UNIVERSITI TUNKU ABDUL RAHMAN

LEE KONG CHIAN FACULTY OF ENGINEERING AND SCIENCE

UECS3213 DATA MINING

ASSIGNMENT

January 2023 Trimester

NO.	STUDENT NAME	STUDENT ID	PROGRAMME
1.	DING YUAN XING	2106071	AM
2.	LEE KIM LAN	2104745	AM
3.	TAN WAN XUEN	2207214	AM

TABLE OF CONTENTS

1.0 Introduction

Classification, a supervised machine learning technique, has applications such as email spam detection and fraud detection as it categorises data into different classes. Classification involves predictive models like k-Nearest Neighbour (kNN), logistic regression, decision trees, support vector machine (SVM), random forest, naive bayes and neural network. Essentially, it can predict with precision and accuracy whether the data falls into predetermined categories by assigning a probability score, such as spam or not spam, yes or no, among others (Ramakrishnan, 2022). Returning to our question stated: should the bank approve or reject the loan applications of the 20 new applicants? Obviously, the answer will be yes or no, indicating the classification in machine learning.

The datasets related to bank loan approval revealed a substantial dataset, comprising 255,327 observations in the 'BankLoanApproval.csv' file. By relating the 20 new applicants in the 'NewApplicants.csv' file, we are tasked with identifying whether the loan applications should be approved or should be rejected. In order to predict the new outcomes, Python, involving scikit-learn, was utilised as a toolkit to explore, preprocess, and analyse the 'BankLoanApproval.csv' dataset. Moreover, to conduct thorough data mining, the stages of a comprehensive machine learning project are invaluable. It involves preparation, development, and evaluation. In simpler terms, the data mining process begins with data collection and preparation. Next, a model is selected and trained, followed by model evaluation. Finally, hyperparameters are tuned and predictive models are generated (Banoula, 2023).

After predicting all the models involving classification, a confusion matrix is used to compute the performance measures. Generally, confusion matrix is a table visualising and summarising the performance of a classification model by comparing predicted and actual labels (*Confusion Matrix - an Overview | ScienceDirect Topics*, n.d.). By reviewing all the calculated confusion matrices, the best predictive model can be determined.

2.0 Data preparation

Data preparation is one of the crucial steps of analysing a set of data. It involves three stages which are data exploration, data splitting and data preprocessing.

2.1 Data Exploration

A dataset 'BankLoanApproval.csv' is read by importing the pandas library. The first three rows of the dataset are then displayed. With the three rows provided, it was noticed that the data is built up by categorical and numerical data.

```
      LoanID  Age  Income  LoanAmount  CreditScore  MonthsEmployed  \
0  IA35XVH6ZO   28  140466      163781          652              94
1  Y8UETC3LSG   28  149227      139759          375              56
2  RM6QSRHIYP   41   23265       63527          829              87

   NumCreditLines  InterestRate  LoanTerm  DTIRatio    Education  \
0               2          9.08        48      0.23  High School
1               3          5.84        36      0.80          PhD
2               4          9.73        60      0.45     Master's

  EmploymentType MaritalStatus HasMortgage HasDependents LoanPurpose  \
0     Unemployed       Married          No            No   Education
1      Full-time      Divorced          No            No   Education
2      Full-time      Divorced         Yes            No        Auto

  HasCoSigner  Default
0          No        0
1         Yes        1
2         Yes        0
```

Also, to ensure the columns are aligned with rows printed, the column names are checked. The data types are checked too. The table below shows the types of data.

Attribute	**Description**	**Type**
LoanID	ID verification for loan applicants	Categorical
Age	Age of loan applicants	Numerical
Income	Income of loan applicants	Numerical
LoanAmount	Amount loaned by the applicants	Numerical
CreditScore	Loan applicants' credit worthiness	Numerical
MonthsEmployed	Duration has been employed	Numerical
NumCreditLines	Total of credit accounts hold by an individual	Numerical

InterestRate	Annual percentage rate of principal charged by the loan applicant	Numerical
LoanTerm	Duration of borrowing money	Numerical
DTIRatio	Debt-to-Income Ratio	Numerical
Education	Level of education	Categorical
EmploymentType	Type of employment	Categorical
MaritalStatus	Loan applicants' legal relationship	Categorical
HasMortgage	Availability of mortgage on their home	Categorical
HasDependents	Availability of financial support for the individual	Categorical
LoanPurpose	Purpose of borrowing money	Categorical
HasCoSigner	Availability of cosigner	Categorical
Default	Approval or rejection of loan (0 indicates approval while 1 indicates rejection)	Numerical

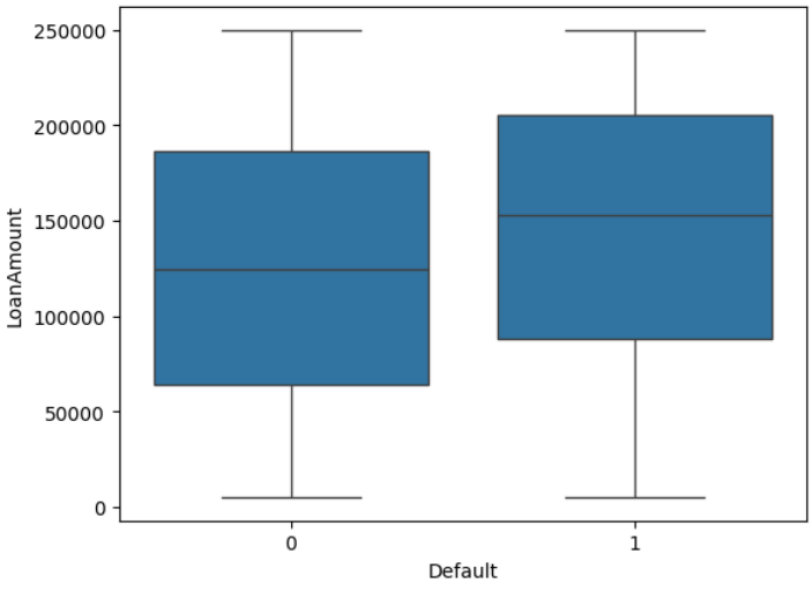
Furthermore, the information of dataset is checked. The data type of each column of data is listed out clearly. It was noticed that there are equally 255327 data for each column.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 255327 entries, 0 to 255326
Data columns (total 18 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   LoanID          255327 non-null  object
 1   Age             255327 non-null  int64
 2   Income          255327 non-null  int64
 3   LoanAmount      255327 non-null  int64
 4   CreditScore     255327 non-null  int64
 5   MonthsEmployed  255327 non-null  int64
 6   NumCreditLines  255327 non-null  int64
 7   InterestRate    255327 non-null  float64
 8   LoanTerm        255327 non-null  int64
 9   DTIRatio        255327 non-null  float64
 10  Education       255327 non-null  object
 11  EmploymentType  255327 non-null  object
 12  MaritalStatus   255327 non-null  object
 13  HasMortgage     255327 non-null  object
 14  HasDependents   255327 non-null  object
 15  LoanPurpose     255327 non-null  object
 16  HasCoSigner     255327 non-null  object
 17  Default         255327 non-null  int64
dtypes: float64(2), int64(8), object(8)
memory usage: 35.1+ MB
```

The descriptive statistic of the dataset is shown. Also, the shape of data, which is (255327,18), is determined.

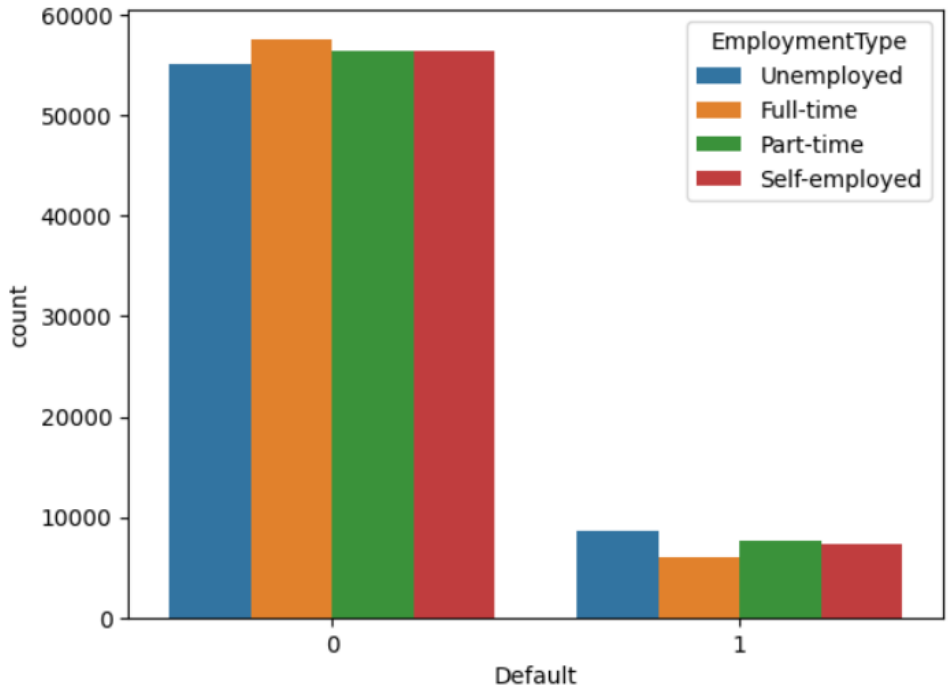
	count	mean	std	min	25%	50%	75%	max
Age	255327.0	43.498059	14.990304	18.0	31.00	43.00	56.00	69.0
Income	255327.0	82500.225585	38963.150663	15000.0	48826.00	82467.00	116219.00	149999.0
LoanAmount	255327.0	127579.236559	70841.308245	5000.0	66156.00	127557.00	188986.50	249999.0
CreditScore	255327.0	574.266125	158.904496	300.0	437.00	574.00	712.00	849.0
MonthsEmployed	255327.0	59.542516	34.643129	0.0	30.00	60.00	90.00	119.0
NumCreditLines	255327.0	2.501036	1.117021	1.0	2.00	2.00	3.00	4.0
InterestRate	255327.0	13.492848	6.636456	2.0	7.77	13.46	19.25	25.0
LoanTerm	255327.0	36.025896	16.969297	12.0	24.00	36.00	48.00	60.0
DTIRatio	255327.0	0.500222	0.230917	0.1	0.30	0.50	0.70	0.9
Default	255327.0	0.116118	0.320367	0.0	0.00	0.00	0.00	1.0

Next, data visualisation is a part of data preparation as it visualises the dataset by different types of graphs. Two graphs are drawn. The first graph is plotted by ‘Default’ versus ‘LoanAmount’. The boxplot shows no outliers. It can be observed that the loan amount successfully approved is lower. Both successful approval and rejection have the same range of amounts. However, the quantiles and median of successful approval is lower than rejection. The phenomenon can be attributed to the high loan amounts extended to applicants, which may not be successfully repaid to the bank, thereby exposing the bank to heightened risk.



.

Moreover, the second graph is a count plot which calculates the number of ‘Default’, where hue is the ‘EmploymentType’. It is noticed that most of the loan applications were approved as the number of people being approved is more than rejected. It was interesting to see that most loan applications being accepted are made up of the employed group, where the largest portion is the full-time employees, and the self-employed and part-time employees share the same portion. While the unemployed loan applicants form the least portion. As the number of loan applicants rejected is significantly lower than those approved, there is a notable portion of the rejected applicants comprises the unemployed, who do not receive any income. Basically, individuals lacking a steady income indicate a higher risk of defaulting on payments. Hence, the bank may exercise greater caution when considering loan applications from unemployed applicants as they may have less favorable credit profiles or insufficient collateral.



Other than that, the exploratory data analysis involves identifying unique values, determining the number of occurrences for each category within a column, and calculating their respective counts. The unique value and numbers of unique value of each category in the columns are identified. This helps us to understand more about the components in the categorical data. Also, the count of unique value is determined.

