Project Report

Credit Card Defaulter Prediction

Project Report submitted on the fulfilment of the requirements of Post graduate Diploma in Big data Analytics

Authors:
Patil Harshal Arjun
Patil Nayan Sunil
Patil Pranoti Satapa
Patil Rushikesh Shrikant
Patil Sanket Shriram

Co-Ordinator:
Prof. Roopa Panicker

STDC

CDAC Thiruvananthapuram

Trivandrum, Kerala 695581

ABSTRACT

Credit card default is a major issue for banks and financial institutions. Accurate prediction of credit card default is crucial for risk management and financial stability. In this project, we propose a machine learning approach for credit card default prediction using Python.

We used the UCI Credit Card Default dataset for training and testing our model. The dataset contains demographic and financial information of credit card holders, along with their payment history and default status. We pre-processed the data by performing feature engineering, handling missing values, and scaling the numerical features.

We experimented with several classification algorithms, including logistic regression, random forest, support vector machine, and neural networks. We evaluated the performance of these algorithms using various performance metrics such as accuracy and F1-score.

Our results indicate that the support vector machine outperformed other algorithms, achieving an accuracy of 82% and an F1-score of 0.80.

Our project demonstrates the feasibility and effectiveness of machine learning for credit card default prediction. The developed model can help financial institutions to identify customers who are likely to default and take necessary measures to prevent or mitigate financial losses.

INTRODUCTION

Credit card default prediction is a crucial task in the banking and finance industry as it helps in identifying customers who are at a higher risk of defaulting on their credit card payments. Predicting credit card default can help financial institutions take proactive measures to mitigate the risk of potential loss. In this project, we will be using machine learning techniques in Python to predict credit card default.

The dataset used in this project contains various features such as the customer's age, gender, education level, marital status, and payment history, among others. We will pre-process the data, visualize and analyse it, and then build a machine learning model to predict credit card default.

The machine learning model we will use is a binary classification algorithm that will classify whether a customer is likely to default on their credit card payment or not. We will compare the performance of different machine learning algorithms such as Logistic Regression, Decision Trees, Random Forests, and Support Vector Machines (SVMs) to select the best performing model.

We will evaluate the performance of the models using metrics such as accuracy, precision, recall, and F1-score.

The end goal of this project is to build a robust machine learning model that can accurately predict credit card default and help financial institutions take proactive measures to mitigate the risk of potential loss.

LITERATURE SURVEY

Prediction of Credit card default requires the use of various machine learning techniques. Some of the work done by various researchers is summarized below:

Real-time Credit Card Fraud/scam Detection Using Machine Learning. [1] This paper centres around four principle fraud events in certifiable transactions. Every fraud is tended to strategy is chosen through an assessment. Significant key territory which we discourse in our venture is constant credit card scam identification.

Ref. [2] proposes model for providing the measures for loss probabilities as well as the evaluation of credit risk. The data used for this purpose consists of account level data from six different banks. Three different machine learning techniques including the decision trees, random forests and logistic regression are evaluated in the proposed model. A credit card amount that is not recovered for a period of more than 90 days is considered as non-recoverable or a default.

E-commerce industry is growing rapidly and this leads to the increased usage of credit card payments for online purchases. In this paper investigation of the performance of logistic regression, random forest and decision tree for credit card fraud detection is carried out. The dataset for credit card fraud detection is gathered from kaggle and this dataset consists of over 2, 84,808 credit card transaction data of a European bank. Fraud transactions are considered as positive class and the genuine transactions are considered as negative class. Dataset consists of imbalanced 0.172% of fraud transitions and the reaming transactions are genuine. Performance is evaluated based on accuracy, sensitivity, error rate and specificity. [3]

[4] In this paper the author has used a case sensitive method which is based on Bays maximum risk and then it is presented using proposed cost measure. The dataset is based on the real-life transaction data obtained from a European company and maintaining the confidentiality of the personal data. The accuracy of the algorithm used is 50%. The main significance of this paper is to reduce the cost. The result obtained was 23%.

Credit Card scam identification- Machine Learning methods [5] Credit Card Scam identification database was utilized in an analysis. Since the database was profoundly non balanced, destroyed strategy was utilized in over sampling. Later on, highlight determination was done and database was part in two sections, preparing data and testing data. The techniques utilized for the investigation were Logistic Regression, Random Forest, and Naive Bayes with Multilayer Perception.

Credit Card Fraud/scam Detection using Deep Learning [6] this paper is tied in with developing a credit card scam identification framework utilizing Deep Learning Neural Networks. Regardless of whether or not the Neural Network is prepared above an extensive variety of emphases, that isn't always effectively accurate to categorize the data as fraud or valid because of skewness of the database. We make use of two sampling systems: Under-Sampling, from lessening number of legitimate perceptions and OverSampling, in which the fraud class perception is copied. Detection of Credit Card Fraud/scam Transactions Using Machine Learning Algorithms and Neural Networks [7] Credit card fraud coming about because of abuse for the framework is characterized like burglary or abuse of someone’s credit card data that is utilized for individual increases unescorted by the consent of the owner

of card. For identifying these scams, this is essential for checking the use examples for a client by the previous transaction. Contrasting the utilization example and present-day transaction, we could categorize this like one or the other scam or a real transaction. In this research, the procedures utilized are KNN, Naïve Bayes, Logistic Regression, Chebyshev Functional Link Artificial Neural Network (CFLANN), Multi-Layer Perceptron and Decision Trees.

This paper checks and investigates the performance of Random Forest, SVM, logistic regression and Decision tree on a highly skewed credit card fraud data. The dataset was gathered by a European cardholder consisting of about 2, 84,786 transactions. The result obtained was 97.7% accuracy by Logistic regression, 97.5% by SVM and 98.6% precise accuracy obtained by Random Forest. [8]

In this paper one of the best data mining algorithms called machine learning algorithm was introduced, which was used to recognize the credit card fraud. A half bread grouping framework with exception recognition was utilized in order to differentiate between misrepresentations of internet recreations. The framework obtained online calculations with factual data in order to distinguish various extraction types. This framework attained extreme location rate at 98% along with 0.1% fault rate. [9]

This paper discusses about supervisor-based classification using Bayesian network classifiers such as Naïve Bayes, K2, Tree Augmented Naïve Bayes (TAN, logistics and J48 classifiers. The datasets are pre-processed by using normalization and principal component analysis. Two datasets were used dummy dataset which represented the characteristics of the credit card data and newly generated dataset using data normalization and principal component analysis technique. All these classifiers achieved over 95% accuracy. [10]

PROBLEM STATEMENT

Predicting potential credit default accounts in advance is challenging. Traditional statistical techniques typically cannot handle large amounts of data and the dynamic nature of fraud and humans. To tackle this problem, recent research has focused on artificial intelligence and machine learning based approaches. In this project, we present and validate a heuristic approach to mine potential default accounts in advance where a risk probability is pre computed from all previous data. This projects focus is to build such model which identifies whether a customer is going to default for credit card payment or not.

