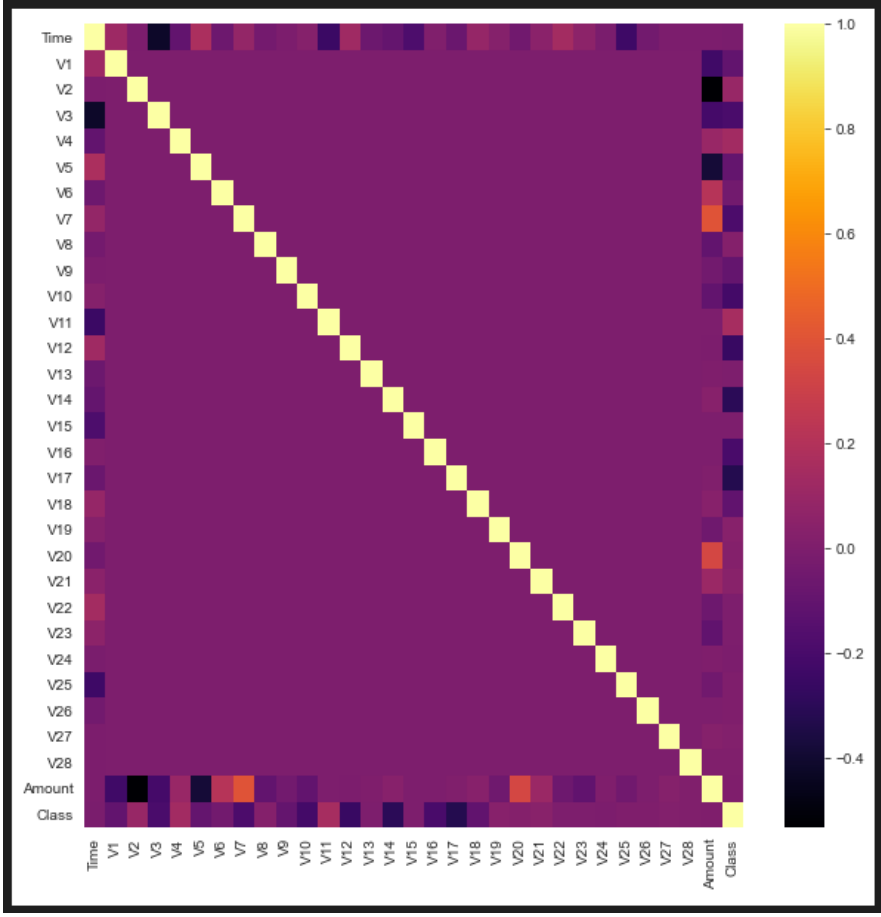

```
#Thus , from the above graphs , it is clear that Time is not at all a factor
#to be analyzed or work upon to reduce the loss.
```

```
#Now let us create sample from the dataset for modelling and testing purpose.

cred.hist(figsize = (25,20))
```






```
#Lets create the heatmap (for correlation among the features)

pylt.figure(figsize=(10,10))
sbn.heatmap(data=cred.corr(), cmap="inferno")
pylt.show()
```



```
#If we check the highest correlations among all the features , the following would be the
#results --

'''
Time and V3 : -0.42
Amount and V2 : -0.53
Amount and V4 : 0.4
'''
```

```
#Lets do some data pre-processing

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

scalar = StandardScaler()

X = cred.drop('Class', axis=1)
Y = cred.Class

X_train_v, X_test, Y_train_v, Y_test = train_test_split(X, Y,
                                                        test_size=0.3, random_state=42)
X_train, X_validate, Y_train, Y_validate = train_test_split(X_train_v, Y_train_v,
                                                            test_size=0.2, random_state=42)

X_train = scalar.fit_transform(X_train)
X_validate = scalar.transform(X_validate)
X_test = scalar.transform(X_test)

w_p = Y_train.value_counts()[0] / len(Y_train)
w_n = Y_train.value_counts()[1] / len(Y_train)

print(f"Fraudulant transaction weight: {w_n}")
print(f"Valid transaction weight: {w_p}")
```

```
Fraudulant transaction weight: 0.0017994745785028623
Valid transaction weight: 0.9982005254214972
```

```
print(f"TRAINING : X_train : {X_train.shape}, Y_train : {Y_train.shape}\n{'_'*55}")
print(f"VALIDATION : X_validate : {X_validate.shape}, y_validate : {Y_validate.shape}\n{'_'*50}
")
print(f"TESTING : X_test : {X_test.shape}, y_test: {Y_test.shape}")
```

```
TRAINING : X_train : (159491, 30), Y_train : (159491,)
_______________________________________________________
VALIDATION : X_validate : (39873, 30), y_validate : (39873,)
__________________________________________________
TESTING : X_test : (85443, 30), y_test: (85443,)
```

```
#Shape is a function which represent a tuple consisting Rows , Columns (in the same order as
#written)
```

```python
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, f1_score

def print_score(label, prediction, train=True):
    if train:
        clf_report = ps.DataFrame(classification_report(label, prediction, output_dict=True))
        print("Train Result:\n================================================")
        print(f"Accuracy Score: {accuracy_score(label, prediction) * 100:.2f}%")
        print("_____________________________________________")
        print(f"Classification Report:\n{clf_report}")
        print("_____________________________________________")
        print(f"Confusion Matrix: \n {confusion_matrix(Y_train, prediction)}\n")