```
Education: ['High School' 'PhD' "Master's" "Bachelor's"]
Employment Type:  ['Unemployed' 'Full-time' 'Part-time' 'Self-employed']
Marital Status:  ['Married' 'Divorced' 'Single']
Loan Purpose:  ['Education' 'Auto' 'Home' 'Other' 'Business']
```

```
Education: 4
Employment Type:  4
Marital Status:  3
Loan Purpose:  5
```

```
Education: Bachelor's     64360
High School    63895
Master's       63538
PhD            63534
Name: Education, dtype: int64

Employment Type:  Part-time        64156
Unemployed       63819
Self-employed    63702
Full-time        63650
Name: EmploymentType, dtype: int64

Marital Status:  Married     85295
Divorced    85024
Single      85008
Name: MaritalStatus, dtype: int64

Loan Purpose:  Business     51296
Home         51280
Education    51002
Other        50909
Auto         50840
Name: LoanPurpose, dtype: int64
```

2.2 Data Splitting

To have a better model performance, the original dataset is usually split into three smaller sets, namely Test, Validation and Train by the process of data splitting. The main purpose of this process is to prevent overfitting, which arises when the train set fits the model excessively well. Thus, it cannot give accurate prediction during the model evaluation. Based on the above explanation, we would like to split the data before conducting the exploratory data analysis (EDA). And, the EDA will exclusively focus on the training set, thereby reducing the likelihood of model influence by outliers or patterns. This ensures the model's evaluation on unseen data, represented by the test set, which accurately reflects the population.

Generally, Train set is the largest portion of the three datasets. For instance, there will be 70% Train set, 20% Test set and 10% Validation set. The table below shows the purpose of each of the three smaller sets.

Datasets	**Purpose**
Train	• Construct the potential models that appear to make practical sense
Test	• Give accurate model estimations • Provide final measure of the model's performance on new data before deploying the model into production
Validation	• Compare the performance of different models and select the most suitable model for predicting values of the dependent variable • Fine-tuning model parameters to achieve optimal performance • Adjust model complexity and balance it from being overfit or underfit

At the beginning of the data splitting process, the unused column 'LoanID' is deleted to enhance the clarity of the data information. Typically, the output variable is not involved in the model evaluation process. Hence, it should be removed and separated from the input variables to prevent the output variable from being trained. If it was incorporated in the Train set, the model may provide biased or inaccurate predictions on the new data. According to the 'BankLoanApproval.csv' dataset, there are 17 independent variables, and the output variable is named 'Default'. Now, the model is ready to be split and the Train set can learn generalizable patterns and relationships.

Note that there are categorical data which cannot be interpreted directly, the categorical data are handled by transforming them into binary data. All the categorical data are extracted and transformed into new columns while the original categorical data are eliminated. Then, there will be a new dataset which is concatenated with the latest transformed categorical data.

	Age	Income	LoanAmount	CreditScore	MonthsEmployed	NumCreditLines	InterestRate	LoanTerm	DTIRatio	Education_High School	...
0	28	140466	163781	652	94	2	9.08	48	0.23	1	...
1	28	149227	139759	375	56	3	5.84	36	0.80	0	...
2	41	23265	63527	829	87	4	9.73	60	0.45	0	...
3	53	117550	95744	395	112	4	3.58	24	0.73	1	...
4	57	139699	88143	635	112	4	5.63	48	0.20	0	...
...	...	...	...	...	...	...	...	...	...	...	...
255322	67	76558	122456	734	90	2	14.30	60	0.53	0	...
255323	22	45575	43355	545	99	1	21.86	12	0.52	0	...
255324	69	34859	105905	715	25	2	20.28	48	0.35	0	...
255325	41	124446	249800	650	60	4	3.32	24	0.21	1	...
255326	60	20868	81574	584	8	2	5.21	60	0.77	0	...

55327 rows × 24 columns

EmploymentType_Unemployed	MaritalStatus_Married	MaritalStatus_Single	HasMortgage_Yes	HasDependents_Yes	LoanPurpose_Business
1	1	0	0	0	0
0	0	0	0	0	0
0	0	0	1	0	0
1	0	1	0	0	0
0	0	0	0	0	0
...	...	...	...	...	...
0	0	1	0	0	0
1	1	0	1	1	0
0	0	0	0	1	1
1	0	1	1	0	0
0	0	1	0	0	1

LoanPurpose_Education	LoanPurpose_Home	LoanPurpose_Other	HasCoSigner_Yes
1	0	0	0
1	0	0	1
0	0	0	1
0	0	0	1
0	1	0	0
...	...	...	...
0	0	1	1
1	0	0	0
0	0	0	0
0	0	1	0
0	0	0	0

The data types are checked again to ensure it is aligned with the latest data.

```
Age                                int64
Income                             int64
LoanAmount                         int64
CreditScore                        int64
MonthsEmployed                     int64
NumCreditLines                     int64
InterestRate                     float64
LoanTerm                           int64
DTIRatio                         float64
Education_High School              uint8
Education_Master's                 uint8
Education_PhD                      uint8
EmploymentType_Part-time           uint8
EmploymentType_Self-employed       uint8
EmploymentType_Unemployed          uint8
MaritalStatus_Married              uint8
MaritalStatus_Single               uint8
HasMortgage_Yes                    uint8
HasDependents_Yes                  uint8
LoanPurpose_Business               uint8
LoanPurpose_Education              uint8
LoanPurpose_Home                   uint8
LoanPurpose_Other                  uint8
HasCoSigner_Yes                    uint8
dtype: object
```

The data is now split into Train, Test and Validation sets. 70% of data is used for the Train data, 20% of data is used for the Test data and 10% of data is used for the Validation set. The three datasets then undergo normalization by using the 'StandardScaler' to improve their performance and stability. The Train, Test and Validation sets sample size are then printed.

```
Train set size: (178728, 24)
Validation set size: (25277, 24)
Test set size: (51322, 24)
```

2.3 Data Preprocessing

After conducting data exploration and data splitting, various steps of Exploratory Data Analysis (EDA) were completed. This incorporates the examination of data distribution, data visualisation and encoding of categorical variables. However, EDA still involves many more processes like handling missing values, removing duplicates, conducting outlier treatment, and

carrying out bivariate analysis. Thus, all these processes should be done in the data preprocessing stage.

Considering a question, why should EDA be performed? Firstly, it aids in identifying and addressing data quality issues, thereby enabling us to test our assumptions and hypotheses about the data effectively. Additionally, EDA allows us to gain a comprehensive understanding of the data, facilitating clear communication and presentation of findings and insights to others. Moreover, through EDA, we can generate new features, select the most relevant features, and determine the appropriate Machine Learning techniques to apply.

Now, let's continue the data preparation process with data preprocessing. The existence of missing value of X Train set is checked. The Train set is converted to a data frame. From the figure below, there is no missing value.

```
0     0
1     0
2     0
3     0
4     0
5     0
6     0
7     0
8     0
9     0
10    0
11    0
12    0
13    0
14    0
15    0
16    0
17    0
18    0
19    0
20    0
21    0
22    0
23    0
dtype: int64
```

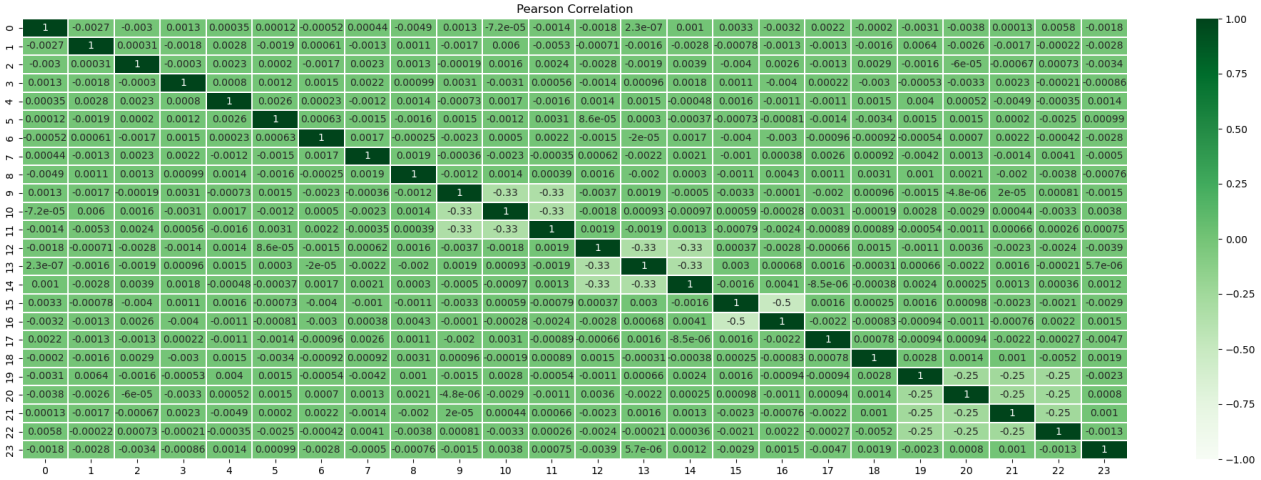
The duplicate entries and rows are also checked. If there were any duplicated rows, then they will be removed. It was realised that there are not any duplicated values or rows.