Objectives

The objective of our paper is to offer a system or tool to detect 100% of the fraudulent transactions while minimizing the fraud or defaulter's classifications. In this process we filter the records which we got from the financial sector or banking data and on the filter data we train the data and apply to test phase. Here we are using multiple algorithms to find the best out of best accuracy result from them and whichever gives us a best accuracy prediction result we will use them in our system. Our Objective is to determine whether a person defaults the credit card payment for the next month.

PROPOSED SYSTEM

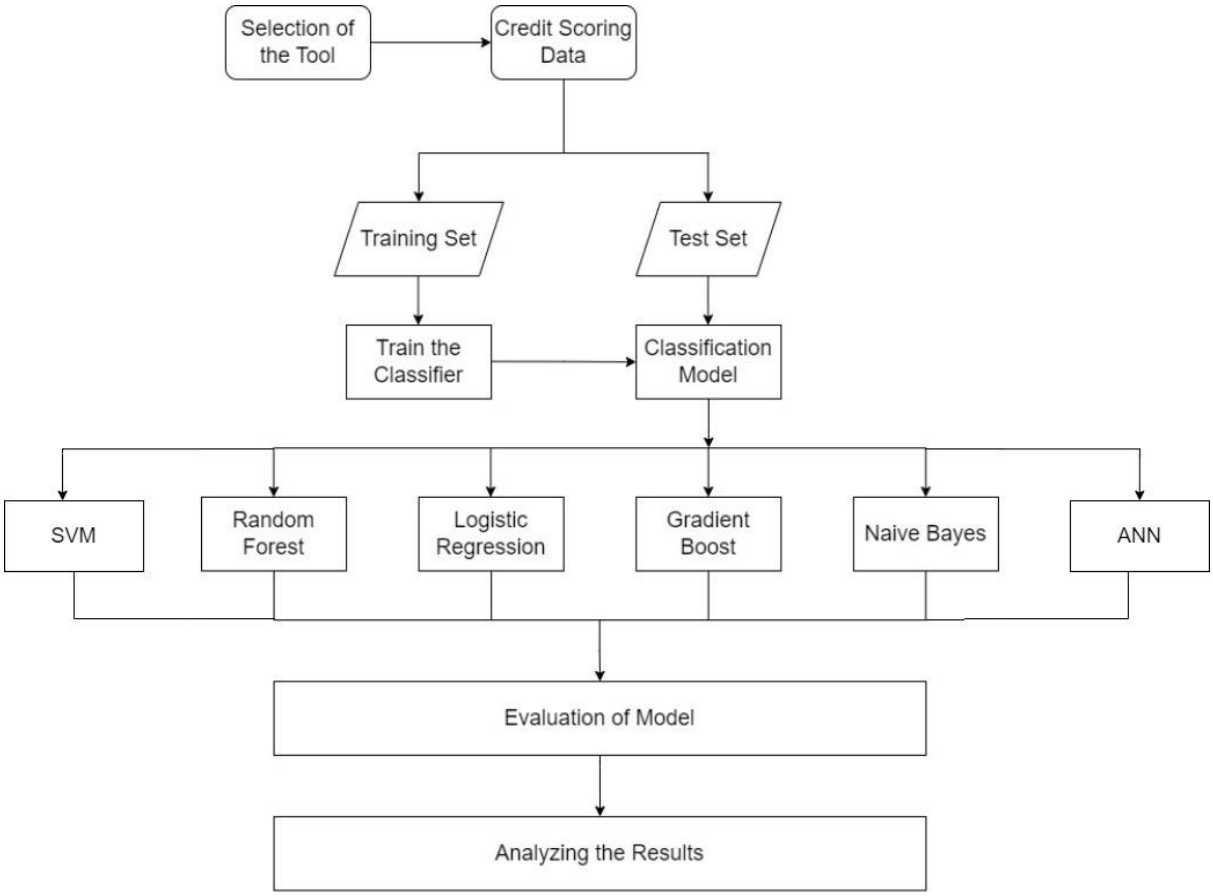


Fig: System Flow Diagram

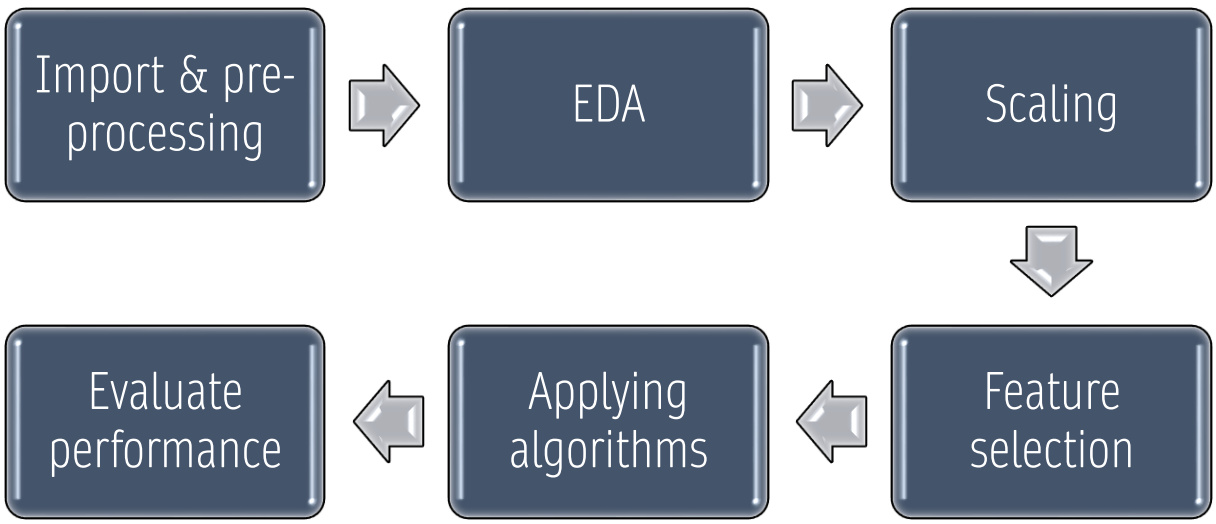


Fig: Flow Diagram

In proposed system, we propose four different algorithms to find the best accuracy in predicting the default candidate for the next month. The credit card data set is divided into two parts- the training set as well as the test set. The classification model is trained using the training set and the remaining observations passes to next level to perform the prediction task using different techniques.

