Project Phase #2

Fall - CSE 587

Teammate 1: Harikumar Reddy Vengi Reddy (50518453)

Teammate 2: Deekshitha Devalla (50546172)

LOAN APPLICANT DATA FOR CREDIT RISK ANALYSIS

Problem Statement: In the context of financial lending, the goal is to develop predictive models that evaluate an applicant's creditworthiness in respect to loans.

The purpose of this task is to predict the probability of loan default using the dataset containing applicant characteristics such as age, income, home ownership status, years in job, intended purpose for loan, loan amount, interest rate, historical time of borrower history, and records of missed payments or this would aim at giving lenders a credible tool that helps them make safer choices, reduce default risks and enhance assessment processes in general.

Dataset Description: The dataset contains all relevant information regarding applicants of loans and their attributes.

Features are age, annual income, home ownership, employment length (in years), loan intent, loan grade, loan amount, loan interest rate, loan status, loan percent income, default history, credit history length.

Data source: Dataset is taken from Kaggle: https://www.kaggle.com/datasets

To train the dataset, we need to import few libraries that are required for training and plotting graphs like sklearn, matplotlib, pandas etc.

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import pickle
import xgboost as xgb

from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_recall_fscore_support
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix

from sklearn.cluster import KMeans
from sklearn import svm
from sklearn.linear_model import LogisticRegression
from sklearn import svm
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import AdaBoostClassifier
```

To start training, we need to split data into testing and training. For training the data we only take the input features that are required for predicting the loan_status of a person. Considering the important features and dropping the output column X data frame is created. Taking the output column (loan_status) alone Y is created.

```
X.columns
```

```
Index(['age_range_cat', 'income_range_cat', 'person_home_ownership_cat',
       'person_emp_length_normalized', 'loan_intent_cat', 'loan_grade_cat',
       'loan_amount_range_cat', 'loan_int_rate_normalized',
       'loan_percent_income', 'cb_person_default_on_file_cat',
       'cb_person_cred_hist_length_normalized'],
      dtype='object')
```

```
Y = risk_df[['loan_status']]
Y.columns
```

```
Index(['loan_status'], dtype='object')
```

```
count = Y['loan_status'].value_counts()[0]
print(count)
```

```
25318
```

```
count1 = Y['loan_status'].value_counts()[1]
print(count1)
```

```
7088
```

Eighty percent of the data is taken to train the model, leaving twenty percent for testing. Randomly dividing the dataset into train and test subsets ensure that both subsets are representative of the overall data distribution.

splitting dataset

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
```

```
X_train = X_train.to_numpy()
X_test = X_test.to_numpy()
Y_train = Y_train.to_numpy()
Y_test = Y_test.to_numpy()

Y_train = Y_train.ravel()
Y_train
```

```
array([0, 0, 1, ..., 0, 1, 0])
```

```
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
```

```
(25924, 11)
(6482, 11)
(25924,)
(6482, 1)
```

ML Algorithms

Predicting whether the person's loan application should be accepted or not based on the column loan_status (0 for non-default and 1 for default), which means a person with loan_status 0 is good status (meaning the loan is being repaid on time) and 1 is bad loan status (there are late payments and not following the agreement for repayments).

Based on many input features related to the person, loan_status is predicted, in this case of predicting whether loan_status is 0 or 1 will be a binary classification problem.

1. Logistic Regression:

Logistic Regression is one of the most widely used statistical methods for binary classification problems, especially for credit risk analysis. Estimating the loan_status based upon the predictors is considered as a binary classification problem.

Reasons to choose logistic regression:

- It is easy to implement and interpret and the algorithm is also fast due to its simplicity. As we have already pre-processed the data set and did exploratory data analysis understanding the impact of different factors on the prediction is easier.
- Outcome is binary, predicting whether the person will default the loan (1) or not (0).
- It uses sigmoid function where the probability score is between 0 and 1, they can be classified as default or non-default.

logistic regression

```
def LogisticRegression_sklearn(X_train, Y_train, X_test):
    logistic_regression_model = LogisticRegression()
    logistic_regression_model.fit(X_train, Y_train)
    save_model(logistic_regression_model, 'logistic_regression_model.pkl')
    Y_pred = logistic_regression_model.predict(X_test)
    return Y_pred
```

After training the model using a scikit-learn library, some of the evaluation metrics is printed such as accuracy, precision, recall and F1 score.

```
logistic_regression_y_pred = LogisticRegression_sklearn(X_train, Y_train, X_test)
```

```
get_metrics(Y_test, logistic_regression_y_pred)
```

```
Acuracy of the model: 0.8403270595495218
Precision of the model: 0.710789766407119
Recall of the model: 0.4519094766619519
F1 score of the model: 0.5525291828793774
```

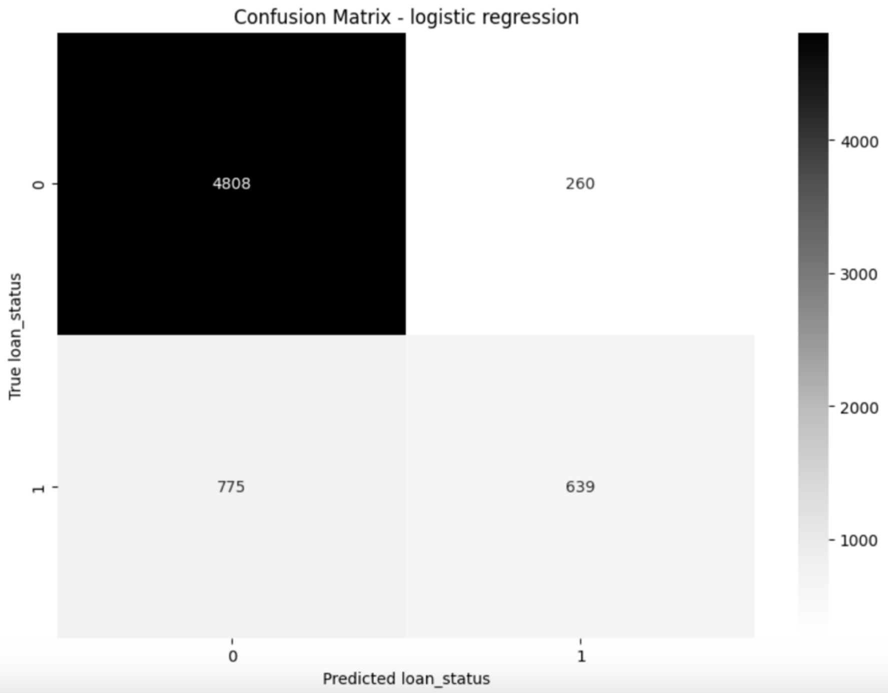
Interpreting the results obtained from logistic regression model via visualization.