    elif train==False:
        clf_report = ps.DataFrame(classification_report(label, prediction, output_dict=True))
        print("Test Result:\n================================================")
        print(f"Accuracy Score: {accuracy_score(label, prediction) * 100:.2f}%")
        print("_____________________________________________")
        print(f"Classification Report:\n{clf_report}")
        print("_____________________________________________")
        print(f"Confusion Matrix: \n {confusion_matrix(label, prediction)}\n")
```

```python
#NOW COMES THE BIG PART - CONSTRUCTING THE MODEL OF ARTIFICIAL NEURAL NETWORK (ANN)

from tensorflow import keras

model = keras.Sequential([
    keras.layers.Dense(256, activation='relu', input_shape=(X_train.shape[-1],)),
    keras.layers.BatchNormalization(),
    keras.layers.Dropout(0.3),
    keras.layers.Dense(256, activation='relu'),
    keras.layers.BatchNormalization(),
    keras.layers.Dropout(0.3),
    keras.layers.Dense(256, activation='relu'),
    keras.layers.BatchNormalization(),
    keras.layers.Dropout(0.3),
    keras.layers.Dense(1, activation='sigmoid'),
])

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 256)               7936
_________________________________________________________________
batch_normalization (BatchNo (None, 256)               1024
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               65792
_________________________________________________________________
batch_normalization_1 (Batch (None, 256)               1024
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               65792
_________________________________________________________________
batch_normalization_2 (Batch (None, 256)               1024
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 257
_________________________________________________________________
```

```python
#Every Model has some performance and efficiency.
#These can be measured via a number of evaluation metrics.
#Some of the most famous / trending metrics are used below.

METRICS = [
    keras.metrics.Accuracy(name='Accuracy'),
    keras.metrics.FalseNegatives(name='falseNeg'),
    keras.metrics.FalsePositives(name='falsePos'),
    keras.metrics.TrueNegatives(name='TrueNeg'),
    keras.metrics.TruePositives(name='Turepos'),
    keras.metrics.Precision(name='Precision'),
    keras.metrics.Recall(name='Recall') # Recall is also known as Sensitivity
]

model.compile(optimizer=keras.optimizers.Adam(1e-3), loss='binary_crossentropy',
metrics=METRICS)

callbacks = [keras.callbacks.ModelCheckpoint('fraud_model_at_epoch_{epoch}.h5')]
class_weight = {0:w_p, 1:w_n}

r = model.fit(
    X_train, Y_train,
    validation_data=(X_validate, Y_validate),
    batch_size=2048,
    epochs=300,
#     class_weight=class_weight,
    callbacks=callbacks,
)
```