A brief introduction of these techniques is as follows:

Random Forest: Random Forest is a type of supervised learning algorithm getting to know set of rules this is used broadly in Classification and Regression problems. It builds decision trees on distinct samples and takes their majority vote for classification and average in case of regression. In Random Forest model, Random means each tree is only trained on a random subset of samples drawn from the training set (with repetition) and possibly a random subset of features and Forest because there are several trees.

Logistic Regression: Logistic regression model attempts to pick out the correlation between the dependent and the independent variables. It is basically used for binary classification where the target variable is binary and one or more independent variables can be continuous or binary. This model uses the logistic function to identify or to track the probability of the output with respect to the input. The classification is applied such that a threshold is provided, and all the probability values greater than a certain threshold are assigned one class and the values less than the threshold are assigned the other class.

Gradient Boost: The main purpose of Gradient Boost model is to help weak prediction models becomes stronger. It works by building one tree at a time, and correct errors made by previously tree. It can be used for predicting not only continuous target variable (as a Regressor) but also categorical target variable (as a Classifier).

Naive Bayes: Naive Bayes classifier is one of the supervised learning algorithms that is primarily based totally on Bayes theorem and makes use of the probabilistic features. It assumes the independence among all the features for a particular model. Naive Bayes classifier is a simplest method to integrate as it assumes conditional independence. The probabilities of conditional method are identified for the attributes and classification is performed such that the class with the most probable hypothesis or maximum a posteriori (MAP) is assigned to the given element

The K-nearest neighbor (KNN) algorithms are used to detect credit card frauds. This technique is a supervised learning technique. KNN is used for classification of credit card fraud detection by calculating its nearest point. If the new transaction is coming and the point is near the fraudulent transaction, KNN identifies this transaction as a fraud [5]. Many people confuse KNN with K-means clustering, whether they are the same techniques or not. K-means and KNN are different. K-means is an unsupervised learning technique, used for clustering. K-Means tries to determine new patterns from the data and by clustering the data into groups. Conversely, KNN is the number used to compare the nearest neighbor to classify or predict a new transaction based on previous history. The distance in KNN between two data instances can be calculated by using different method, but mostly by using the Euclidean distance. KNN is very useful.

Artificial Neural Networks (ANNs): Artificial neural networks are used to develop relationships between the input and output variables through a learning process. This is done by formulating non-linear mathematical equations to describe these relationships. It can perform a number of classification tasks at once, although commonly each network performs only one. The best solution is usually to train separate networks for each output, then to combine them into an ensemble so that they can be run as a unit. Back propagation algorithm is the best-known example of neural networks algorithm. This algorithm is applied to classify data. In back propagation neural network, the gradient vector of the error surface is computed. This vector points along the line of steepest descent from the current point, so we know that if we move along it a "short" distance, we will decrease the error. A sequence of such moves will eventually find a minimum of some sort. The difficult part is to decide how large the steps should be. Large steps may converge more quickly, but may also overstep the solution or go off in the wrong direction.

Dataset Information

Attribute Name	Attribute Description
Limit_Bal	Amount Of Given Credit
Sex	Gender (1 = Male; 2 = Female)
Education	Education (1 = Graduate School; 2 = University; 3 = High School; 4: Others)
Marriage	Marital Status (1 = Married; 2 = Single; 3 = Others).
Age	Age (Year)
Pay_1 to Pay_6	History of past payment (From April to September 2005): X6 = The repayment status in September 2005… X11 = The repayment status in April, 2005 History of past payment tracked via past monthly payment records (-1 = Payment on time; 1 = Payment delay for one month; 2 = Payment delay for two months; . . . ; 8 = Payment delay for eight months; 9 = Payment delay for nine months and above).
Bill_Amt1 to Bill_Amt6	Amount of bill statement (Dollar) X12 = amount of bill statement in September 2005… X17 = amount of bill statement in April 2005
Pay_Amt1 to Pay_Amt6	Amount of previous payment (Dollar) X18 = Amount paid in September 2005… X23 = Amount paid in April 2005
default payment next month	Default= 1 and healthy= 0

There are 24 Features we are using in this system. We are using 30,000 records from the kaggle.

Dataset pre-processing

1.1 Importing The Required Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

1.2. Importing The Dataset

```
df=pd.read_excel("G:/CDAC/Project/default of credit card clients.xls")
df.head(5)
```

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	...	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT4	PAY_AMT5	PAY_AMT6	DEFAULT_NEXT_MONTH
0	1	20000	2	2	1	24	2	2	-1	-1	...	0	0	0	0	689	0	0	0	0	1
1	2	120000	2	2	2	26	-1	2	0	0	...	3272	3455	3261	0	1000	1000	1000	0	2000	1
2	3	90000	2	2	2	34	0	0	0	0	...	14331	14948	15549	1518	1500	1000	1000	1000	5000	0
3	4	50000	2	2	1	37	0	0	0	0	...	28314	28959	29547	2000	2019	1200	1100	1069	1000	0
4	5	50000	1	2	1	57	-1	0	-1	0	...	20940	19146	19131	2000	36681	10000	9000	689	679	0

5 rows × 25 columns

```
df.columns
```

```
Index(['ID', 'LIMIT_BAL', 'SEX', 'EDUCATION', 'MARRIAGE', 'AGE', 'PAY_0',
       'PAY_2', 'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6', 'BILL_AMT1', 'BILL_AMT2',
       'BILL_AMT3', 'BILL_AMT4', 'BILL_AMT5', 'BILL_AMT6', 'PAY_AMT1',
       'PAY_AMT2', 'PAY_AMT3', 'PAY_AMT4', 'PAY_AMT5', 'PAY_AMT6',
       'DEFAULT_NEXT_MONTH'],
      dtype='object')
```