Visualization:

Confusion matrix is used especially in classification task, as the performance is described in form of heat map.

```
cm_lr = confusion_matrix(Y_test, logistic_regression_y_pred)
```

```
plt.figure(figsize=(10,7))
sns.heatmap(cm_lr, annot=True, fmt='g', cmap='Greys')
plt.xlabel('Predicted loan_status')
plt.ylabel('True loan_status')
plt.title('Confusion Matrix - logistic regression')
plt.show()
```




```
count = Y_test['loan_status'].value_counts()[0]
print(count)
```

```
5068
```

```
count1 = Y_test['loan_status'].value_counts()[1]
print(count1)
```

```
1414
```

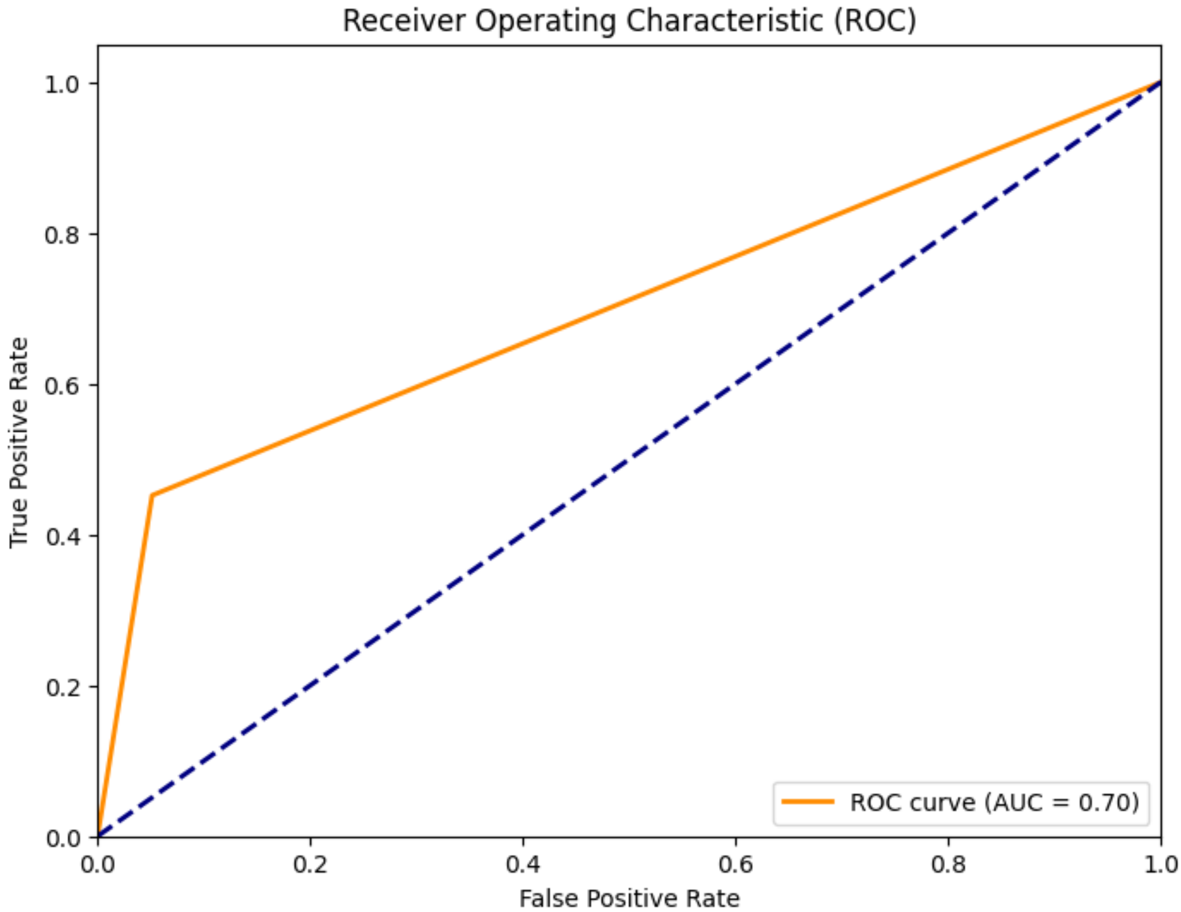
```
print(classification_report(Y_test, logistic_regression_y_pred))
```