```
Number of duplicate rows :  (0, 24)
Number of duplicate rows :  (0, 24)
```

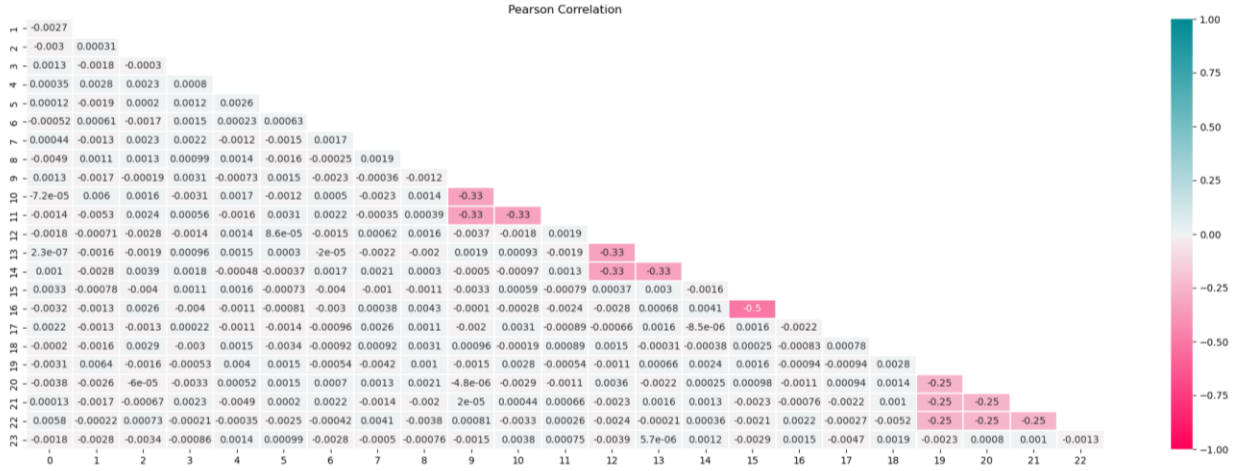
Besides that, to determine the occurrence of outliers, calculation of Z-score is used. The data point is considered as outlier when it is distinct from the other data points as the Z-score is greater than 3. It will be then removed. However, there are no outliers since the data shape shown is same as the initial printed data shape of X Train set.

```
(204261, 24)
```

Correlation between all the variables is determined. A heatmap ranging from –1 to 1 is plotted. It can be observed that most of the values in the heatmap are negative and close to zero. The heatmap shows a negative correlation by representing an inverse relationship between the variables. When one of the variables increases, the other variable decreases, and vice versa.



Furthermore, the masking correlation matrix helps to remove values of correlation repeated twice in the heatmap by revealing only the lower part of the heat map while hiding the upper half. It indicates a negative correlation which is same as the heatmap.



Knowing that the Test set is unseen data, it is used to access the trained model performance on the new data. It is still important to discover and handle the missing values and duplicate rows. However, outliers' detection and handling are not implemented as it was not prioritized in real-world data applications. The existence of missing value of X Test set is checked. The Test set is converted to a data frame. From the figure below, there is no missing value.

```
0     0
1     0
2     0
3     0
4     0
5     0
6     0
7     0
8     0
9     0
10    0
11    0
12    0
13    0
14    0
15    0
16    0
17    0
18    0
19    0
20    0
21    0
22    0
23    0
dtype: int64
```

The duplicate entries and rows are also checked. If there were any duplicated rows, then they will be removed. It was realised that there are not any duplicated values or rows in the X Test set.

```
Number of duplicate rows :  (0, 24)
Number of duplicate rows :  (0, 24)
```

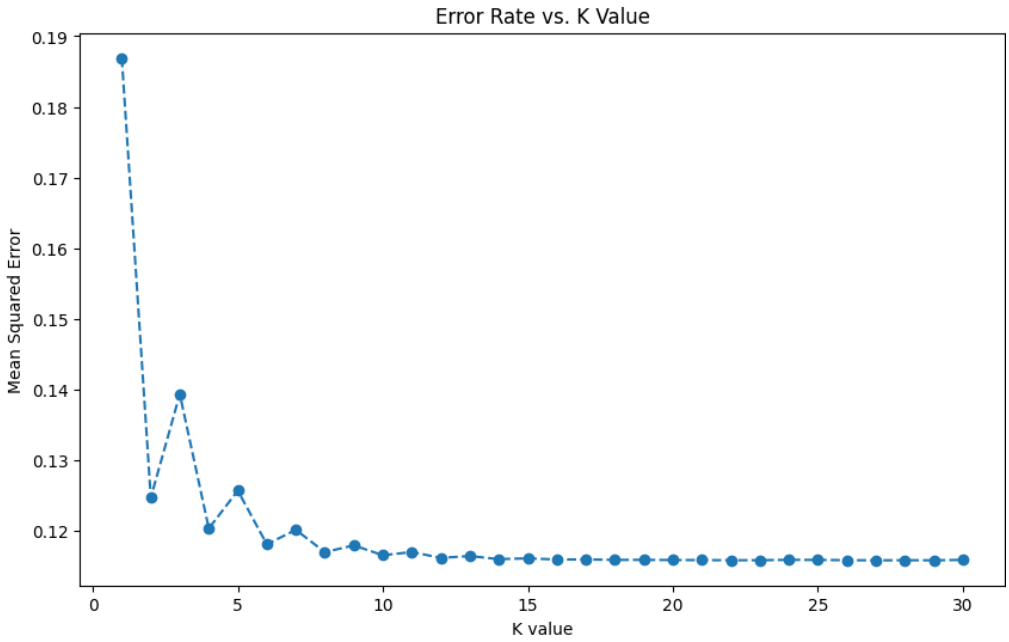
3.0 Creation

Predictive models like k-Nearest Neighbor (kNN), logistic regression, decision trees, support vector machine (SVM), random forest, naive bayes and neural network are used.

3.1.1 k-Nearest Neighbor (kNN)

First, the k-Nearest Neighbor (kNN), under supervised learning, is chosen and trained to fit the dataset, kNN algorithm is simple and efficient at training. It determines the optimal k, number of nearest neighbors and calculates the similarity between the data points by applying the formula of Euclidean distance. The class predicted for the desired data points is the one with the highest frequency among the neighbors (GeeksforGeeks, 2018).

Before the implementation of kNN, cross validation is carried out to find the optimal value of k for the kNN classifier. The purpose of cross validation is to prevent overfitting of the data (Cross-Validation – Amazon Machine Learning, n.d.). Determining the optimal value of k is crucial to strike a balance between sensitivity to noise and outliers and avoiding bias in classification decisions. A small k may result in overfitting due to its sensitivity to noise and outliers. Conversely, a large k might introduce bias by including points from other classes in the neighborhood (Band,2020). In the dataset used, the best value of k is 27. Below diagram shows the graph where error rate versus k value.



Next, the classification report and confusion matrix depicted below provide insight into the model's performance. Upon comparing the classification reports of the Test and Validation sets, it's evident that most values are consistent across both sets. The accuracy of both sets, at 0.88, indicates satisfactory model performance. While precision for class 0 remains consistent between the Test and Validation sets, precision for class 1 is lower in the Test set compared to the Validation set. Both sets exhibit identical recall values for class 0 and 1, indicating a comprehensive representation of both classes. Additionally, the F1-score for class 0 is consistent between the Test and Validation sets, signifying a balanced trade-off between precision and recall. However, the F1-score for class 1 is notably lower in both sets. Overall, these findings suggest that the model performs well for class 0 but struggles with class 1 due to significant class imbalance. Consequently, F-measure will be the best measurement of the confusion matrix.

```
Classification report of test set:

              precision    recall  f1-score   support

           0       0.88      1.00      0.94     45297
           1       0.55      0.00      0.00      6025

    accuracy                           0.88     51322
   macro avg       0.72      0.50      0.47     51322
weighted avg       0.84      0.88      0.83     51322


Classification report of validation set:

              precision    recall  f1-score   support

           0       0.88      1.00      0.94     22361
           1       0.89      0.00      0.01      2916

    accuracy                           0.88     25277
   macro avg       0.89      0.50      0.47     25277
weighted avg       0.89      0.88      0.83     25277


kNN confusion matrix:

[[45288     9]
 [ 6014    11]]
```

Moreover, the ROC curve is then plotted and shown below.

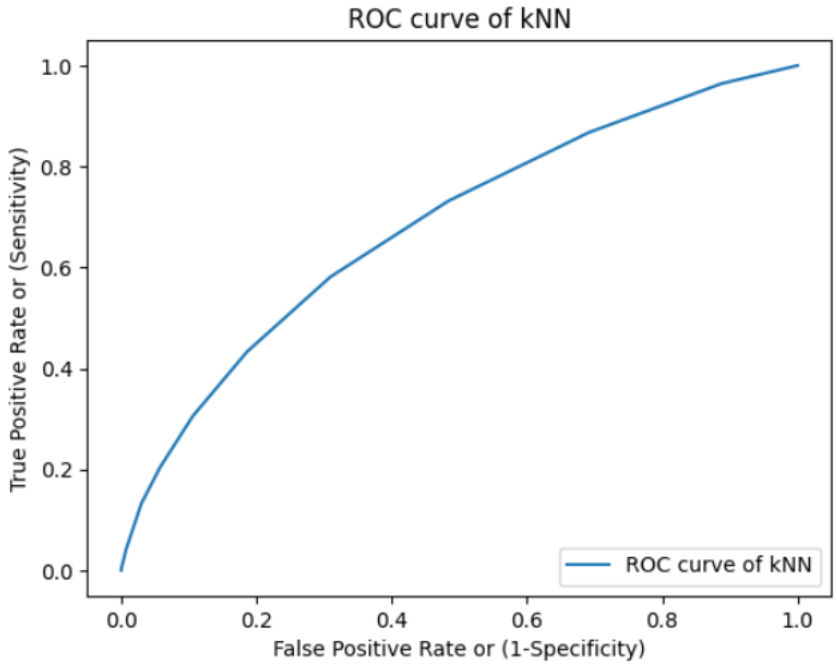


Figure 3.1.1: ROC Curve of kNN

3.1.2 Logistic Regression

Logistic regression is not a regression model, instead of being a classification model that is simple to implement and produces good results with linearly separable classes (Edgar & Manz, 2017). When dealing with binary classification problems where the target variable has two possible outcomes, such as acceptance or rejection, logistic regression can provide a better fit. In comparison to polynomial and linear regression, logistic regression is a simpler and more effective approach for binary and linear classification issues. In addition to being fast and less complicated, logistic regression makes it convenient to interpret the results (Python, n.d.).

```
Model without using GridSearchCV
Classification report of validation set
              precision    recall  f1-score   support

           0       0.89      1.00      0.94     22361
           1       0.61      0.03      0.06      2916

    accuracy                           0.89     25277
   macro avg       0.75      0.52      0.50     25277
weighted avg       0.86      0.89      0.84     25277

Classification report of test set
              precision    recall  f1-score   support

           0       0.89      1.00      0.94     45297
           1       0.61      0.04      0.07      6025

    accuracy                           0.88     51322
   macro avg       0.75      0.52      0.50     51322
weighted avg       0.85      0.88      0.84     51322
```

For both the validation set and test set, the precision is 0.89% for class 0 and the recall is 1.00, indicating that that the model correctly identifies all non-default instances. However, for class 1, the precision is 0.61 and the recall is very low at 0.03 for the validation set and 0.04 for the test set , showing that the model misses a significant portion of actual default instances. The F1-score, the harmonic mean of precision and recall, is relatively low for class 0, of both sets, hence indicating poor performance in correctly identifying default instances. Therefore, accuracy might not provide a complete picture, especially in imbalanced datasets where class 0 dominates.