Fig: Dataframe

Data Manipulation

1.5. Cleaning EDUCATION column

- The variable 'EDUCATION' has three categories with similar information:
- 4: others, 5: unknown, and 6: unknown 0:unknown
- We need to group this three different categories into one single category

```
df["EDUCATION"].unique()
```

```
array([2, 1, 3, 5, 4, 6, 0], dtype=int64)
```

```
df["EDUCATION"]=np.where(df["EDUCATION"]==0, 4, df["EDUCATION"])
df["EDUCATION"]=np.where(df["EDUCATION"]==5, 4, df["EDUCATION"])
df["EDUCATION"]=np.where(df["EDUCATION"]==6, 4, df["EDUCATION"])
```

```
df["EDUCATION"].unique()
```

```
array([2, 1, 3, 4], dtype=int64)
```

1.6. Cleaning MARRIAGE Column

- Similarly, the column 'marriage' should have three categories: 1 = married, 2 = single, 3 = others but it contains a category '0' which will be joined to the category '3'.

```
df["MARRIAGE"].unique()
```

```
array([1, 2, 3, 0], dtype=int64)
```

```
df["MARRIAGE"]=np.where(df["MARRIAGE"]==0, 3, df["MARRIAGE"])
df["MARRIAGE"].unique()
```

```
array([1, 2, 3], dtype=int64)
```

```
print("Number of Rows & Columns:",df.shape)
```

```
Number of Rows & Columns: (30000, 25)
```

Fig: Data Manipulation

```
Total Number of Defaulters: 6636
Total Number of Non-Defaulters: 23364
-----------------------------------------------------------
Total Percent of Defaulters: 22.12 %
Total Percenr of Non-Defaulters: 77.88000000000001 %
-----------------------------------------------------------
```

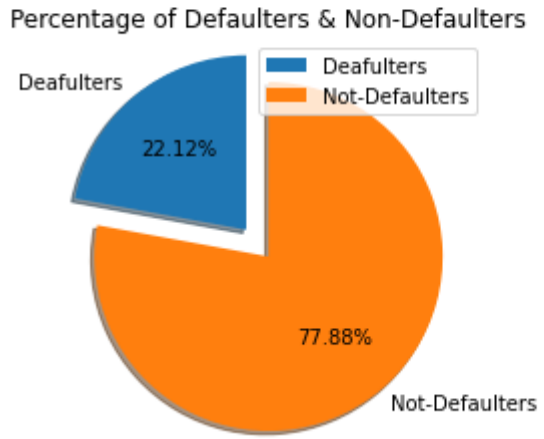


Fig: Distribution of Data

From above figure, we can see that data is highly imbalanced 77.88 percent is credit card amount paid duly 22.12 percent are credit card defaulters.

Correlation coefficient of variables

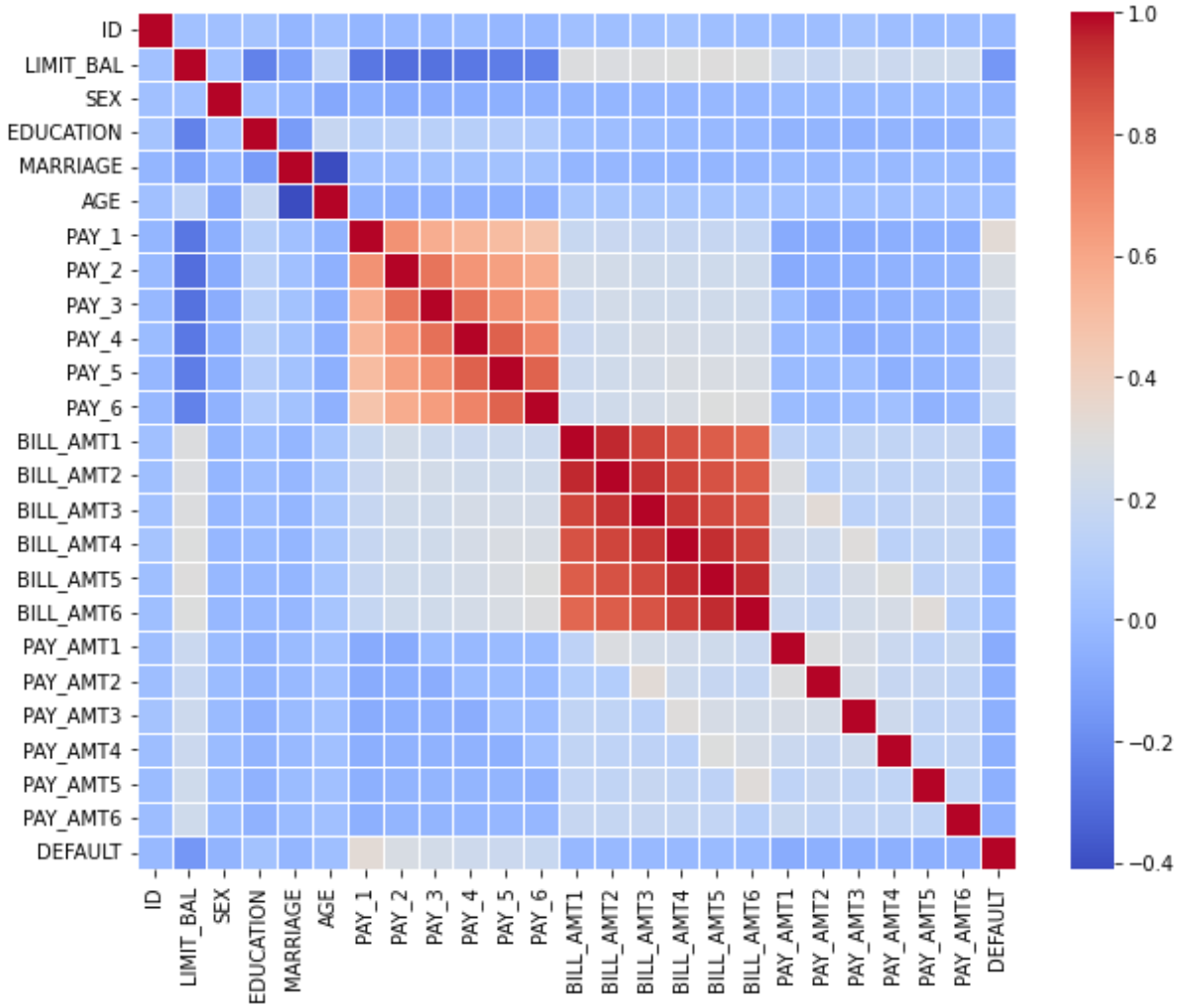
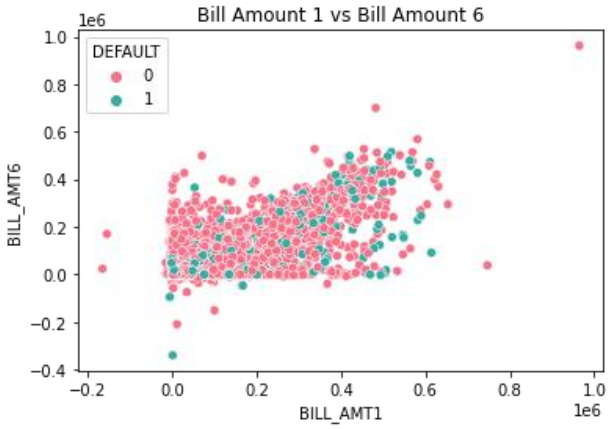
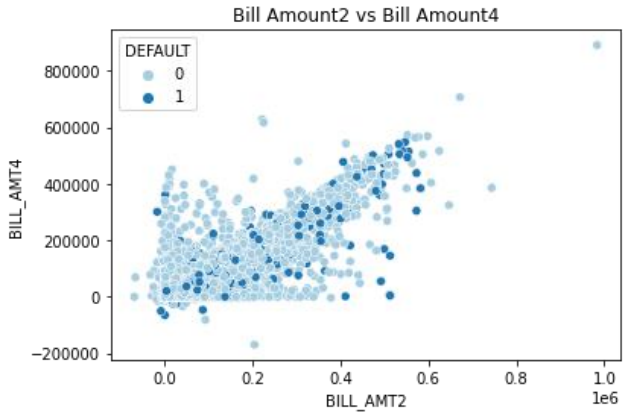