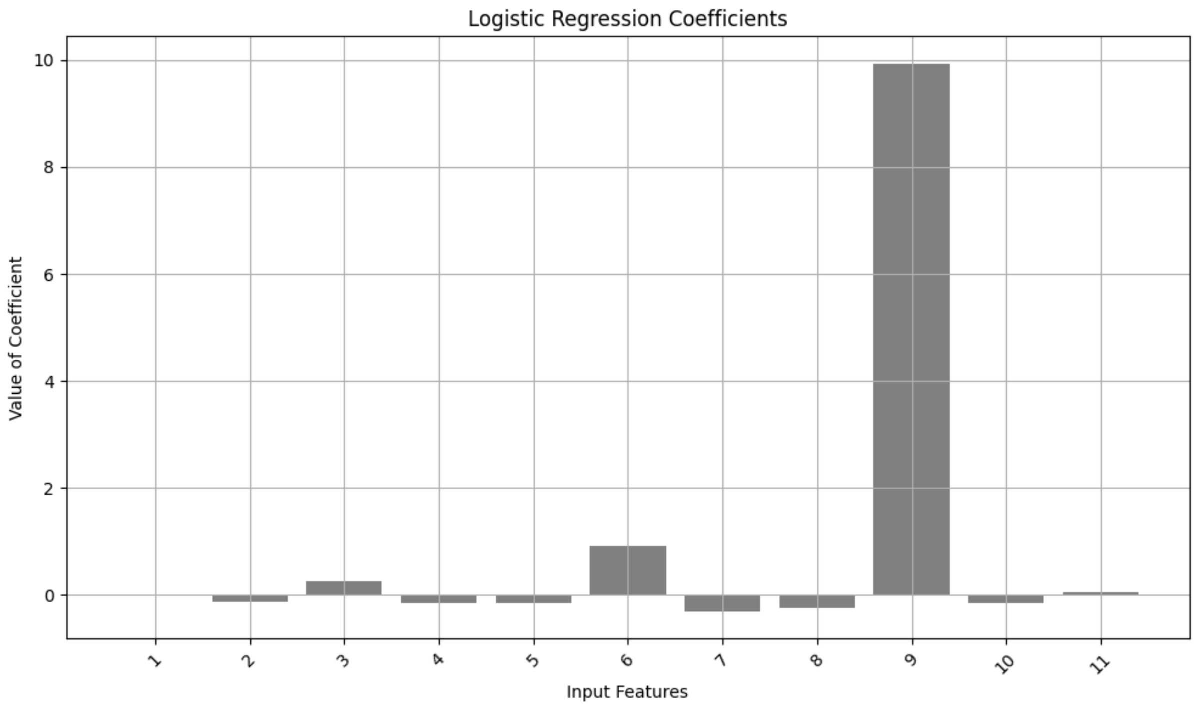
```
              precision    recall  f1-score   support

           0       0.86      0.95      0.90      5068
           1       0.71      0.45      0.55      1414

    accuracy                           0.84      6482
   macro avg       0.79      0.70      0.73      6482
weighted avg       0.83      0.84      0.83      6482
```

The ROC (Receiver Operating Characteristic) curve is a graphical plot used to evaluate the performance of a binary classification system. In the context of credit risk analysis, where the binary classification problem is to predict whether someone will default on a loan (positive class) or not (negative class), the ROC curve provides a powerful method for assessing how well your model distinguishes between the two classes.





Explanation and Analysis:

- To understand it better, confusion matrix is plotted where True Positives(TP), True Negatives (TN), False Positive (FP) and False Negatives(FN) are clearly displayed.
- loan_status that were correctly predicted to default value 1 are TP and with non-default (value 0) are TN.
- FP are loans that were incorrectly predicted to default and to non-default are FN.
- The model has correctly identified a large number of non-defaults, which means it is effective in recognizing good loans with the number 4808 (TN).
- Only 639 are identified default which is relatively low compared to the number of actual defaults (TP).
- False positives are 260 were predicting to default but did not. Borrowers might get denied loans based on the model's prediction which is false.
- FN value is 775 where model failed to predict the default persons which can be a financial loss if the lender sanctions the loan to these people as they originally have default as a status.
- Accuracy is 84% which indicates that the loan status predictions are almost correct and accurate.
- Precision tells that when the model predicted a loan would default, it was correct about 71% of the time.

- Recall states that the model identified 45% of all actual defaults. The recall is relatively low, which means that more than half of the defaults were not caught by the model.
- F1 score of 55% indicates that the model is not as effective in balancing precision and recall, leaning more towards precision.
- Even though accuracy is high, the recall is low and needs to be improved. The model is missing a significant number of actual defaults and results in less recall value.
- The logistic regression model has delivered good accuracy and precision which exhibits a significant shortfall in recall. This aspect is particularly critical in financial risk management, where the costs associated with missed defaults are substantial.
- The logistic regression coefficients help to illustrate how much each predictor matters for the probability of default.
- Due to linearity of logistic regression, its simplicity may miss those complex patterns that cannot be found out until some additional feature engineering becomes involved.
- Data pre-processing had been a crucial part for implementing logistic regression model on credit risk analysis dataset.
- To enhance the model ability we need to potentially explore more sophisticated modelling techniques to achieve a more equal balance between precision and recall.

2. SVM (Support Vector Machines):

SVM is an machine learning algorithm widely used for classification tasks. As our problem is a binary classification, SVM best suits to predict the loan status of a person. SVM operates by finding the hyperplane that separates negative and positive values upon finding the nearest points (support vectors) from each class as objective is to maximize the margin between the dataset's classes.

Reasons to choose support vector machines:

- SVM is effective in high dimensional space and in our dataset we have many input features for analysing.
- SVM is also best adapting non-linear relationships using kernel functions, which can model the complex relation between the features and loan status prediction.
- SVM is robust against overfitting, making the model suitable for datasets with a more features.

Below is the SVM model fitting with kernel as a linear function.

```python
def SVM_sklearn(X_train, Y_train, X_test):
    svm_model = svm.SVC(kernel='linear')
    svm_model.fit(X_train, Y_train)
    save_model(svm_model, 'svm_model.pkl')
    Y_pred = svm_model.predict(X_test)
    return Y_pred
```

Evaluation metrics for Support Vector machines is as follows:

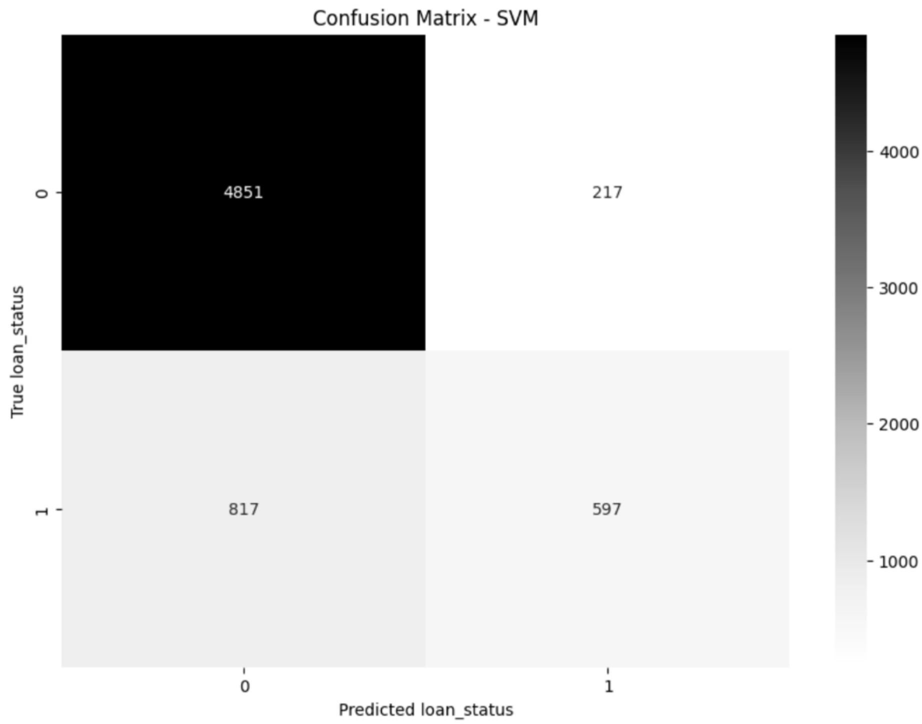
```python
svm_y_pred = SVM_sklearn(X_train, Y_train, X_test)
```

```python
get_metrics(Y_test, svm_y_pred)
```

```
Acuracy of the model: 0.8404813329219377
Precision of the model: 0.7334152334152334
Recall of the model: 0.4222065063649222
F1 score of the model: 0.5359066427289049
```

```python
cm_svm = confusion_matrix(Y_test, svm_y_pred)
```

```python
plt.figure(figsize=(10,7))
sns.heatmap(cm_svm, annot=True, fmt='g', cmap='Greys')
plt.xlabel('Predicted loan_status')
plt.ylabel('True loan_status')
plt.title('Confusion Matrix - SVM')
plt.show()
```

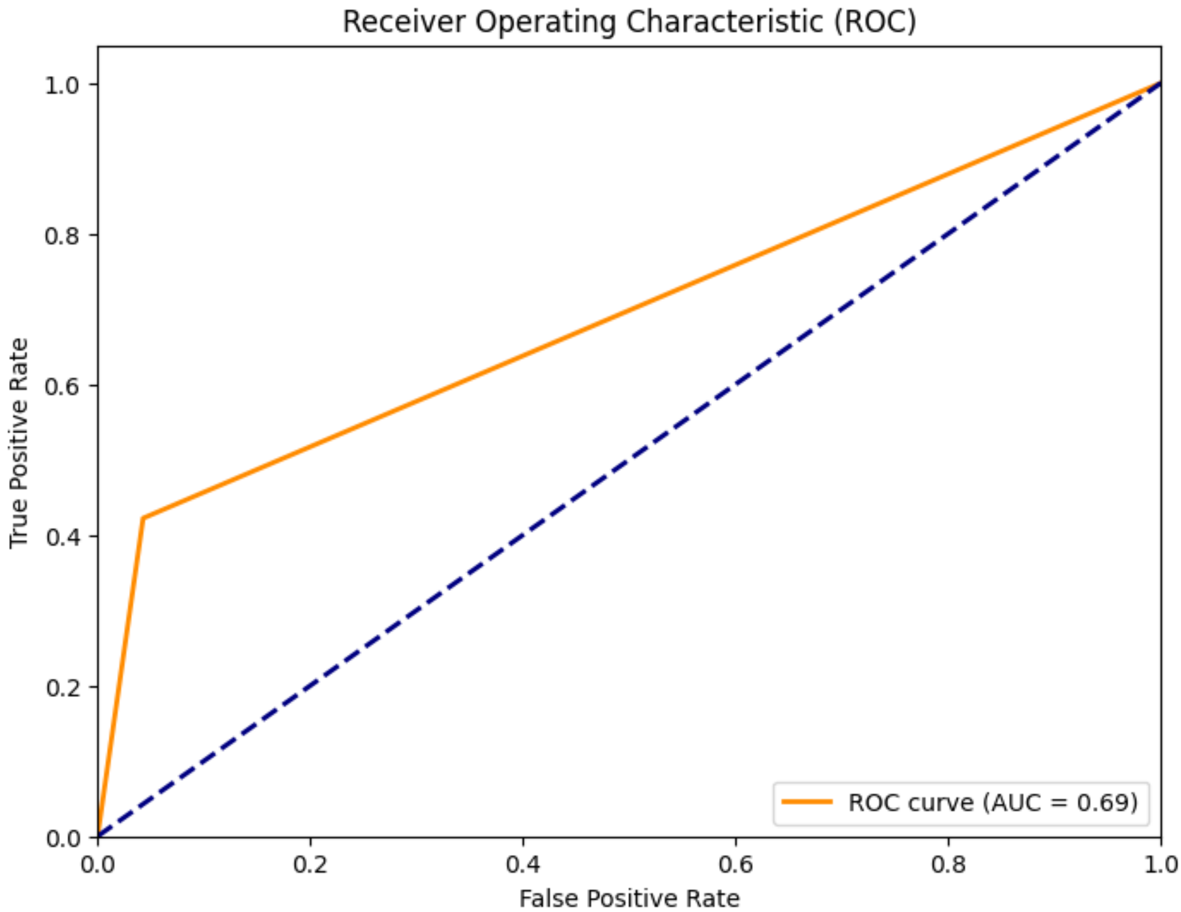


```
print(classification_report(Y_test, svm_y_pred))
```

```
              precision    recall  f1-score   support

           0       0.86      0.96      0.90      5068
           1       0.73      0.42      0.54      1414

    accuracy                           0.84      6482
   macro avg       0.79      0.69      0.72      6482
weighted avg       0.83      0.84      0.82      6482
```