```
Model with GridSearchCV
Fitting 5 folds for each of 4 candidates, totalling 20 fits
Best parameter after tuning:
  {'C': 1, 'solver': 'liblinear'}
Classification report of validation set
              precision    recall  f1-score   support

           0       0.89      1.00      0.94     22361
           1       0.61      0.03      0.06      2916

    accuracy                           0.89     25277
   macro avg       0.75      0.52      0.50     25277
weighted avg       0.86      0.89      0.84     25277

Classification report of test set
              precision    recall  f1-score   support

           0       0.89      1.00      0.94     45297
           1       0.61      0.04      0.07      6025

    accuracy                           0.88     51322
   macro avg       0.75      0.52      0.50     51322
weighted avg       0.85      0.88      0.84     51322
```

Then, GridSearchCV method is used to optimize logistic regression model by automating the search for optimal hyperparameters (Kumar, 2023). By integrating cross-validation, it enhances model performance by ensuring generalizability to new data. This approach not only saves time but also ensures an objective selection of the best model. GridSearchCV systematically evaluates different parameter combinations and selects those that yield the best performance based on the chosen scoring metric. The grid of parameters defined in this model includes 'C' and 'solver'. 'C', the regularization parameter is a positive floating-point number that determines the strength of regularization to prevent overfitting by penalizing large coefficient values. A smaller value of 'C' indicates stronger regularization. The 'solver' parameter selects the optimization algorithm used to fit the logistic regression model to the training data. The output {'C': 1, 'solver': 'liblinear'} indicates the best parameter values found after tuning the logistic regression model using grid search. In this case, the best value found for 'C' is 1 to balance between bias and variance in the model. The 'liblinear' solver is efficient for small to medium-sized datasets and supports both L1 and L2 regularization penalties.

The classification reports with GridSearchCV are identical to those without GridSearchCV. Therefore, the hyperparameter tuning performed by GridSearchCV did not lead to

any improvement in the model's performance metrics. Hence, the default hyperparameters used in the logistic regression model without tuning may be optimal for the dataset.

```
Logistic regression confusion matrix:

[[45158   139]
 [ 5808   217]]

Logistic regression matrix after using GridSearchCV:

[[45158   139]
 [ 5808   217]]
```

Through the confusion matrix, the model correctly identified 217 instances of default and incorrectly classified 139 instances as default when they were non-default. Additionally, the model failed to identify 5808 instances of default and most instances which is 45158 were correctly classified as non-default. In both confusion matrices, the number of true negatives (TN) and true positives (TP) remains the same before and after using GridSearchCV. This indicates that the model correctly predicted the majority class, non-default instances, and minority class, default instances. Similarly, the number of false positives (FP) and false negatives (FN) remains consistent between the two confusion matrices, suggesting that the model made the same errors in predicting both classes before and after hyperparameter tuning. Overall, the confusion matrices are identical, thus GridSearchCV for hyperparameter tuning did not lead to any change in the model's performance in terms of classifying instances as default or non-default.

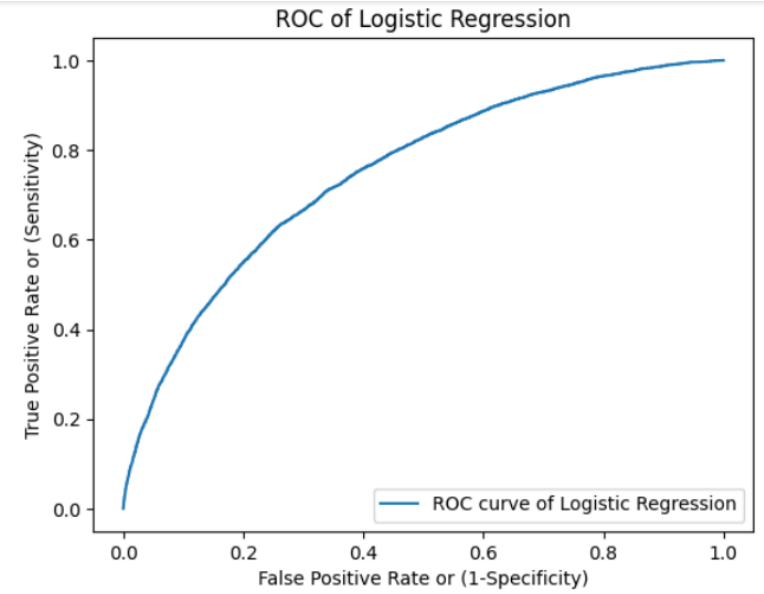


Figure 3.1.2: ROC Curve

3.1.3 Support Vector Machine (SVM)

Apart from that, Support Vector Machine (SVM) model is fitted to the training dataset for classification. SVM model solves complex classification problems by executing the optimal data transformation, which establishes the boundaries between data points using preset outputs, labels or classes (Kanade, 2022). In this case, misclassification issue is not occurred because outliers do not exist in the dataset which was verified during the data preprocessing step.

Since the training dataset's size is very large, using hyperparameter tuning tools such as GridSearchCV and RandomizedSearchCV can take a long time for computation. Instead, four SVM models were built with different kernel functions to compare and select the one that outperformed the others. The kernel functions used in this case are linear, radial basis function (RBF), polynomial and sigmoid kernels.

```
Classification report of validation set
              precision    recall  f1-score   support

           0       0.88      1.00      0.94     22361
           1       1.00      0.00      0.00      2916

    accuracy                           0.88     25277
   macro avg       0.94      0.50      0.47     25277
weighted avg       0.90      0.88      0.83     25277


Classification report of test set
              precision    recall  f1-score   support

           0       0.88      1.00      0.94     45297
           1       1.00      0.00      0.00      6025

    accuracy                           0.88     51322
   macro avg       0.94      0.50      0.47     51322
weighted avg       0.90      0.88      0.83     51322
```

After the models were built, we found that the classification reports for SVM model with linear, RBF and polynomial kernel are the same. Hence, the interpretation of the evaluation metrics for these three models is the same. Figure above shows the classification report. We can see that for both testing and validation dataset, the evaluation metrics such as precision, recall, f1-score and accuracy are similar. From both of them, the precision for class 0 is 0.88, which indicates 88% of class 0 was predicted correctly while the precision for class 1 is 1.00, which suggests that 100% of class 1 was predicted correctly. A recall of 1.00 recommended that 100% of class 0 was recognized correctly while class 1's recall of 0.00 showed that no class 1 was correctly identified. The f1-score of class 0 and class 1 are 94% and 0% respectively and the overall accuracy turned out to be 88%.

```
Confusion matrix of SVM with linear kernel function:

[[45297     0]
 [ 6025     0]]
```

Besides, the confusion matrix of SVM models with linear, RBF and polynomial kernel are also the same. Figure above shows the confusion matrix. It can be seen that there exist a severe imbalanced class case. According to the confusion matrix, a total of 45297 of class 0 was predicted correctly and 0 of class 1 was predicted rightly. The false positive (FP) points out 6025 of class 0

was predicted wrongly instead actually they are class 1. Since the class is imbalance, it is not wise to use accuracy for evaluation.

However, the ROC curve for three of them are slightly different which are shown below.

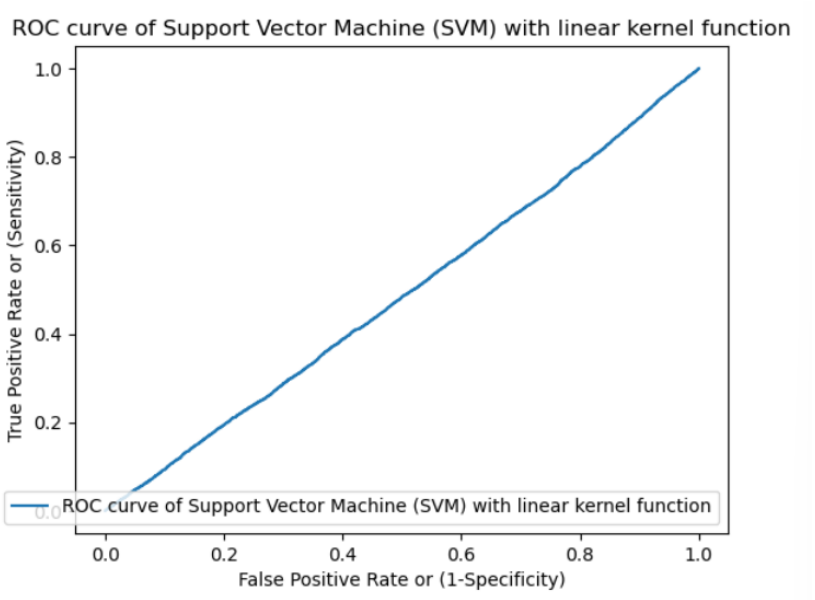


Figure 3.1.3(a): ROC Curve of Support Vector Machine (SVM) with linear kernel function

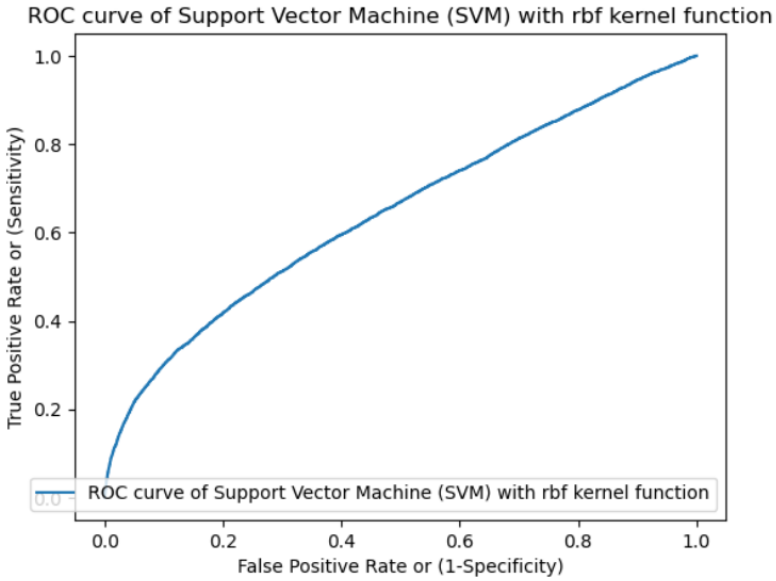


Figure 3.1.3(b): ROC Curve of Support Vector Machine (SVM) with RBF

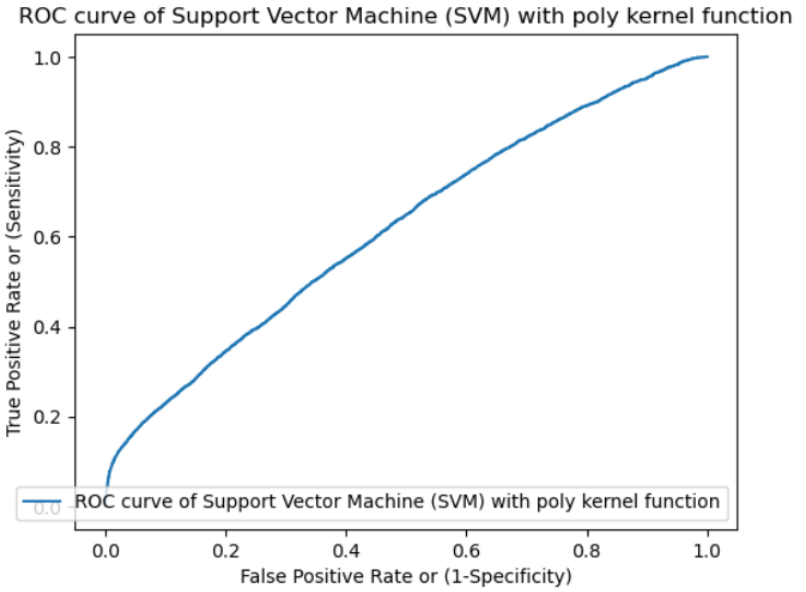


Figure 3.1.3(c): ROC Curve of Support Vector Machine (SVM) with polynomial kernel

Then, we use sigmoid kernel to transform the training dataset and a classification report is shown below.