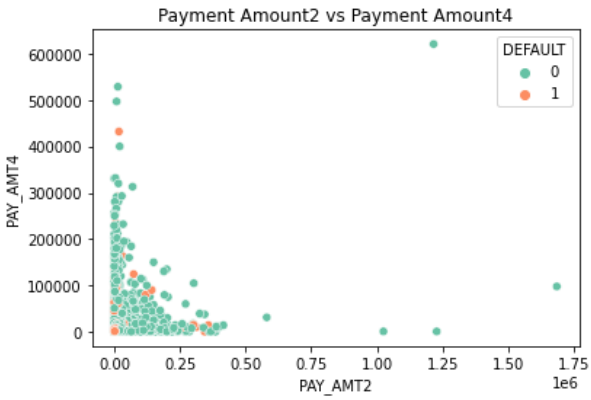
Fig: Showing relationship between two variables

Above figure shows the relationship between two variables; it is generated using heatmap function of machine learning Heatmap is also used in finding the correlation between different sets of attributes.

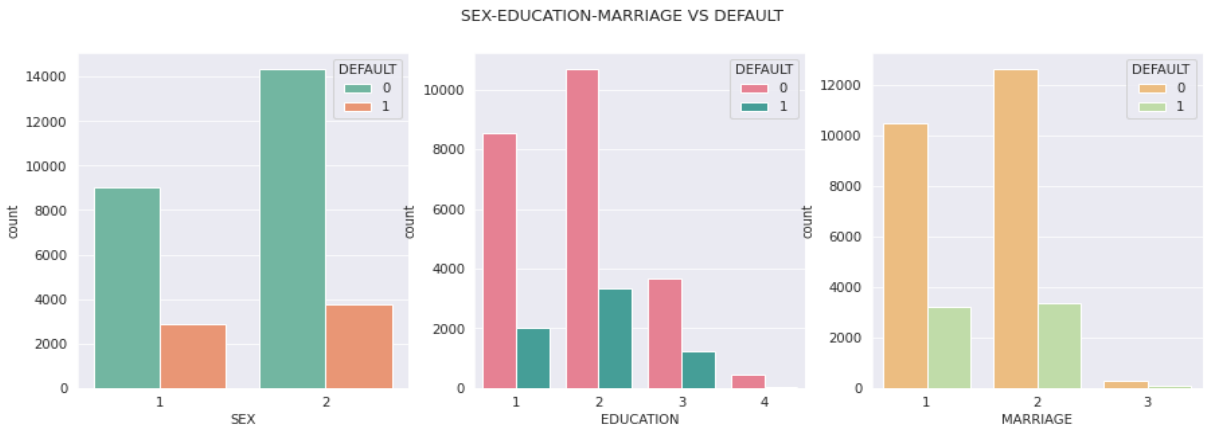
Finding the linear relationship between the independent variables using the scatterplot







Plotting the bar chart to find defaulters based on sex, education, marriage



Scaling the training dataset using StandardScaler

```
from sklearn.preprocessing import StandardScaler
train_scaler=StandardScaler()
test_scaler=StandardScaler()
```

```
scaled_train_data=train_scaler.fit_transform(x_train)
scaled_train_data=pd.DataFrame(data=scaled_train_data,columns=x_train.columns,index=x_train.index)
scaled_train_data
```

Scaling the testing dataset StandardScaler

```
pd.set_option("display.max_columns",30)
scaled_test_data=test_scaler.fit_transform(x_test)
scaled_test_data=pd.DataFrame(data=scaled_test_data,columns=x_test.columns,index=x_test.index)
scaled_test_data
```

Calculating the Mutual Information Of every Feature

```
from sklearn.feature_selection import mutual_info_classif
info_gain1=mutual_info_classif(scaled_train_data,y_train)
info_gain1
```

```
array([0.00432562, 0.01570172, 0.00041765, 0.00361577, 0.00271921,
       0.00344913, 0.07370867, 0.04883756, 0.03581575, 0.03640583,
       0.03385316, 0.03009299, 0.00713363, 0.0054994 , 0.0031593 ,
       0.00426286, 0.01072095, 0.00782432, 0.02237743, 0.01537864,
       0.021196  , 0.00969716, 0.01503073, 0.01146445])
```