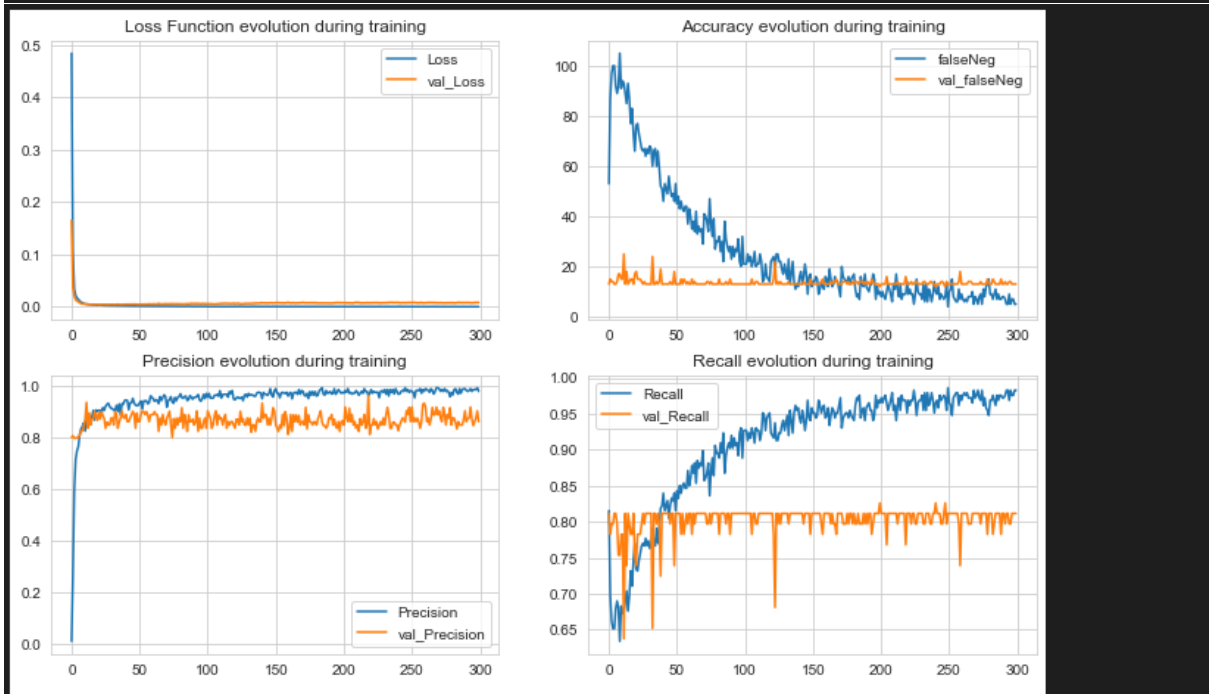
```
78/78 [==============================] - 4s 57ms/step - loss: 1.9183e-04 - Accuracy: 0.0029 - falseNeg:
5.0000 - falsePos: 4.0000 - TrueNeg: 159200.0000 - Turepos: 282.0000 - Precision: 0.9860 - Recall: 0.98
26 - val_loss: 0.0077 - val_Accuracy: 0.0025 - val_falseNeg: 14.0000 - val_falsePos: 10.0000 - val_True
Neg: 39794.0000 - val_Turepos: 55.0000 - val_Precision: 0.8462 - val_Recall: 0.7971
Epoch 297/300
78/78 [==============================] - 4s 57ms/step - loss: 1.9816e-04 - Accuracy: 0.0028 - falseNeg:
7.0000 - falsePos: 4.0000 - TrueNeg: 159200.0000 - Turepos: 280.0000 - Precision: 0.9859 - Recall: 0.97
56 - val_loss: 0.0078 - val_Accuracy: 0.0025 - val_falseNeg: 13.0000 - val_falsePos: 9.0000 - val_TrueN
eg: 39795.0000 - val_Turepos: 56.0000 - val_Precision: 0.8615 - val_Recall: 0.8116
Epoch 298/300
78/78 [==============================] - 5s 59ms/step - loss: 1.8613e-04 - Accuracy: 0.0031 - falseNeg:
7.0000 - falsePos: 4.0000 - TrueNeg: 159200.0000 - Turepos: 280.0000 - Precision: 0.9859 - Recall: 0.97
56 - val_loss: 0.0082 - val_Accuracy: 0.0030 - val_falseNeg: 13.0000 - val_falsePos: 8.0000 - val_TrueN
eg: 39796.0000 - val_Turepos: 56.0000 - val_Precision: 0.8750 - val_Recall: 0.8116
Epoch 299/300
78/78 [==============================] - 5s 59ms/step - loss: 1.3088e-04 - Accuracy: 0.0035 - falseNeg:
5.0000 - falsePos: 2.0000 - TrueNeg: 159202.0000 - Turepos: 282.0000 - Precision: 0.9930 - Recall: 0.98
26 - val_loss: 0.0081 - val_Accuracy: 0.0028 - val_falseNeg: 13.0000 - val_falsePos: 6.0000 - val_TrueN
eg: 39798.0000 - val_Turepos: 56.0000 - val_Precision: 0.9032 - val_Recall: 0.8116