```
Classification report of validation set
              precision    recall  f1-score   support

           0       0.90      0.90      0.90     22361
           1       0.21      0.20      0.21      2916

    accuracy                           0.82     25277
   macro avg       0.55      0.55      0.55     25277
weighted avg       0.82      0.82      0.82     25277


Classification report of test set
              precision    recall  f1-score   support

           0       0.89      0.90      0.90     45297
           1       0.21      0.20      0.20      6025

    accuracy                           0.82     51322
   macro avg       0.55      0.55      0.55     51322
weighted avg       0.81      0.82      0.81     51322
```

According to the classification report, the evaluation metrics are similar for both testing and validation dataset. The precision of 0.90 means 90% of class 0 was predicted correctly and precision of 0.21 indicates 21% of class 1 was predicted correctly. Next, class 0 possess a recall of 0.90 which designate 90% of class 0 was captured correctly while recall of 0.20 in class 1 states

that 20% of class 1 was captured correctly. The f1-scores are 0.90 and 0.21 for class 0 and class 1 respectively and the overall accuracy accounts 82%.

```
Confusion matrix of SVM with sigmoid kernel function:

[[40690  4607]
 [ 4816  1209]]
```

Figure above shows the confusion matrix of SVM model with sigmoid kernel. It illustrates that 40690 of class 0 were predicted correctly and 1209 of class 1 were predicted accurately. There are 4816 cases predicted to be class 0 but they actually are class 1. In contrast, 4607 cases are predicted to be class 1 but instead they are from class 0. The class is imbalanced which accuracy may not be the best for evaluation measurement.

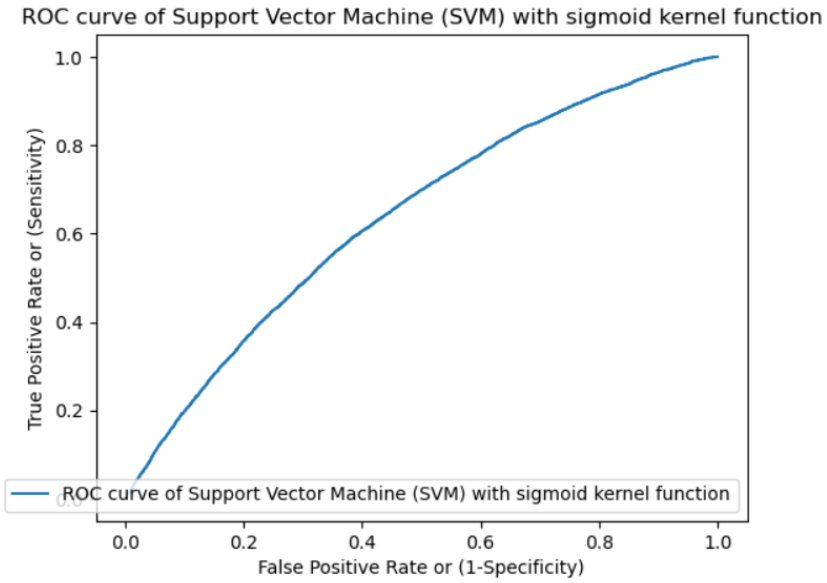


Figure 3.1.3(d): ROC Curve of Support Vector Machine (SVM) with sigmoid kernel

In conclusion, among these four SVM models with different kernel functions, we conclude that SVM model with RBF kernel outperforms the others. This is due to it has the largest area under the ROC curve and it has a higher f1-score compared to sigmoid kernel.

3.1.4 Decision Trees (CART)

Next, we use a tree model to train the dataset. In accordance with GfG (2023), decision trees (CART), said to be one of the most effective supervised learning techniques for classification and regression applications. It constructs a tree structure that resembles a flowchart, with each internal node signifying a test on an attribute, each branch designating a test outcome and each leaf node holds a class label. In this case, a classification trees model is implemented. Figure below shows the classification report without the use of GridSearchCV tool.

```
Model without GridSearchCV

Classification report of validation set
              precision    recall  f1-score   support

           0       0.89      1.00      0.94     22361
           1       0.56      0.02      0.04      2916

    accuracy                           0.89     25277
   macro avg       0.72      0.51      0.49     25277
weighted avg       0.85      0.89      0.84     25277


Classification report of test set
              precision    recall  f1-score   support

           0       0.88      1.00      0.94     45297
           1       0.62      0.02      0.04      6025

    accuracy                           0.88     51322
   macro avg       0.75      0.51      0.49     51322
weighted avg       0.85      0.88      0.83     51322
```

From the report, we can see that the evaluation metrics for both testing and validation dataset are mostly similar except for the precision of class 1. Class 0 has 0.89 precision which indicates 89% of class 0 cases were predicted correctly. For class 1, the precision is 0.56 in validation dataset and 0.62 in testing dataset, which means the accuracy to predict class 1 precisely decreases when we use this trained model to evaluate validation dataset. Furthermore, the recall for class 0 and class 1 are 1.00 and 0.02 respectively. This shows that 100% of class 0 was captured correctly and only 2% of class 1 was captured accurately. F1-score of class 0 is 0.94 while for class 1 is 0.04 which is very low value indicates that it is not perform well for class 1. Finally, the overall accuracy of decision trees model is 88% for testing dataset and 89% for validation dataset.

The hyperparameters are then tuned by using GridSearchCV tool and the classification report is shown below.

```
Model with GridSearchCV

Best parameter after tuning:  {'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 1, 'min_samples_split': 2}

Classification report of validation set
              precision    recall  f1-score   support

           0       0.89      1.00      0.94     22361
           1       0.56      0.02      0.04      2916

    accuracy                           0.89     25277
   macro avg       0.72      0.51      0.49     25277
weighted avg       0.85      0.89      0.84     25277


Classification report of test set
              precision    recall  f1-score   support

           0       0.88      1.00      0.94     45297
           1       0.62      0.02      0.04      6025

    accuracy                           0.88     51322
   macro avg       0.75      0.51      0.49     51322
weighted avg       0.85      0.88      0.83     51322
```

In accordance with the report shown, the best parameters after tuning for the decision tree model suggest that the criterion for splitting nodes is entropy, indicating that the model prioritizes information gain. The maximum depth of the tree is set to 5, implying a moderately complex decision boundary to capture underlying patterns in the data without overfitting. Additionally, the minimum number of samples required to be at a leaf node is 1, suggesting the model is willing to create nodes that represent even small subsets of data. The minimum number of samples required to split an internal node is 2, ensuring that splits are made only when there's sufficient data to support them. Overall, this configuration reflects a balanced approach aimed at maximizing information gain while controlling for overfitting, making it suitable for the given dataset.

Class 0 has excellent precision, recall, and F1-score in both reports, suggesting that the model does a good job of accurately predicting instances of this class. Class 1 has significantly lower precision, recall, and F1-score, indicating that the model has difficulty correctly identifying instances of this class. The poor recall numbers, which show that the model misses a sizable percentage of true positive cases, make this very clear. The low performance metrics for class 1 indicate possible problems with class imbalance and the model's ability to discriminate between the two classes successfully, even with the high overall accuracy. This shows that the model needs to be further refined, possibly using methods to better manage the dataset's imbalance and enhance the model's functionality.

```
Decision Tree confusion matrix:

[[45222    75]
 [ 5902   123]]

Decision Tree confusion matrix after using GridSearchCV:

[[45222    75]
 [ 5902   123]]
```

The model's performance does not seem to have changed following parameter adjustment with GridSearchCV, based on the provided confusion matrices. The counts for true positives, true negatives, false positives, and false negatives are the same in both confusion matrices. This suggests that, at least according to the confusion matrix, the model's capacity to accurately identify occurrences in the validation or test set was not materially altered during the tuning phase. Although this consistency would indicate that the initial parameter values were already close to ideal for the dataset, additional analysis and evaluation of performance metrics are necessary to verify the efficacy of the parameter tuning procedure.

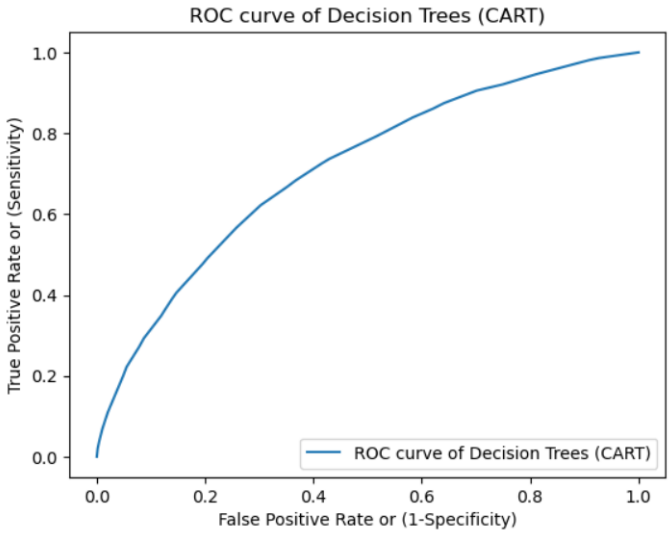


Figure 3.1.4: ROC Curve of Decision Trees (CART)

3.1.5 Random Forest

Moreover, the training dataset is fitted with Random Forest model. Leo Breiman and Adele Cutler created the popular machine learning algorithm Random Forest, which aggregates the output of several decision trees to produce a single outcome. Its versatility and ease of use,

combined with its ability to handle both regression and classification problems, have driven its popularity (R, 2024). The classification is shown below.