Explanation and Analysis:

- Similar to the metrics displayed and discussed in logistic regression, plotted confusion matrix and printed the evaluation metrics.
- We see that the accuracy of the model is about 84% which is almost similar to the logistic regression model accuracy. It implies that this SVM model is correctly predicting the majority of loan status.
- SVMs precision is little high (73%) when compared to logistic regression, indicating that when it predicts a default, it is likely to be correct resulting in minimizing the False positive values.
- Recall (42%) being little low than the precision values indicating that there is an improvement in predicting loan status correctly over logistic regression model. This clearly depicts that SVM model is good at predicting true defaults.
- F1 Score (53%), balances the precision and recall values, defining that the model is relatively balanced.
- SVM model identified 4851 non-defaults (TN), which is good.
- These are instances of 217 (FP) where the model predicted defaults that did not happen, which is relatively low.
- 597 cases were correctly identified by the model as number of defaults (True Positives) that are crucial for credit risk management.
- There are defaults of count 817 (FN) that model failed to predict. This needs to be minimised by implementing another algorithms.
- The SVM performance metrics suggest that it is effective at predicting loan defaults, with a good balance between identifying true defaults and avoiding false alarms.
- The model’s decision function is based on the separation margin between classes, seems to handle the dataset's complexity well.
- The margin distances from the decision boundary gave a sense of confidence in the predictions.
- The support vectors highlighted the cases that were most difficult to classify suggesting deeper insights into the patterns associated with credit risk.
- This model seems to find an good balance when taking into account the costs associated with false positives and false negatives.
- SVM models are harder to understand than logistic regression models, especially when the kernels are non-linear.
- This could make it difficult to communicate the model's conclusions to stakeholders.
- SVM is comparatively improved than logistic regression recall.

3. Random Forest:

Random Forest is an ensemble learning technique that works by building several decision trees in the training stage and producing a class that represents the mean prediction (regression) or the mode of the classes (classification) of the individual trees. It improves accuracy and robustness by fusing flexibility with the simplicity of decision trees.

In binary classification, one of two probable outcomes is predicted. The two results of credit risk analysis are usually repayment of the loan (non-default) or non-repayment of the loan (default). Financial institutions need to be able to predict this binary conclusion with accuracy in order to control and reduce risk.

Reasons to choose random forest:

- Random Forest is known as being strong and good at building precise models that have a high degree of complexity interactions and cross relations.
- Credit risk is influenced by complex interactions between various factors such as age, income, employment history, credit history, etc. as random forest can naturally model these interactions, best algorithm to implement for our credit risk dataset.
- It is well known that predicting power is more for random forest algorithm in predicting loan status which is a binary classification.
- This algorithm works as an interpretable model which estimates the significance of various attributes in predicting loan defaults.
- Random Forest performs good with regard to biased data sets, and such situation is frequently present in the credit risk where “Non-defaults” outranging “Defaults”.

```
def random_forest_sklearn(X_train, Y_train, X_test):
    rf_model = RandomForestClassifier()
    rf_model.fit(X_train, Y_train)
    save_model(rf_model, 'rf_model.pkl')
    Y_pred = rf_model.predict(X_test)
    return Y_pred
```

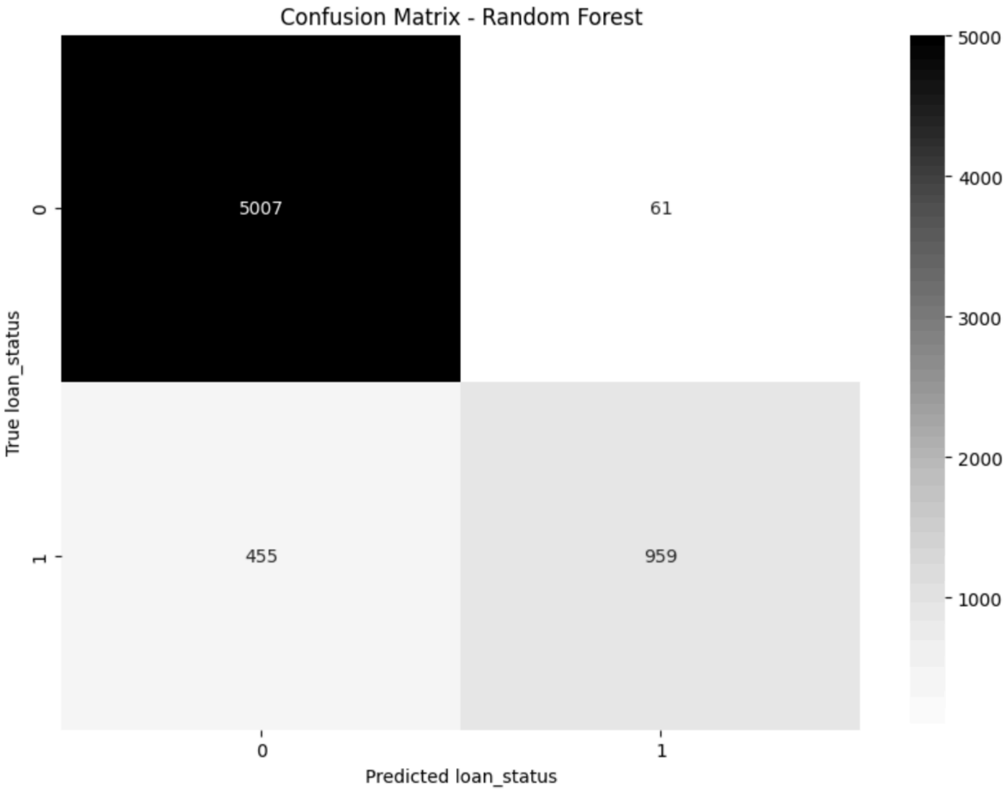
```
rf_y_pred = random_forest_sklearn(X_train, Y_train, X_test)
```

```
get_metrics(Y_test, rf_y_pred)
```

```
Acuracy of the model: 0.9203949398333847
Precision of the model: 0.9401960784313725
Recall of the model: 0.6782178217821783
F1 score of the model: 0.7880032867707478
```

```
cm_rf = confusion_matrix(Y_test, rf_y_pred)
```

```
plt.figure(figsize=(10,7))
sns.heatmap(cm_rf, annot=True, fmt='g', cmap='Greys')
plt.xlabel('Predicted loan_status')
plt.ylabel('True loan_status')
plt.title('Confusion Matrix - Random Forest')
plt.show()
```