Epoch 300/300
78/78 [==============================] - 5s 59ms/step - loss: 1.7742e-04 - Accuracy: 0.0036 - falseNeg:
5.0000 - falsePos: 6.0000 - TrueNeg: 159198.0000 - Turepos: 282.0000 - Precision: 0.9792 - Recall: 0.98
26 - val_loss: 0.0082 - val_Accuracy: 0.0033 - val_falseNeg: 13.0000 - val_falsePos: 9.0000 - val_TrueN
eg: 39795.0000 - val_Turepos: 56.0000 - val_Precision: 0.8615 - val_Recall: 0.8116
```

```python
score = model.evaluate(X_test, Y_test)
print(score)
```

```
2671/2671 [==============================] - 3s 1ms/step - loss: 0.0052 - Accuracy: 0.0029 - falseNeg: 2
4.0000 - falsePos: 17.0000 - TrueNeg: 85290.0000 - Turepos: 112.0000 - Precision: 0.8682 - Recall: 0.8235
[0.005233634263277054, 0.002949334681034088, 24.0, 17.0, 85290.0, 112.0, 0.8682170510292053, 0.8235294222
831726]
```

```
pylt.figure(figsize=(12, 16))

pylt.subplot(4, 2, 1)
pylt.plot(r.history['loss'], label='Loss')
pylt.plot(r.history['val_loss'], label='val_Loss')
pylt.title('Loss Function evolution during training')
pylt.legend()

pylt.subplot(4, 2, 2)
pylt.plot(r.history['falseNeg'], label='falseNeg')
pylt.plot(r.history['val_falseNeg'], label='val_falseNeg')
pylt.title('Accuracy evolution during training')
pylt.legend()

pylt.subplot(4, 2, 3)
pylt.plot(r.history['Precision'], label='Precision')
pylt.plot(r.history['val_Precision'], label='val_Precision')
pylt.title('Precision evolution during training')
pylt.legend()

pylt.subplot(4, 2, 4)
pylt.plot(r.history['Recall'], label='Recall')
pylt.plot(r.history['val_Recall'], label='val_Recall')
pylt.title('Recall evolution during training')
pylt.legend()
```

```
<matplotlib.legend.Legend at 0x1b488a08f40>
```