```
Model without RandomizedSearchCV

Classification report of validation set
              precision    recall  f1-score   support

           0       0.89      1.00      0.94     22361
           1       0.69      0.03      0.06      2916

    accuracy                           0.89     25277
   macro avg       0.79      0.52      0.50     25277
weighted avg       0.86      0.89      0.84     25277


Classification report of test set
              precision    recall  f1-score   support

           0       0.89      1.00      0.94     45297
           1       0.71      0.03      0.06      6025

    accuracy                           0.88     51322
   macro avg       0.80      0.51      0.50     51322
weighted avg       0.86      0.88      0.83     51322
```

In the absence of RandomizedSearchCV, the model's performance on both the validation and test sets mirrors the previous scenario. The precision for class 0 remains robust, indicating accurate classification of instances belonging to that class. However, the recall and F1-score for class 1 continue to be low, suggesting persistent difficulty in accurately identifying instances of this class. While there is a slight improvement in the precision for class 1, indicating a reduction in false positives, the recall for class 1 remains inadequate, indicating the model's struggle to capture true positives for this class. Despite maintaining high overall accuracy, largely due to the prevalence of class 0, class 1 exhibits poor performance in terms of recall and F1-score. This underscores the model's need for enhancement in correctly classifying instances of class 1, despite its effectiveness in identifying class 0 instances.

```
Model with RandomizedSearchCV
Fitting 3 folds for each of 20 candidates, totalling 60 fits

Best parameter after tuning:  {'bootstrap': True, 'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_sample
s_split': 9, 'n_estimators': 300}

Classification report of validation set
              precision    recall  f1-score   support

           0       0.89      1.00      0.94     22361
           1       0.71      0.03      0.06      2916

    accuracy                           0.89     25277
   macro avg       0.80      0.52      0.50     25277
weighted avg       0.87      0.89      0.84     25277


Classification report of test set
              precision    recall  f1-score   support

           0       0.89      1.00      0.94     45297
           1       0.73      0.03      0.06      6025

    accuracy                           0.88     51322
   macro avg       0.81      0.51      0.50     51322
weighted avg       0.87      0.88      0.84     51322
```

These parameters suggest that the model should utilize bootstrapping for sampling, not set a maximum depth for decision tree splits, use square root of the number of features for the maximum number of features considered at each split, have a minimum number of samples required to be at a leaf node set to 1, a minimum number of samples required to split an internal node set to 9, and employ 300 estimators in the random forest ensemble. These parameters were determined to be the most suitable for the given dataset during the tuning process.

The model's performance on both the validation and test sets is in close agreement with prior findings after being tuned using RandomizedSearchCV. Class 0 precision is still high, meaning that occurrences of that class are accurately classified. Recall and the F1-score for class 1 remain poor, nevertheless, suggesting that it will remain challenging to correctly identify examples of this class. Although the precision for class 1 shows a little improvement, indicating a decrease in false positives, the recall for class 1 is still insufficient, indicating that the model has difficulty collecting real positives for this class. Even with class 0's dominance, which is the main reason for the high overall accuracy, class 1 performs poorly in terms of recall and F1-score.

```
Random Forest confusion matrix:

[[45224    73]
 [ 5850   175]]

Random Forest confusion matrix after using RandomizedSearchCV:

[[45233    64]
 [ 5849   176]]
```

Comparing the two confusion matrices, we notice minor differences in the Random Forest model's performance before and after employing RandomizedSearchCV for parameter optimization. In both instances, the model effectively distinguishes instances of class 0, with a high number of true negatives (TN) and a low count of false negatives (FN). However, regarding class 1, there are discernible alterations. Post parameter tuning, there's a slight uptick in true positives (TP) and a reduction in false positives (FP), indicating a slight improvement in correctly identifying class 1 instances while decreasing misclassifications of class 0 as class 1. Overall, these adjustments imply that parameter tuning with RandomizedSearchCV has marginally boosted the model's ability to accurately classify class 1 instances.

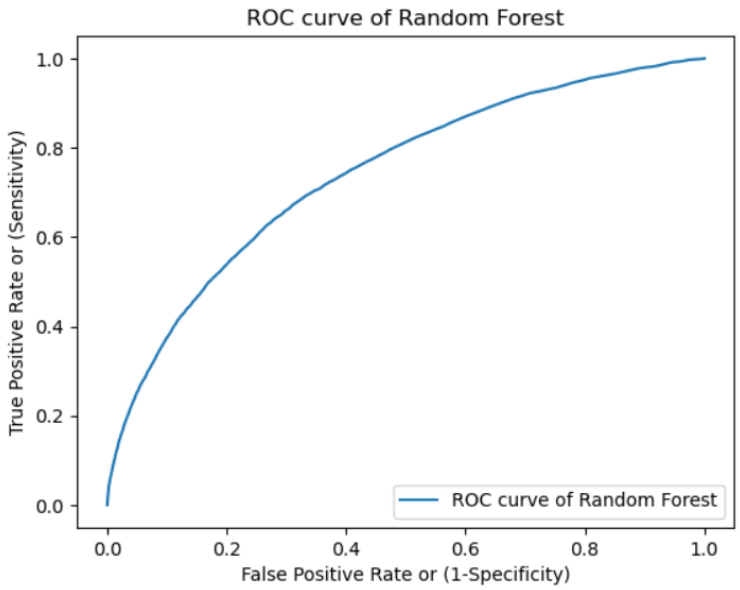


Figure 3.1.5: ROC Curve of Random Forest

3.1.6 Naive Bayes

On the other hand, Naive Bayes is selected and used to train the dataset. It was interesting to note that there is a story behind the name “Naive Bayes”. The “Naive” part of the name arises from the assumption that the occurrence of one feature is independent of others. Meanwhile, “Bayes” is a reference to Reverend Thomas Bayes, the creator of Bayes’ Theorem (Kumar, 2019). Normally, Naive Bayes can mostly be found in text classification, sentiment analysis and

recommendation system. There are 4 types of Naive Bayes Classifiers which are Optimal Naive Bayes, Gaussian Naive Bayes, Multinomial Naive Bayes and Bernoulli Naive Bayes. However, in this assignment, we are using the Gaussian Naive Bayes to train the dataset. Figure below shows the classification report and confusion matrix generated by Naive Bayes.

```
Classification report of validation set
              precision    recall  f1-score   support

           0       0.89      1.00      0.94     22361
           1       0.62      0.03      0.05      2916

    accuracy                           0.89     25277
   macro avg       0.75      0.51      0.49     25277
weighted avg       0.86      0.89      0.84     25277

Classification report of test set
              precision    recall  f1-score   support

           0       0.89      1.00      0.94     45297
           1       0.63      0.03      0.06      6025

    accuracy                           0.88     51322
   macro avg       0.76      0.51      0.50     51322
weighted avg       0.86      0.88      0.83     51322


Naive Bayes confusion matrix:

[[45195   102]
 [ 5851   174]]
```

From the classification report of Test set and Validation set, most of the values in both reports are mostly the same. The accuracy of Validation set, and Test set are 0.89 and 0.88 respectively, denoting the model is performing well. Precision for class 0 in Test set and Validation set are the same while precision for class 1 in Test set is slightly higher with 0.01 than precision for class 1 in Validation set. Both Test and Validation set share the same recall for both class 0 and 1, which are 1.00 and 0.03. They also share the same F1-score in class 0 which is 0.94 representing a well balance between precision and recall. However, the F1-score in class 1 for both sets is significantly low. Based on all these observations, class 0 has a well model performance.

Furthermore, from the confusion matrix, the classifier is facing class imbalance problem. F-measure will be the best measurement of the confusion matrix.

In Naive Bayes, it was found that hyperparameter tuning does not improve the classification performance due to the limited number of hyperparameters and the simplicity of Naive Bayes classifiers. The number of hyperparameters to be tuned is based on the number of classes. Since there are only two classes in the respective dataset, there are too few parameters to tune. Hence, it can conclude that hyperparameter tuning is not required and is an invalid method for improving Naive Bayes model (Tokuç, 2021).

Moreover, the ROC curve is then plotted and shown below.

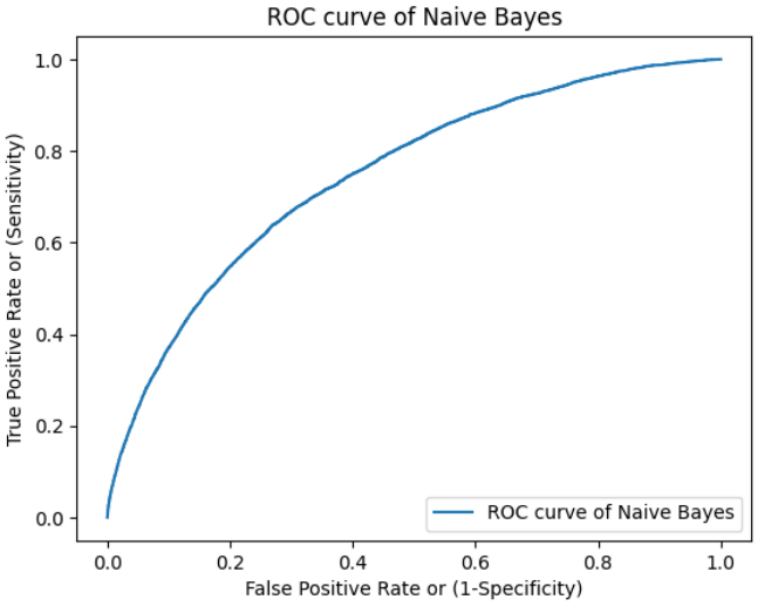


Figure 3.1.6: ROC Curve of Naive Bayes

3.1.7 Neural Network

A neural network is a machine learning program that uses layers of interconnected nodes to build models based on training data and predict unknown data (IBM, 2023). A neural network is a type of deep learning technique that leverages neurons in a layered structure similar to the human brain to make decisions. It can learn. Neural networks utilize training data to learn and model complex connections between input and output data. Once fine-tuned for accuracy over time, it is useful model for efficiently classifying and clustering data (AWS, 2023).

```
Model without RandomizedSearch CV
Classification report of validation set
              precision    recall  f1-score   support

           0       0.89      0.99      0.94     22361
           1       0.61      0.08      0.14      2916

    accuracy                           0.89     25277
   macro avg       0.75      0.53      0.54     25277
weighted avg       0.86      0.89      0.85     25277

Classification report of test set
              precision    recall  f1-score   support

           0       0.89      0.99      0.94     45297
           1       0.58      0.07      0.13      6025

    accuracy                           0.89     51322
   macro avg       0.73      0.53      0.54     51322
weighted avg       0.85      0.89      0.84     51322
```

Based on the classification reports for the validation set and test set without hyperparameter tuning, the model achieves high precision and recall, resulting in a high F1-score of 94%. Consequently, the model accurately identifies most class 0 instances while minimizing false positives. However, the model struggles to accurately classify instances of class 1. This is evident from the low F1-score for class 1, indicating that the model fails to capture a significant portion of class 1 instances. Moreover, the relatively low precision suggests that the model incorrectly labels some instances as class 1, while the low recall indicates that the model fails to capture a significant portion of class 1 instances.