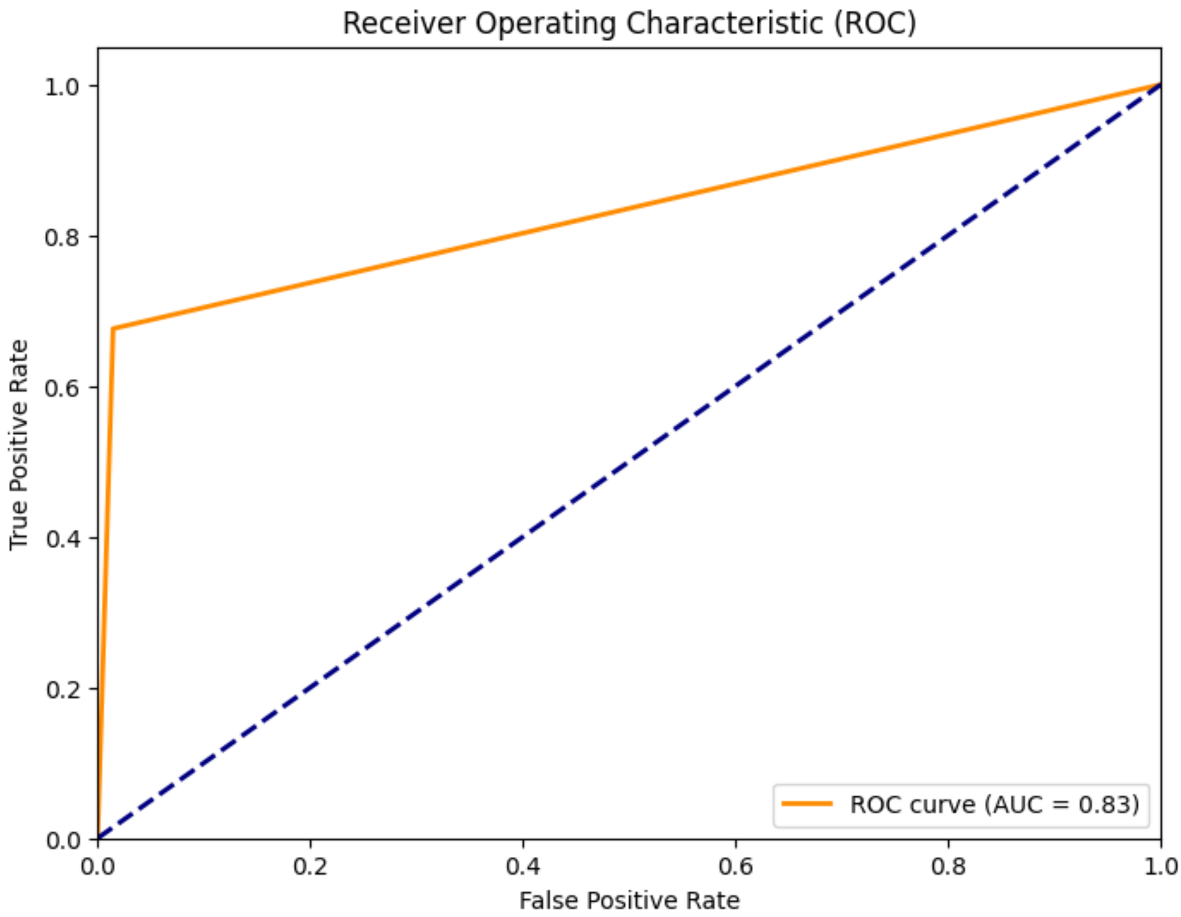



```
print(classification_report(Y_test, rf_y_pred))
```

```
              precision    recall  f1-score   support

           0       0.92      0.99      0.95      5068
           1       0.94      0.68      0.79      1414

    accuracy                           0.92      6482
   macro avg       0.93      0.83      0.87      6482
weighted avg       0.92      0.92      0.92      6482
```



Explanation and Analysis:

- A high accuracy (92%) indicating that the Random Forest model correctly predicted a majority of the loan statuses.
- This high precision (94%) suggests that when the model predicts a default, it is very likely to be correct, minimizing the risk of false positives.
- The recall resulting in 67% which is significantly higher than both the SVM and logistic regression models, indicating that Random Forest is more effective at identifying actual defaults of loan.
- F1 score – 78% indicates that there is a robust balance between precision and recall.
- TN – 5007 means that the random forest model is highly effective in predicting the non-default values.
- FP – 61 where there are very few loans were incorrectly classified as defaults, which could mean fewer customers are incorrectly denied loans.
- TP – 959 are the number of defaults for loan were correctly identified, which is key for risk management.
- FN – 455 the number of missed defaults is lower compared to the SVM and Explanation and Analysis:
- Random Forest shows a strong performance in both identifying non-defaults and defaults, making it a powerful tool for credit risk assessment.