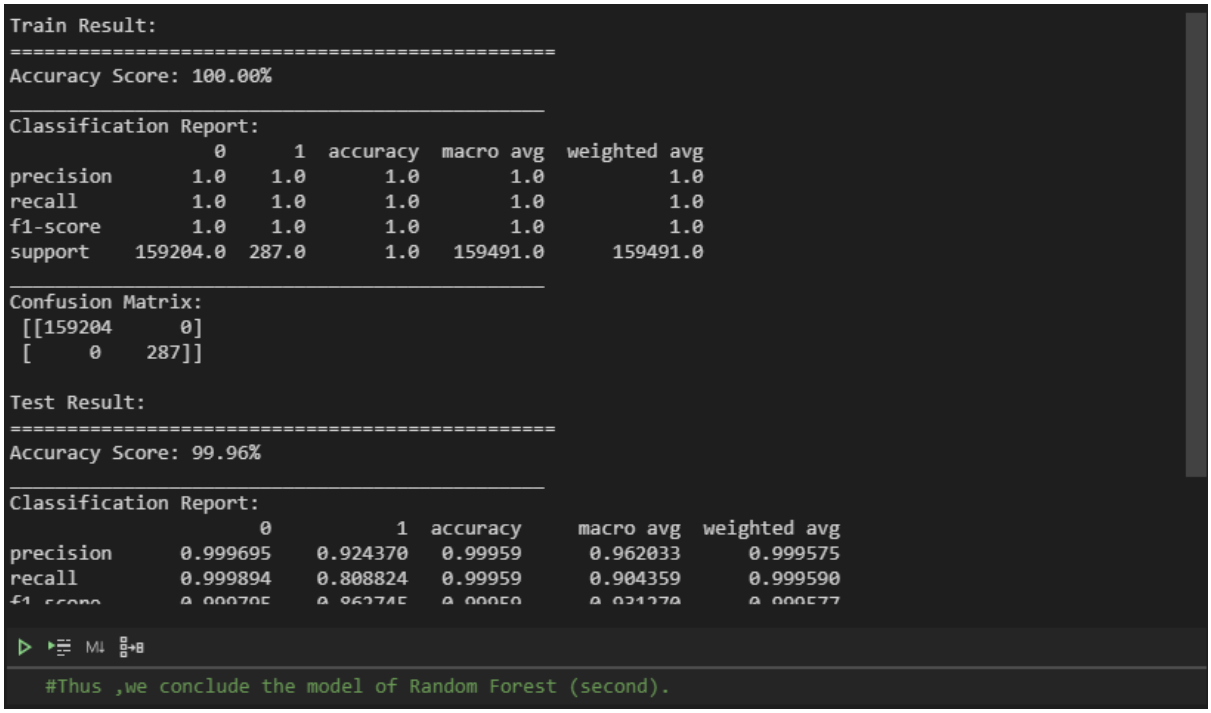


```
Y_train_pred = model.predict(X_train)
Y_test_pred = model.predict(X_test)

print_score(Y_train, Y_train_pred.round(), train=True)
print_score(Y_test, Y_test_pred.round(), train=False)

scores_dict = {
    'ANNs': {
        'Train': f1_score(Y_train, Y_train_pred.round()),
        'Test': f1_score(Y_test, Y_test_pred.round()),
    },
}
```

```
Train Result:
================================================
Accuracy Score: 100.00%

_____________________________________________
Classification Report:
                      0           1  accuracy      macro avg   weighted avg
precision      0.999994    0.993056  0.999981       0.996525       0.999981
recall         0.999987    0.996516  0.999981       0.998252       0.999981
f1-score       0.999991    0.994783  0.999981       0.997387       0.999981
support    159204.000000  287.000000  0.999981  159491.000000  159491.000000
_____________________________________________
Confusion Matrix:
 [[159202      2]
 [     1    286]]

Test Result:
================================================
Accuracy Score: 99.95%

_____________________________________________
Classification Report:
                      0           1  accuracy     macro avg  weighted avg
precision      0.999719    0.868217   0.99952      0.933968      0.999509
recall         0.999801    0.823529   0.99952      0.911665      0.999520
f1-score       0.999760    0.845283   0.99952      0.922521      0.999514
```

```
#Thus , we conclude the Model training and testing of ANN
```

```
#Though we have already created model for Random Forest earlier , still we want to test here
#how much it differs

from sklearn.ensemble import RandomForestClassifier

ranfor_clf = RandomForestClassifier(n_estimators=100, oob_score=False)
ranfor_clf.fit(X_train, Y_train)

Y_train_pred = ranfor_clf.predict(X_train)
Y_test_pred = ranfor_clf.predict(X_test)

print_score(Y_train, Y_train_pred, train=True)
print_score(Y_test, Y_test_pred, train=False)

scores_dict['Random Forest'] = {
        'Train': f1_score(Y_train,Y_train_pred),
        'Test': f1_score(Y_test, Y_test_pred),
}
```