```
Best score:  0.8861398300332581
Best parameter after tuning:  {'max_iter': 50, 'hidden_layer_sizes': 91, 'alpha': 0.01, 'activation': 'relu'}
```

```
Classification report of validation set
              precision    recall  f1-score   support

           0       0.89      0.99      0.94     22361
           1       0.60      0.09      0.16      2916

    accuracy                           0.89     25277
   macro avg       0.75      0.54      0.55     25277
weighted avg       0.86      0.89      0.85     25277

Classification report of test set
              precision    recall  f1-score   support

           0       0.89      0.99      0.94     45297
           1       0.57      0.09      0.16      6025

    accuracy                           0.89     51322
   macro avg       0.73      0.54      0.55     51322
weighted avg       0.85      0.89      0.85     51322
```

Through our research, we have identified Grid Search and Randomized Search as two commonly used techniques in hyperparameter tuning (Banerjee, 2022). While Grid Search exhaustively checks for every combination of specified hyperparameter values, Randomized Search does not equally consider all given parameter values. Instead, it samples a random combination of hyperparameters with each iteration, and this sampling can be specified in advance. Grid Search becomes impractical for large datasets due to the extensive search through all combinations, resulting in increased processing time. In contrast, Randomized Search offers faster processing time by randomly sampling hyperparameter combinations. However, it may not guarantee finding the optimal combination. Therefore, we chose Randomized Search to tune the hyperparameters for the Neural Network.

We tune five hyperparameters of the Neural Network classifier, which are max_iter, hidden_layer_sizes, alpha, activation. After fitting the training data into the model, the best parameters obtained through Randomized Search can be extracted from the final result. With Randomized Search CV, the precision, recall, and F1-score for class 1 have improved slightly compared to the classification report without hyperparameter tuning. Overall, hyperparameter

tuning has led to some enhancements in the model's ability to classify instances particularly for class 1.

```
Neural network confusion matrix:

[[44973   324]
 [ 5577   448]]

Neural network confusion matrix after using RandomizedSearch CV:

[[44885   412]
 [ 5483   542]]
```

Referring to the confusion matrix obtained, there is an increase in the number of true positives after hyperparameter tuning, with True Positives increasing from 448 to 542. Hence, there is an improvement in the model's ability to correctly identify instances of class 1. After using RandomizedSearch CV, the model correctly identified 542 instances as class 1 out of all instances that were actually class 1 and identified 44885 instances as class 0 out of all instances that were actually class 0.

Next, the ROC curve is plotted and shown below.

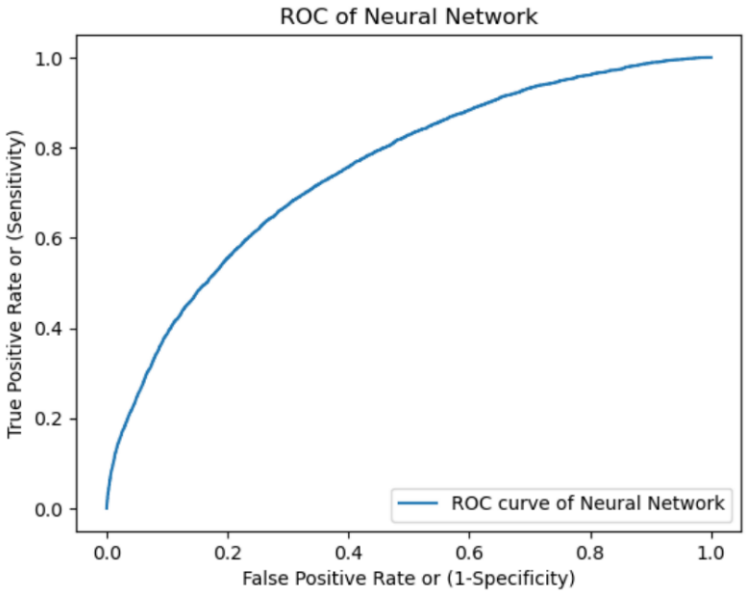


Figure 3.1.7: ROC Curve of Neural Network

3.2 Comparison & Evaluation

Evaluation metrics are numerical measurements employed to assess the performance and efficiency of a machine learning model (Srivastava, 2019). These metrics help in comparing various models or algorithms and offer insights into the model's effectiveness. The metrics included in the comparison and evaluation process are F1-score, AUC score and ROC curve.

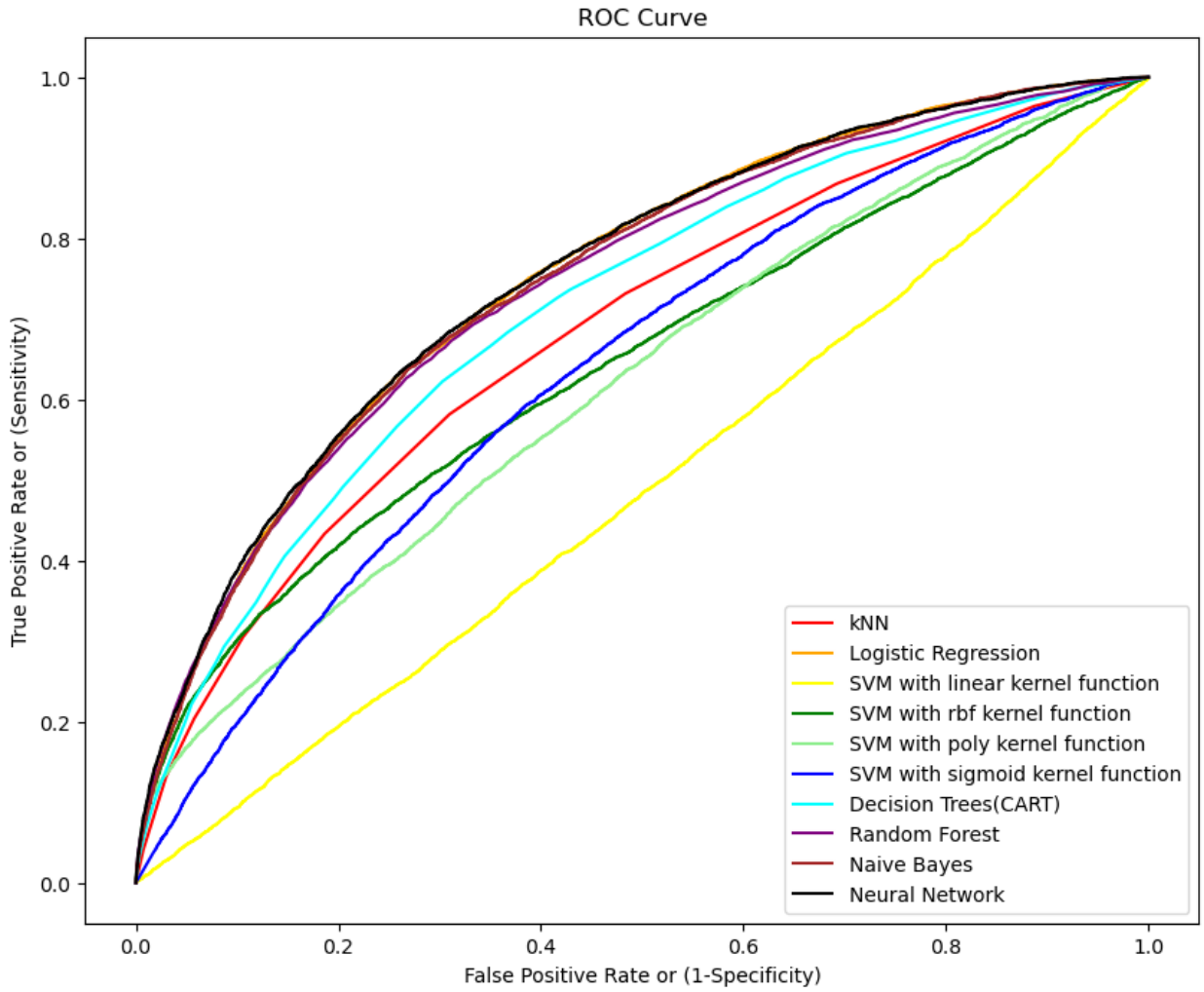
```
AUC score