- The high precision and recall indicate that the model is robust, providing confidence in its predictive power.
- In order to help with credit policy decisions, the Random Forest algorithm can also show which features have the greatest predictive power for defaults.
- The fact that the Random Forest model can rank features by importance helps to make it easier to interpret, even though it is more complex than logistic regression and likely SVM.
- The increase in recall value compared to SVM and logistic regression, suggests that random forest is better suited for our problem which may result in modelling of complex structures with in the dataset.
- The Random Forest model performs much better than the previously tested logistic regression and SVM models, showing significant gains in recall and precision.
- It demonstrates why it is a good fit for the credit risk analysis task by minimizing false positive predictions and successfully striking a balance between the requirement to predict loan defaults with accuracy.
- The feature importance scores can also help risk analysts and decision-makers better understand the main causes of loan defaults, which can improve their ability to make strategic decisions about risk mitigation and the creation of credit policies.

4. Decision Trees:

Decision Tree is like a flow chart tree in which inner node represents the input feature and branch represents the decision rule and the leaf node resembling the outcome that needs to be predicted loan_status in credit risk analysis dataset. In case of binary classification problem there are only two outcomes one is positive and other being negative.

This implies that based on a series of nodes (features) the leaf node should be categorized into default loan status and non-default loan status. Internal nodes are like person's income, loan intent, age, employment history of a person etc.

Reasons to choose Decision Trees:

- One of the main reason is that decision tree is simpler and easier to understand and implement for binary classification problem.
- Decision trees are less computationally demanding to train, which makes them appropriate for situations requiring rapid model development.
- In credit risk datasets, where the relationship between variables like income and loan repayment is not always linear, they can naturally model non-linear relationships between features.
- Decision trees will provide a clear understanding about predicting the loan_status value which helps lenders to sanction the loan or not.

```
def decision_tree_sklearn(X_train, Y_train, X_test):
    dt_model = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
    dt_model.fit(X_train, Y_train)
    save_model(dt_model, 'dt_model.pkl')
    Y_pred = dt_model.predict(X_test)
    return Y_pred
```

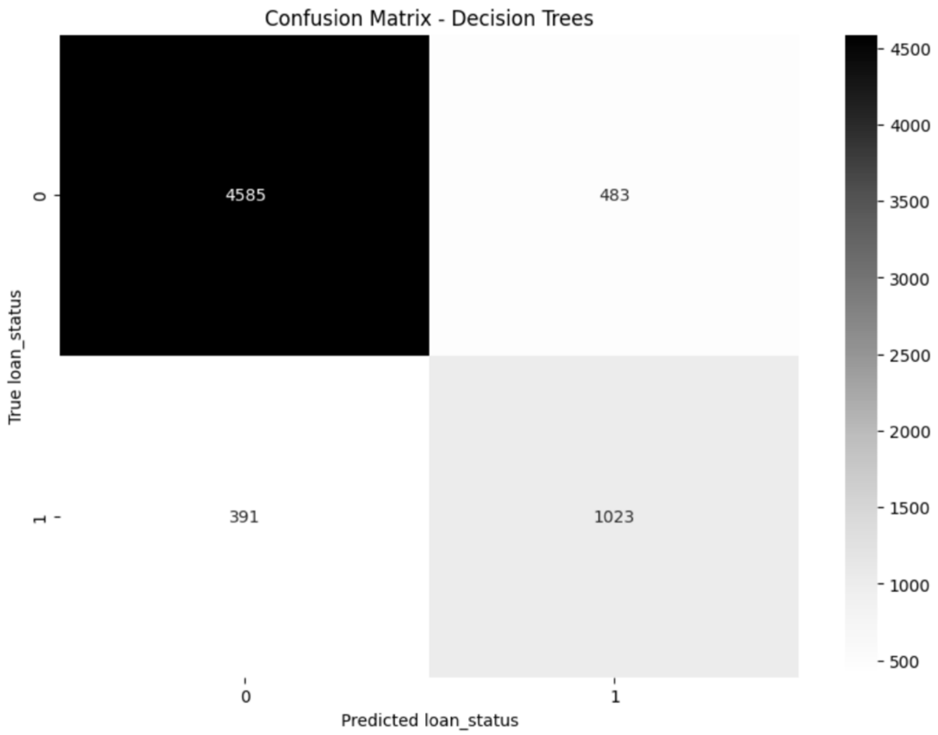
```
decision_tree_y_pred = decision_tree_sklearn(X_train, Y_train, X_test)
```

```
get_metrics(Y_test, decision_tree_y_pred)
```

```
Acuracy of the model: 0.8651650725084851
Precision of the model: 0.6792828685258964
Recall of the model: 0.7234794908062234
F1 score of the model: 0.7006849315068494
```

```
cm_decision_tree = confusion_matrix(Y_test, decision_tree_y_pred)
```

```
plt.figure(figsize=(10,7))
sns.heatmap(cm_decision_tree, annot=True, fmt='g', cmap='Greys')
plt.xlabel('Predicted loan_status')
plt.ylabel('True loan_status')
plt.title('Confusion Matrix - Decision Trees')
plt.show()
```