Evaluation Metrics applied to ANN model ---

Accuracy : 99.95%
Precision : 99.97%
Recall : 99.98%
F1 Score : 99.97%

VI. COMPARATIVE STUDY

Now that we have designed , developed and implemented our model , let us compare the same with other existing systems as well (in terms of metrics and modules usage). But before that we shall combine our result obtained via the evaluation metrics in a tabular format.

Technique / Method	Accuracy	Precision	F1 Score	Recall
CNN – Convolutional Neural Network	93.65%	-	-	-
Decision Tree	99.92%	81.00%	78.00%	76.00%
K-NN - (K-Nearest Neighbour)	99.93%	85.42%	81.19%	77.36%
SVM – Support Vector Machine	94.93%	100.00%	92.54%	86.11%
Naïve Bayes Classifier	98.34%	98.48%	87.76%	95.47%
Random Forest	99.90%	95.10%	86.00%	78.60%
ANN – Artificial Neural Network	99.95%	99.97%	99.97%	99.98%

As we have all the evaluation metrics' values here , we shall now proceed for the comparison portion of the different articles with that of ours.

Article(s)	Common method(s) of the others' project with that of ours'	Accuracy of other's project	Accuracy of Our project	Precision of other's project	Precision of Our project	F1 Score of other's project	F1 Score of Our project	Recall of other's project	Recall of Our project
[4] Navanshu Khare , Saad Yunus Sait (2018) , Credit Card Fraud Detection Using Machine Learning Models and Collating Machine Learning Models	Decision Tree	95.50%	99.92%	99.50%	81.00%	-	78.00%	95.50%	76.00%
	Random Forest	98.60%	99.90%	99.70%	95.10%	-	86.00%	98.40%	78.60%
	SVM	97.50%	94.93%	99.60%	100.00%	-	92.54%	97.30%	86.11%
[8] Yashvi Jain , NamrataTiwari , Shripriya Dubey , Sarika Jain (2019) , A Comparative Analysis of Various Credit Card Fraud Detection Techniques	SVM	94.65%	94.93%	85.45%	100.00%	-	92.54%	-	86.11%
	ANN	99.71%	99.95%	99.68%	95.1%	-	99.97%	-	99.98%
	K-NN	97.15%	99.93%	96.84%	85.42%	-	81.19%	-	77.36%
	Decision Tree	97.93%	99.92%	77.80%	81.00%	-	78.00%	-	76.00%
[23] Altyeb Altaher Taha , Sharaf Jameel Malebary (2020) , An Intelligent Approach to Credit Card Fraud Detection Using an Optimized Light Gradient Boosting Machine	Decision Tree	95.50%	99.92%	94.53%	81.00%	27.63%	78.00%	22.62%	76.00%

VII. CONCLUSION AND FUTURE WORK

The recognition of Credit Card Fraud is important to the improved usage of credits cards. With huge and proceeding with monetary misfortunes being capable by monetary firms and given the expanding trouble of distinguishing Visa extortion, it is essential to foster more powerful methodologies for identifying Invalid / fraudulent Credit Card Transactions.

This paper compared various machine learning and deep learning techniques with respect to Evaluation Metrics , which are – Accuracy , Precision , F1 Score , Recall. Among all the discussed models in our project , we conclude that – ANN (Artificial Neural Network) has the highest Accuracy (99.95%) and CNN (Convolutional Neural Network) has the lowest Accuracy (93.65%). In case of Precision , SVM dominates above all (100.00%) while Decision Tree comes at the bottom (81.00%). For Recall (Sensitivity) , ANN dominates (99.98%) , while Decision Tree drops down (76.00%). Last but not the least , for F1 Score , ANN dominates (99.97%) while Decision Tree drops (78.00%).

Overall , the most efficient method among all the mentioned Machine Learning and Deep Learning methods / techniques is - ANN (Artificial Neural Network). Thus , this shows the sheer performance and efficiency towards the training and testing and usage of ANN , hence , preferred the most.

FUTURE WORK / ENHANCEMENTS

Though it's tough to reach a 100% perfect score in terms of accuracy for any method , we still are much close to the goal (only be a margin of 0.1%). But , with this kind of model(s) and numerous datasets , there is always some scope for improvement.
Datasets can play a key role in the score improvement. As demonstrated before, the precision of the algorithms increases when the size of dataset is increased (sometimes may be exceeding the range of 30% - 50% of the present values). Hence, more data will surely make the model more accurate in detecting frauds and reduce the number of false positives.
Combination of numerous modules shall drastically increase the score of Accuracy , but the outputs have to be in a similar format for easy comparison and flawless appease.

VIII. REFERENCES

[1] Vaishnavi Nath Dornadula , S Geetha (2019) , Credit Card Fraud Detection using Machine Learning Algorithms

[2] Massimiliano Zanin , Miguel Romance , Santiago Moral , Regino Criado (2018) , Credit Card Fraud Detection through Parenclitic Network Analysis

[3] Yvan Lucas, Johannes Jurgovsky (2020) , Credit card fraud detection using machine learning: A survey

[4] Navanshu Khare , Saad Yunus Sait (2018) , Credit Card Fraud Detection Using Machine Learning Models and Collating Machine Learning Models

[5] S P Maniraj, Aditya Saini, Shadab Ahmed, Swarna Deep Sarkar (2019) , Credit Card Fraud Detection using Machine Learning and Data Science