kNN:  0.6817271567818375
Logistic Regression:  0.7507091755959767
#SVM with linear kernel function:  0.4861901491648893
SVM with rbf kernel function:  0.6425554274018311
SVM with poly kernel function:  0.6183075079303704
SVM with sigmoid kernel function:  0.6378868027954184
Decision tree:  0.713224878091365
Random Forest:  0.7413701016353386
Naive Bayes:  0.7472501499325293
Neural Network:  0.7527278779786009
```

The AUC score is a performance metric for evaluating binary classification models. It represents the area under the ROC curve, which plots the true positive rate against the false positive rate at various threshold settings. A higher AUC score indicates better performance, with the model effectively distinguishing between the two classes. Models with AUC scores closer to 1 exhibit strong performance, while scores closer to 0.5 suggest random classification. AUC scores facilitate direct comparisons between different models, helping in the selection of the most suitable model for the classification task. Therefore, AUC is valuable for datasets with imbalanced class distributions since it can evaluate the classifier's performance across both positive and negative classes in a two-dimensional manner.

Among the models evaluated, Logistic Regression, Naive Bayes, Random Forest, and the Neural Network exhibit relatively high AUC scores. Hence, this can indicate an effective separation of positive and negative instances. Conversely, models such as SVM with linear kernel function have poorer performance which can be shown by lower AUC scores. **Neural Network**

with the highest AUC scores of 0.7527 among all the models, signifies good model performance that is able to make accurate predictions and distinguish between classes with greater precision.



ROC curve visually represents a model's ability to differentiate between classes by plotting the True Positive Rate (Sensitivity) against the False Positive Rate (1 — Specificity) across various classification thresholds. A perfect classifier would pass through the top-left corner of the ROC space, where the TPR is 1 and the FPR is 0 (www.linkedin.com, n.d.). In contrast, a random guess would yield a diagonal line from the bottom-left to the top-right, which has AUC of 0.5 (Kılıç, 2023). ROC analysis remains unbiased towards models that excel in predicting the minority class while neglecting the majority class. Therefore, ROC curves demonstrate robustness to class imbalance which provide an unbiased evaluation even in datasets with uneven class distributions.

Based on the ROC curve, we observe that Neural Network, Naïve Bayes and Random Forest are closer to the top-left corner. Whereas SVM with linear kernel function has an almost diagonal line that show poorer performance. Overall, **Neural Network** model has the closest distance with the top-left corner which signify high sensitivity and specificity. Thus, it exhibits better discrimination between positive and negative cases.

Models of test set	f1-score (0)	f1-score(1)	AUC score
kNN	0.94	0.01	0.681727157
Logistic Regression	0.94	0.07	0.750709176
SVM with linear kernel function	0.94	0	0.486190149
SVM with rbf kernel function	0.94	0	0.642555427
SVM with poly kernel function	0.94	0	0.618307508
SVM with sigmoid kernel function	0.9	0.2	0.637886803
Decision Trees	0.94	0.04	0.713224878
Random Forest	0.94	0.06	0.741370102
Naïve Bayes	0.94	0.06	0.74725015
Neural Network	0.94	0.12	0.752727878

F1 score, a harmonic mean of precision and recall, is used as an evaluation metric in binary class classification tasks (Arize AI, n.d.). It combines precision and recall into a single measure to offer a comprehensive assessment of model performance. The F1 score accounts for both false positives and false negatives, making it particularly beneficial for imbalanced datasets. Unlike accuracy, which considers the correctly identified data across the dataset, may only be reliable when the dataset has an equal number of samples for each class and skewed by class imbalances. Therefore, F1 score, ROC and AUC score offer deeper insights into model performance, to enable us to fine-tune our models and select the optimal ones for deployment.

Based on the obtained F1-scores, we can see that most models achieve high F1-scores for class 0, thus showing strong performance in correctly identifying instances of the majority class. However, the F1-scores for class 1 vary across models. Models like SVM with sigmoid kernel function, Neural Network and Logistic Regression exhibit relatively higher F1-scores for class 1.

The **Neural Network** model stands out with a consistently high F1-score so that the performance is relatively balanced across both classes.

In conclusion, after evaluating F1-scores, ROC curve, and AUC scores, we determined that the **Neural Network** model is the optimal choice for deployment and making predictions on new datasets.

4.0 Interaction

The dataset 'NewApplicants.csv' is imported using the pandas library and all the 20 rows are displayed for observation. The shape of the dataset is determined to be (20, 24). Initially, the column 'LoanID' is removed to enhance clarity. As the output variable should not be included in the model evaluation process, it is separated from the input variables named 'Default'. Since categorical data cannot be directly interpreted, they are transformed into binary data. The original categorical columns are replaced with new binary columns. Finally, the transformed categorical data is concatenated with the dataset to create a new dataset for analysis.

	Age	Income	LoanAmount	CreditScore	MonthsEmployed	NumCreditLines	\
0	56	85994	50587	520	80	4	
1	69	50432	124440	458	15	1	
2	46	84208	129188	451	26	3	
3	32	31713	44799	743	0	3	
4	60	20437	9139	633	8	4	
5	25	90298	90448	720	18	2	
6	38	111188	177025	429	80	1	
7	56	126802	155511	531	67	4	
8	36	42053	92357	827	83	1	
9	40	132784	228510	480	114	4	
10	64	73743	140354	300	0	2	
11	68	21711	168231	352	78	2	
12	51	69492	122962	348	66	2	
13	41	61809	119238	444	34	2	
14	40	129890	116119	701	38	3	
15	19	37979	210682	541	109	4	
16	32	51953	189899	511	14	2	
17	56	84820	208294	597	70	3	
18	42	85109	60575	809	40	1	
19	62	22418	18481	636	113	2	

	InterestRate	LoanTerm	DTIRatio	Education_High School	...	\
0	15.23	36	0.44	False	...	
1	4.81	60	0.68	False	...	
2	21.17	24	0.31	False	...	
3	7.07	24	0.23	True	...	
4	6.51	48	0.73	False	...	
5	22.72	24	0.10	True	...	
6	19.11	12	0.16	False	...	
7	8.15	60	0.43	False	...	
8	23.94	48	0.20	False	...	
9	9.09	48	0.33	True	...	
10	4.12	12	0.24	False	...	
11	9.71	60	0.36	False	...	
12	10.83	48	0.27	True	...	
13	19.99	36	0.31	False	...	
14	9.91	24	0.23	True	...	
15	14.11	12	0.85	False	...	
16	11.55	24	0.21	True	...	
17	5.29	60	0.50	True	...	
18	20.90	48	0.44	True	...	
19	6.73	12	0.48	False	...	

	EmploymentType_Unemployed	MaritalStatus_Married	MaritalStatus_Single	\
0	False	False	False	
1	False	True	False	
2	True	False	False	
3	False	True	False	
4	True	False	False	
5	True	False	True	
6	True	False	True	
7	False	True	False	
8	False	False	False	
9	False	True	False	
10	False	False	True	
11	False	False	False	
12	False	False	False	
13	False	True	False	
14	False	False	False	
15	False	True	False	
16	False	False	False	
17	False	True	False	
18	False	False	True	
19	True	False	False	

	HasMortgage_Yes	HasDependents_Yes	LoanPurpose_Business
0	True	True	False
1	False	False	False
2	True	True	False
3	False	False	True
4	False	True	False
5	True	False	True
6	True	False	False
7	False	False	False
8	True	False	False
9	True	False	False
10	True	False	False
11	True	True	False
12	False	False	False
13	True	True	False
14	True	False	False
15	False	False	False
16	False	False	False
17	True	True	False
18	True	True	False
19	True	False	False

	LoanPurpose_Education	LoanPurpose_Home	LoanPurpose_Other
0	False	False	True
1	False	False	True
2	False	False	False
3	False	False	False
4	False	False	False
5	False	False	False
6	False	True	False
7	False	True	False
8	True	False	False
9	False	False	True
10	True	False	False
11	False	True	False
12	False	True	False
13	False	False	False
14	False	True	False
15	False	False	True
16	False	True	False
17	False	False	False
18	False	False	True
19	True	False	False

	HasCoSigner_Yes
0	True
1	True
2	False
3	False
4	False
5	True
6	True
7	True
8	False
9	True
10	True
11	False
12	False
13	True
14	True
15	False
16	False
17	True
18	False
19	True

[20 rows x 24 columns]

Through the checking of the existence of missing values in the data set, no missing value exists based on the figure below.

```
Missing value:
 Age                0
Income              0
LoanAmount          0
CreditScore         0
MonthsEmployed      0
NumCreditLines      0
InterestRate        0
LoanTerm            0
DTIRatio            0
Education           0
EmploymentType      0
MaritalStatus       0
HasMortgage         0
HasDependents       0
LoanPurpose         0
HasCoSigner         0
Default            20
dtype: int64
```

Duplicate entries and rows are checked for in the dataset. If any duplicates are found, they are removed. It is determined that there are no duplicated values or rows present.

```
Duplicate entries:  False
Number of duplicate rows :  (0, 17)
Number of duplicate rows :  (0, 17)
```

Then, a new dataset is predicted using Neural Network model which instantiates with the best hyperparameters obtained. The predictions obtained on the new dataset consist of binary values, 0 and 1. We also have probability scores, which represent the likelihood or confidence of the model's prediction.

```
Predictions on the new dataset:
 [0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0]
Probability scores on the new dataset:
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 0.]
```

Based on the prediction, for the 20 new applicants, 18 loan applications are approved while 2 are rejected.

REFERENCES

Amazon Web Services, Inc. (n.d.). *What is a Neural Network? - Artificial Neural Network Explained - AWS*. [online] Available at: https://aws.amazon.com/what-is/neural- n etwork/#:~:text=A%20neural%20network%20is%20a..

Analytics Vidhya. (2019). *Evaluation Metrics Machine Learning*. [online] Available at: ht tps://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics.

Arize AI. (n.d.). *Understanding and Applying F1 Score: AI Evaluation Essentials with Hands-On Coding Example*. [online] Available at: https://arize.com/blog-course/f1- s core/#:~:text=F1%20score%20is%20a%20measure.

Band, A. (2020). *How to find the optimal value of K in KNN?* [online] Medium. Available at: https://towardsdatascience.com/how-to-find-the-optimal-value-of-k-in-knn-35d936e554eb#:~:text=The%20optimal%20K%20value%20usually.

Banerjee, A. (2022). *Hyperparameter Tuning Using Randomized Search*. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2022/11/hyperparameter-tuning-using-randomized-search

Banoula, M. (2023). *The Complete Guide to Machine Learning Steps*. [online] Simplilearn.com. Available at: https://www.simplilearn.com/tutorials/machine-learning-tutorial/machine-learning-steps.

docs.aws.amazon.com. (n.d.). *Cross-Validation - Amazon Machine Learning*. [online] Available at: https://docs.aws.amazon.com/machine-learning/latest/dg/cross-validation.html.

Edgar, T.W. and Manz, D.O. (2017). *Logistic Regression - an overview | ScienceDirect Topics*. [online] www.sciencedirect.com. Available at: https://www.sciencedirect.com/topics/computer-science/logistic-regression.

GeeksForGeeks (2017). *Decision Tree - GeeksforGeeks*. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/decision-tree/.

GeeksforGeeks (2018). *K-Nearest Neighbours - GeeksforGeeks*. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/k-nearest-neighbours/.

Kanade, V. (2022). *What Is a Support Vector Machine? Working, Types, and Examples*. [online] Spiceworks. Available at: https://www.spiceworks.com/tech/big-data/articles/what-is-support-vector-machine/.

Kumar, A. (2023). *Using GridSearchCV with Logistic Regression Models: Examples*. [online] Analytics Yogi. Available at: https://vitalflux.com/gridsearchcv-logistic-regression-machine-learning-examples/#:~:text=GridSearchCV%20method%20is%20a%20one.

Kumar, N. (2019). *Naive Bayes Classifiers - GeeksforGeeks*. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/naive-bayes-classifiers/.

Medium. (n.d.). *Medium*. [online] Available at: https://medium.com/@ilyurek/roc-curve-and-auc-evaluating-model-performance-c2178008b02.

Python, R. (n.d.). *Logistic Regression in Python – Real Python*. [online] realpython.com. Available at: https://realpython.com/logistic-regression-python/#logistic-regression-in-python.

Ramakrishnan, M. (2022). *What is Classification in Machine Learning and Why is it Important?* [online] Emeritus Online Courses. Available at: https

s://emeritus.org/blog/artificial-intelligence-and-machine-learning-classification-in-machine-learning/.

R, S.E. (2021). *Understand Random Forest Algorithms With Examples (Updated 2024)*. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/#:~:text=Random%20Forest%20is%20a%20widely.

Tokuç, A.A. (2021). *How to Improve Naive Bayes Classification Performance? | Baeldung on Computer Science*. [online] www.baeldung.com. Available at: https://www.baeldung.com/cs/naive-bayes-classification-performance.

www.ibm.com. (n.d.). *What are Neural Networks? | IBM*. [online] Available at: https://www.ibm.com/topics/neural-networks..

www.linkedin.com. (n.d.). *What is the best way to evaluate a model using AUC-ROC curves?* [online] Available at: https://www.linkedin.com/advice/0/what-best-way-evaluate-model-using-auc-roc-curves-ja5nf#:~:text=AUC%2DROC%20curves%20can%20help.

www.sciencedirect.com. (n.d.). *Confusion Matrix - an overview | ScienceDirect Topics*. [online] Available at: https://www.sciencedirect.com/topics/engineering/confusion-matrix#:~:text=A%20confusion%20matrix%20is%20a.