Plotting barplot to visualize columnwise mutual information

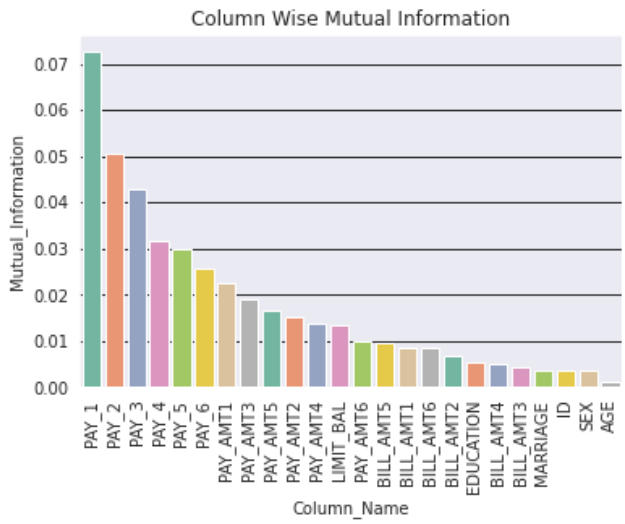


Fig: Column wise mutual information

FEATURE SELECTION USING MUTUAL CLASSIFIER

Selecting the Top-11 Features for the Model Building

```
from sklearn.feature_selection import SelectKBest
eleven_col=SelectKBest(mutual_info_classif,k=11)
eleven_col.fit(scaled_train_data,y_train)
scaled_train_data.columns[eleven_col.get_support()]
```

```
Index(['LIMIT_BAL', 'PAY_1', 'PAY_2', 'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6',
       'PAY_AMT1', 'PAY_AMT2', 'PAY_AMT3', 'PAY_AMT5'],
      dtype='object')
```

Training Dataset

```
scaled_train_fs_data=eleven_col.transform(scaled_train_data)
scaled_train_fs_data=pd.DataFrame(scaled_train_fs_data,columns=scaled_train_data.columns[eleven_col.get_support()],index=scaled_train_data.index)
scaled_train_fs_data
```

	LIMIT_BAL	PAY_1	PAY_2	PAY_3	PAY_4	PAY_5	PAY_6	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT5
15925	-1.063195	-1.756704	-1.554090	-1.526818	-1.523691	-1.532501	-1.487937	-0.334535	-0.245768	-0.301666	-0.316269
10062	0.253364	0.890707	1.756174	1.785922	0.180389	0.227164	0.244044	-0.156089	-0.245520	-0.243881	-0.229656
19376	2.576705	-1.756704	-1.554090	-1.526818	-1.523691	-1.532501	-1.487937	-0.200656	0.783499	2.078955	0.258613
6384	-1.140640	0.008236	0.101042	0.129552	0.180389	0.227164	0.244044	-0.257813	-0.166874	-0.204522	-0.269789
15976	-0.908306	0.890707	1.756174	0.129552	0.180389	0.227164	0.244044	-0.334535	-0.163113	-0.214201	-0.184036
...	...	...	...	...	...	...	...	...	...	...	...
13123	-1.063195	0.008236	0.101042	0.129552	0.180389	0.227164	0.244044	-0.210409	-0.142449	-0.243356	-0.117920
19648	0.330809	0.008236	0.101042	0.129552	0.180389	0.227164	0.244044	0.079218	0.076216	0.106504	0.040760
9845	-0.288748	0.008236	0.101042	0.129552	0.180389	0.227164	1.976026	-0.038997	-0.121785	-0.155891	-0.316269
10799	-0.908306	0.008236	0.101042	0.129552	0.180389	0.227164	0.244044	-0.186766	-0.116619	-0.277351	-0.250152
2732	-0.211304	0.008236	0.101042	0.129552	0.180389	0.227164	0.244044	-0.216320	-0.163113	-0.214201	-0.170813

21000 rows × 11 columns

Testing Dataset

```
scaled_test_fs_data=eleven_col.transform(scaled_test_data)
scaled_test_fs_data=pd.DataFrame(scaled_test_fs_data,columns=scaled_test_data.columns[eleven_col.get_support()],index=scaled_test_data.index)
scaled_test_fs_data
```

	LIMIT_BAL	PAY_1	PAY_2	PAY_3	PAY_4	PAY_5	PAY_6	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT5
8225	-1.127827	0.93875	1.846738	1.868711	1.935378	2.031109	2.031522	-0.361243	-0.179304	-0.287054	-0.309439
10794	-1.127827	0.03077	0.137587	1.868711	0.208561	0.253247	0.274741	-0.144719	-0.291826	-0.233374	-0.245458
9163	0.473002	0.93875	-0.716989	-0.692337	-0.654847	-0.635684	-1.482040	-0.300807	-0.148783	-0.236970	-0.309439
26591	-0.517987	0.03077	0.137587	0.161346	0.208561	0.253247	0.274741	-0.042826	-0.192249	-0.179693	-0.181477
6631	-0.136838	-1.78519	-1.571565	-1.546019	-1.518255	-1.524615	-1.482040	0.083841	-0.250352	0.061225	0.191724
...	...	...	...	...	...	...	...	...	...	...	...
21914	-0.289298	0.03077	0.137587	0.161346	0.208561	0.253247	0.274741	-0.074667	-0.074599	-0.109909	-0.106108
17453	1.159071	-0.87721	1.846738	0.161346	0.208561	0.253247	0.274741	-0.361243	0.106482	-0.018652	0.010466
20344	-0.365528	0.03077	0.137587	0.161346	0.208561	0.253247	0.274741	-0.170193	-0.192249	-0.179693	-0.181477
1878	1.006611	-1.78519	-1.571565	-0.692337	0.208561	0.253247	0.274741	-0.290554	0.315195	-0.270950	-0.049420
6465	1.235301	0.03077	0.137587	0.161346	0.208561	0.253247	0.274741	0.594009	0.206457	0.249750	1.290086

9000 rows × 11 columns

MODELS USED

Once the data is train and test, we have to apply different algorithms to find the accuracy of the prediction for that we use:

1. **Logistic Regression**

   Logistic regression predicts the output of a categorical dependent variable. Therefore, the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the
   exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.

```
import sys
from sklearn.linear_model import LogisticRegression
model_lr=LogisticRegression()
model_lr.fit(scaled_train_fs_data,y_train)
y_pred_logreg=model_lr.predict(scaled_test_fs_data)
```

```
info={"Actual Value":y_test,"Predicted Value":y_pred_logreg}
logreg=pd.DataFrame(info)
logreg.tail(18)
```

2. **SVM (Support Vector Machine)**

   Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection.

```
from sklearn.svm import SVC
model_svm=SVC()
model_svm.fit(scaled_train_fs_data,y_train)
y_pred_svm=model_svm.predict(scaled_test_fs_data)
```

```
info={"Actual Value":y_test,"Predicted Value":y_pred_svm}
svm=pd.DataFrame(info)
svm.sample(18)
```

3. **Naive Bayes**

   Naive Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems. It predicts on the basis of the probability of an object.
   Bayes theorem:

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

Where, P (A | B) is probability of hypothesis

P (B | A) is Probability of the evidence given that the probability of a hypothesis is true.

P (A) is the probability of hypothesis before observing the evidence.

P (B) is the probability of evidence.

```
from sklearn.naive_bayes import GaussianNB
model_navbay=GaussianNB()
model_navbay.fit(scaled_train_fs_data,y_train)
y_pred_navbay=model_navbay.predict(scaled_test_fs_data)
```

```
info={"Actual Value":y_test,"Predicted Value":y_pred_navbay}
NB=pd.DataFrame(info)
NB.sample(18)
```

4. KNN (K-Nearest Neighbours)

The k-nearest neighbors algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. While it can be used for either regression or classification problems, it is typically used as a classification algorithm, working off the assumption that similar points can be found near one another.

```
from sklearn.neighbors import KNeighborsClassifier
model_knn=KNeighborsClassifier()
model_knn.fit(scaled_train_fs_data,y_train)
y_pred_KNN=model_knn.predict(scaled_test_fs_data)
```

```
info={"Actual Value":y_test,"Predicted Value":y_pred_KNN}
KNN=pd.DataFrame(info)
KNN.sample(18)
```

5. Decision Tree

A decision tree is a non-parametric supervised learning algorithm, which is utilized for both classification and regression tasks. It has a hierarchical, tree structure, which consists of a root node, branches, internal nodes and leaf nodes.

```
from sklearn.tree import DecisionTreeClassifier
model_dt=DecisionTreeClassifier()
model_dt.fit(scaled_train_fs_data,y_train)
y_pred_DT=model_dt.predict(scaled_test_fs_data)
```

```
info={"Actual Value":y_test,"Predicted Value":y_pred_DT}
DT=pd.DataFrame(info)
DT.sample(18)
```

6. Random Forest

Random forest is a commonly-used machine learning algorithm trademarked by Leo Breiman and Adele Cutler, which combines the output of multiple decision trees to reach a single result. Its ease of use and flexibility have fueled its adoption, as it handles both classification and regression problems.

```
from sklearn.ensemble import RandomForestClassifier
model_rf=RandomForestClassifier(n_estimators=100)
model_rf.fit(scaled_train_fs_data,y_train)
y_pred_RF=model_rf.predict(scaled_test_fs_data)
```

```
info={"Actual Value":y_test,"Predicted Value":y_pred_RF}
RF=pd.DataFrame(info)
RF.sample(18)
```

7. XG Boost (eXtreme Gradient Boosting)

XGBoost, which stands for Extreme Gradient Boosting, is a scalable, distributed gradient-boosted decision tree (GBDT) machine learning library. It provides parallel tree boosting and is the leading machine learning library for regression, classification, and ranking problems.

It’s vital to an understanding of XGBoost to first grasp the machine learning concepts and algorithms that XGBoost builds upon: supervised machine learning, decision trees, ensemble learning, and gradient boosting.
Supervised machine learning uses algorithms to train a model to find patterns in a dataset with labels and features and then uses the trained model to predict the labels on a new dataset’s features.

```
from xgboost import XGBClassifier
model_xgb=XGBClassifier(n_estimators=500)
model_xgb.fit(scaled_train_fs_data,y_train)
y_pred_XGB=model_xgb.predict(scaled_test_fs_data)
```

```
info={"Actual Value":y_test,"Predicted Value":y_pred_XGB}
XGB=pd.DataFrame(info)
XGB.sample(18)
```

8. ANN (Artificial Neural Network) Using TensorFlow & Keras

Keras is a powerful and easy-to-use free open-source Python library for developing and evaluating deep learning models.
It is part of the TensorFlow library and allows you to define and train neural network models in just a few lines of code.

```
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import Sequential
from tensorflow.keras import layers
from tensorflow.keras.layers import Dense
model_ann=Sequential()
model_ann.add(Dense(12,input_dim=11,activation="sigmoid"))
model_ann.add(Dense(10,activation="relu"))
model_ann.add(Dense(8,activation="relu"))
model_ann.add(Dense(4,activation="sigmoid"))
model_ann.add(Dense(1,activation="sigmoid"))
model_ann.compile(optimizer="Adam",loss="binary_crossentropy",metrics=["accuracy"])
history=model_ann.fit(scaled_train_fs_data,y_train,epochs=100,verbose=1)
```