[6] Dahee Choi , Kyungho Lee (2018) , An Artificial Intelligence Approach to Financial Fraud Detection under IoT Environment: A Survey and Implementation

[7] Samaneh Sorournejad , Zahra Zojaji , Reza Ebrahimi Atani , Amir Hassan Monadjemi (2016) , A Survey of Credit Card Fraud Detection Techniques: Data and Technique Oriented Perspective

[8] Yashvi Jain , NamrataTiwari , Shripriya Dubey , Sarika Jain (2019) , A Comparative Analysis of Various Credit Card Fraud Detection Techniques

[9] Sonal Mehndiratta , Mr. Kamal Gupta (2019) , Credit Card Fraud Detection Techniques: A Review

[10] Kaithekuzhical Leena Kurien , Dr. Ajeet Chikkamannur (2019) , DETECTION AND PREDICTION OF CREDIT CARD FRAUD TRANSACTIONS USING MACHINE LEARNING

[11] Apapan Pumsirirat , Liu Yan (2018) , Credit Card Fraud Detection using Deep Learning based on Auto-Encoder and Restricted Boltzmann Machine
[12] Daniyal Baig (2020) , Credit Card Fraud Detection Using Supervised Learning Algorithms.

[13] S. Abinayaa , H. Sangeetha, R. A. Karthikeyan , K. Saran Sriram, D. Piyush (2020) , Credit Card Fraud Detection and Prevention using Machine Learning

[14] Nishant Sharma (2019) , CREDIT CARD FRAUD DETECTION PREDICTIVE MODELING

[15] Wael Khalifa , Mohamed Ismail Roushdy , Abdel-Badeeh M. Salem , Hossam Eldin , Hossam Eldin Mohammed Abd El-Hamid (2019) , Machine Learning T Machine Learning Techniques for Credit Card Fraud Detection and Detection

[16] Surbhi Gupta , Mrs. Nitima Malsa , Mr. Vimal Gupta (2017) , Credit Card Fraud Detection & Prevention – A Survey

[17] Shiv Shankar Singh (2019) , Electronic Credit Card Fraud Detection System by Collaboration of Machine Learning Models

[18] Ishu Trivedi , Monika , Mrigya Mridushi (2016) , Credit Card Fraud Detection
[19] Yong Fang , Yunyun , Zhang Cheng Huang (2019) , Credit Card Fraud Detection Based on Machine Learning

[20] Aman Gulat , Prakash Dubey , MdFuzailC , Jasmine Norman , Mangayarkarasi R (2017) , Credit card fraud detection using neural network and geolocation

[21] N.Geetha , T.Kavipriya (2017) , Study on Credit Card Fraud Detection Using Data Mining Techniques

[22] Suraj Patil ,Varsha Nemade , PiyushKumar Son (2018) , Predictive Modelling for Credit Card Fraud Detection Using Data Analytics

[23] Altyeb Altaher Taha , Sharaf Jameel Malebary (2020) , An Intelligent Approach to Credit Card Fraud Detection Using an Optimized Light Gradient Boosting Machine

[24] Niloofar Yousefi , Marie Alaghband,Ivan Garibay (2019) , A Comprehensive Survey on Machine Learning Techniques and User Authentication Approaches for Credit Card Fraud Detection

[25] Shimin LEI , Ke XU , YiZhe HUANG , Xinye SHA (2020) , An XGboost based system for financial fraud detection

[26] Mehak Mahajan , Sandeep Sharma (2019) , Detect Frauds in Credit Card using Data Mining Techniques

[27] Ali Yeşilkanat , Barış Bayram , Bilge Köroğlu,Seçil Arslan (2020) , An Adaptive Approach on Credit Card Fraud Detection Using Transaction Aggregation and Word Embeddings

[28] Alejandro Correa Bahnsen,Djamila Aouada,Aleksandar Stojanovic,Björn Ottersten,(2016),Feature engineering strategies for credit card fraud detection

[29] Yaodong Han , Shun YaoTie Wen , Zhenyu Tian , Changyu Wang , Zheyuan Gu (2020) , Detection and Analysis of Credit Card Application Fraud Using Machine Learning Algorithms

[30] Ronish Shakya (2018) , Application of Machine Learning T Application of Machine Learning Techniques in Credit Card Fraud Detection

==========