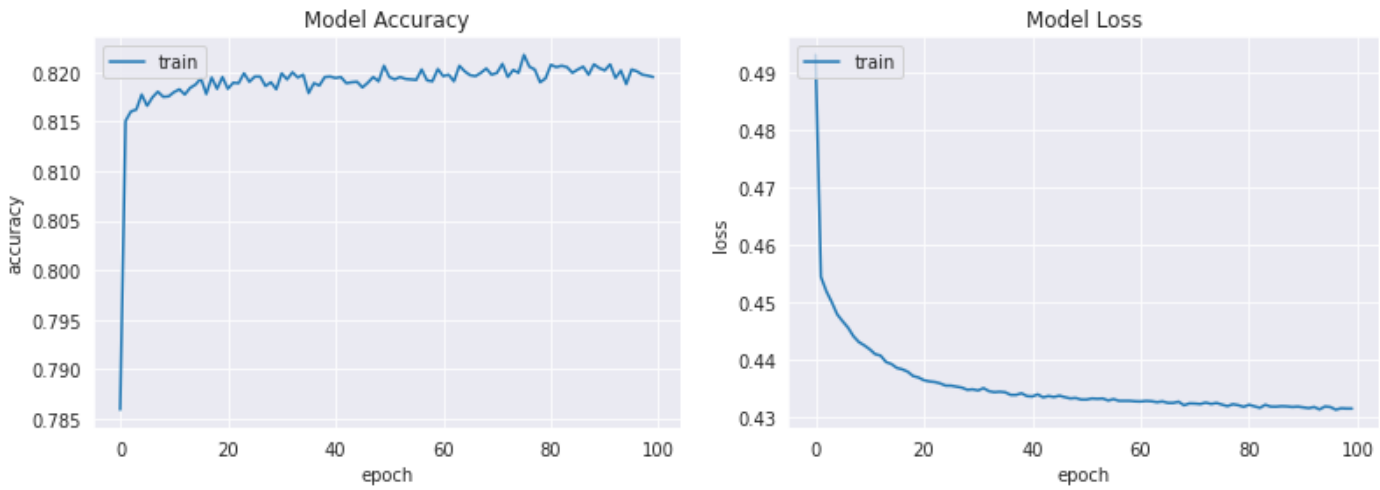


Fig: Model Accuracy and Model Loss using ANN

DISCUSSION AND RESULTS

Index	Accuracy Score	F1_Score
Logistic Regression	0.816444	0.777163
Support Vector Machine	0.824667	0.803047
Naive Bayes	0.761000	0.771294
K-Nearest Neighbour	0.803667	0.787593
Decision Tree	0.716333	0.720207
Random Forest	0.817778	0.794051
XG Boost	0.803667	0.778597
ANN	0.825222	0.825222

Table: Accuracy score and F1 Score of all algorithms

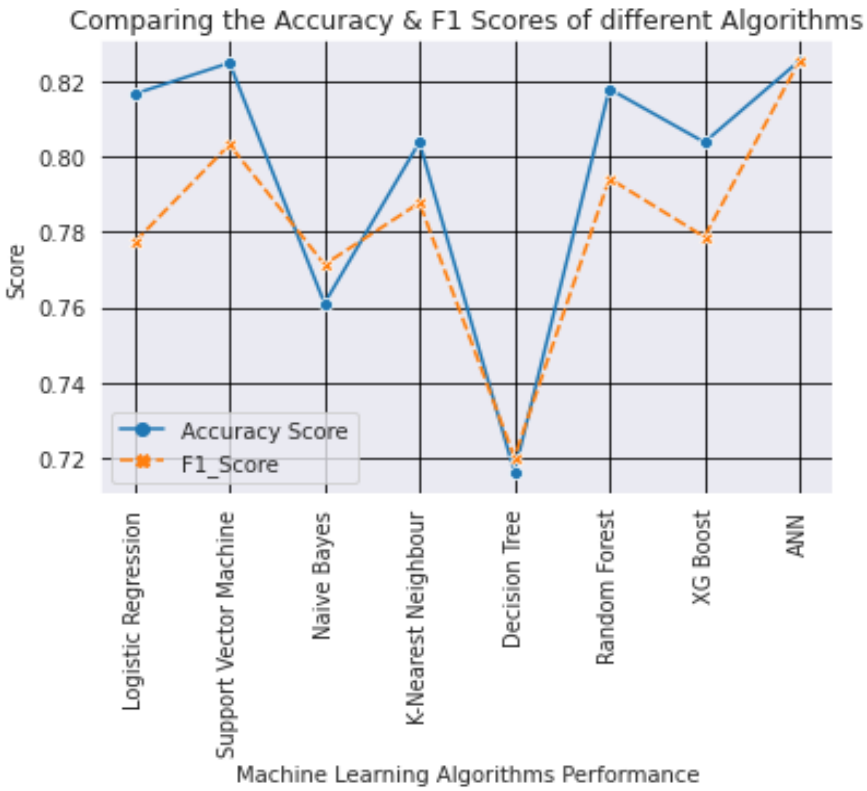


Fig: Comparison of all algorithms using line chart

CONCLUSION

After performing 8 algorithms we can clearly conclude that out of nine algorithms following 2 are the best performing algorithms

1. SVM (Support Vector Machine)

2. ANN (Artificial Neural Network)

REFERENCES

[1] Anuruddha Thennakoon; Chee Bhagyani; Sasitha Premadasa; Shalitha Mihiranga; Nuwan Kuruwitaarachchi 2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence);10-11 Jan. 2019.

[2] Zhou H, Lan Y, Soh Y, Huang G and Zhang R (2012), "Credit risk evaluation with extreme learning machine", IEEE International Conference on Systems, Man, and Cybernetic(SMC), Seoul, 2012, pp. 1064-1069.

[3] M. BM and H. Mohapatra, “Human centric software engineering,” International Journal of Innovations & Advancement in Computer Science (IJIACS), vol. 4, no. 7, pp. 86-95, 2015.

[4] H. Mohapatra, C Programming: Practice, Vols. ISBN: 1726820874, 9781726820875, Kindle, 2018.

[5] Dejan Varmedja; Mirjana Karanovic; Srdjan Sladojevic; Marko Arsenovic; Andras Anderla; 2019 18th International Symposium INFOTEH-JAHORINA (INFOTEH) 20-22 March 2019

[6] Pranali Shenvi; Neel Samant; Shubham Kumar; Vaishali Kulkarni; 2019 IEEE 5th International Conference for Convergence in Technology (I2CT) 29-31 March 2019

[7] Deepti Dighe; Sneha Patil; Shrikant Kokate; 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)16-18 Aug. 2018

[8] H. Mohapatra and A. Rath, Advancing generation Z employability through new forms of learning: quality assurance and recognition of alternative credentials, ResearchGate, 2020.

[9] H. Mohapatra and A. Rath, Fundamentals of software engineering: Designed to provide an insight into the software engineering concepts, BPB, 2020.

[10] V. Ande and H. Mohapatra, “SSO mechanism in distributed environment,” International Journal of Innovations & Advancement in Computer Science, vol. 4, no. 6, pp. 133-136, 2015.