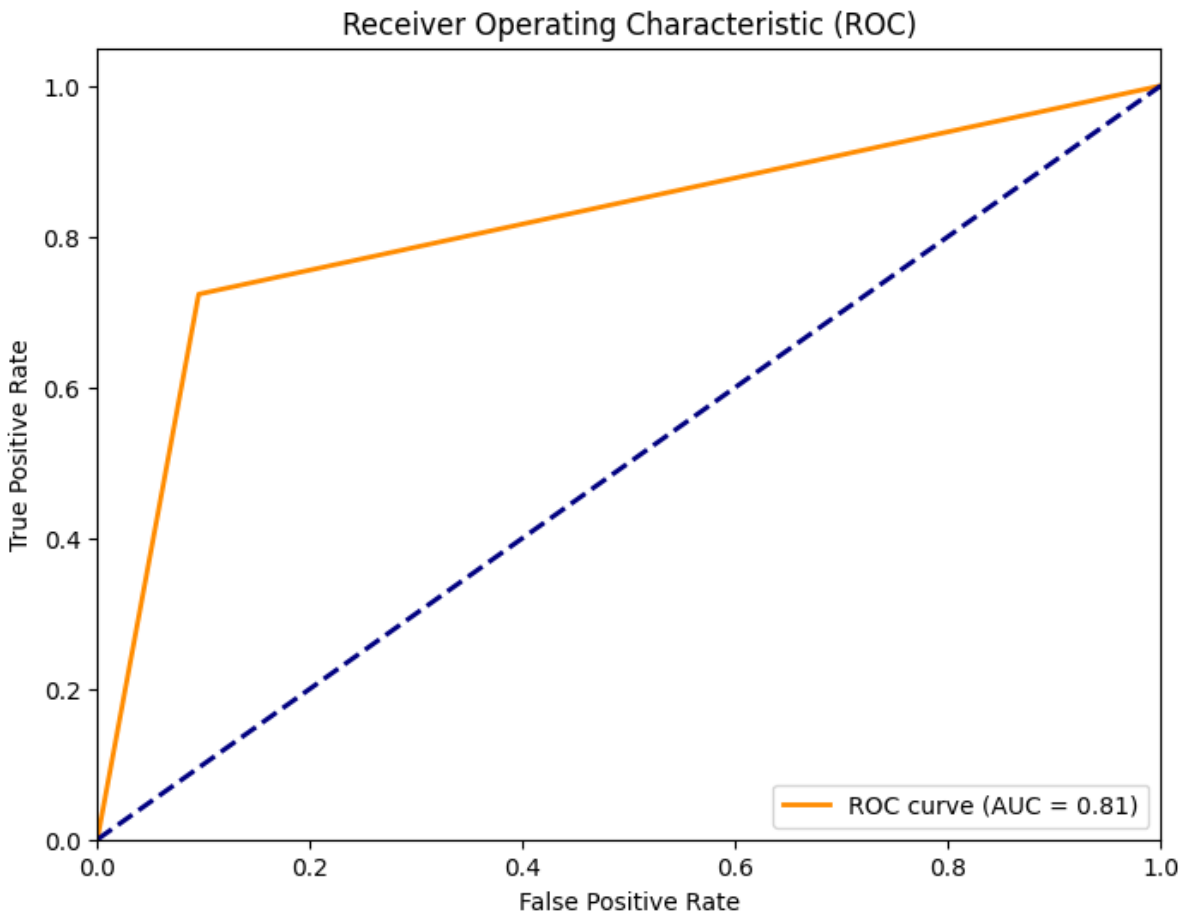


```
print(classification_report(Y_test, decision_tree_y_pred))
```

```
              precision    recall  f1-score   support

           0       0.92      0.90      0.91      5068
           1       0.68      0.72      0.70      1414

    accuracy                           0.87      6482
   macro avg       0.80      0.81      0.81      6482
weighted avg       0.87      0.87      0.87      6482
```



Explanation and Analysis:

- A high accuracy (92%) indicating that the Random Forest model correctly predicted a majority of the loan statuses.
- Evaluation metrics indicates that the Decision Tree correctly predicted the loan status in approximately 86% of cases, which is fairly high but lower than the Random Forest model.
- The model's precision (67%) is moderate, suggesting that when a default is predicted, it is correct about two-thirds of the time.
- The recall is quite high (72%), which means the Decision Tree is able to identify a significant proportion of actual default cases.
- The F1 score (70%) is relatively high, indicating a good balance between precision and recall.
- The model correctly identified many of the non-default cases, which is important for not denying loans to potentially good borrowers indicating TN - 4585.
- The number of false positives is higher (483) than that of the Random Forest model, indicating more instances where the model incorrectly predicted defaults
- The Decision Tree model correctly identified a large number of defaults, which is crucial for credit risk management with TP-1023.
- The model has fewer false negatives than the SVM model, suggesting it is better at catching defaults FN – 391.
- The high recall rate is favourable in the context of credit risk, as missing out on predicting defaults is costlier than false positives.
- In spite of its interpretability, the model's performance suggests that a single Decision Tree can be a competitive predictive tool, although it does not perform as well as the ensemble approach of Random Forest.
- The simplicity of Decision Trees can facilitate regulatory compliance and business decision-making, as each decision rule is clear and explicit.
- The Decision Tree model shows a strong ability to distinguish between defaulting and non-defaulting loans, with a particularly high recall.
- Decision Trees are prone to overfitting, especially if not pruned properly or if the tree is allowed to grow too complex.
- As we already performed data pre-processing and data cleaning, overfitting might not be an issue for this dataset.
- As with Random Forest, Decision Trees provide insights into feature importance, which can be used to understand which features are driving predictions.
- Decision trees suit for this credit risk analysis due to its simplicity and speed resulting in good accuracy.

5.XGBoost:

An enhanced version of gradient boosting algorithms is called eXtreme Gradient Boosting (XGBoost). Its speed and performance have made it more and more popular in the field of machine learning. XGBoost sequentially builds a sequence of decision trees, each of which attempts to fix the errors committed by the one before it.

In credit risk analysis, binary classification using XGBoost entails determining a borrower's likelihood of defaulting on their loan, either (1) or (0). Based on variables like credit score, debt-to-income ratio, employment history, previous financial behaviour.

Reasons to choose XGBoost:

- XGBoost is trained on historical credit data, learning to predict default probabilities. It uses its gradient boosting framework to minimize a loss function and produce a powerful model.
- It is an winning algorithm in machine learning due to its performance, scalability and flexibility.
- This model includes L1 and L2 regularization, which can prevent overfitting which is really important in credit risk modelling where the cost of a wrong prediction can be very high.
- XGBoost can handle a mix of categorical and numerical features with ease.

```
def xgboost_sklearn(X_train, Y_train, X_test):
    xgboost_model = xgb.XGBClassifier(objective='binary:logistic', seed=42)
    xgboost_model.fit(X_train, Y_train)
    save_model(xgboost_model, 'xgboost_model.pkl')
    Y_pred = xgboost_model.predict(X_test)
    return Y_pred
```

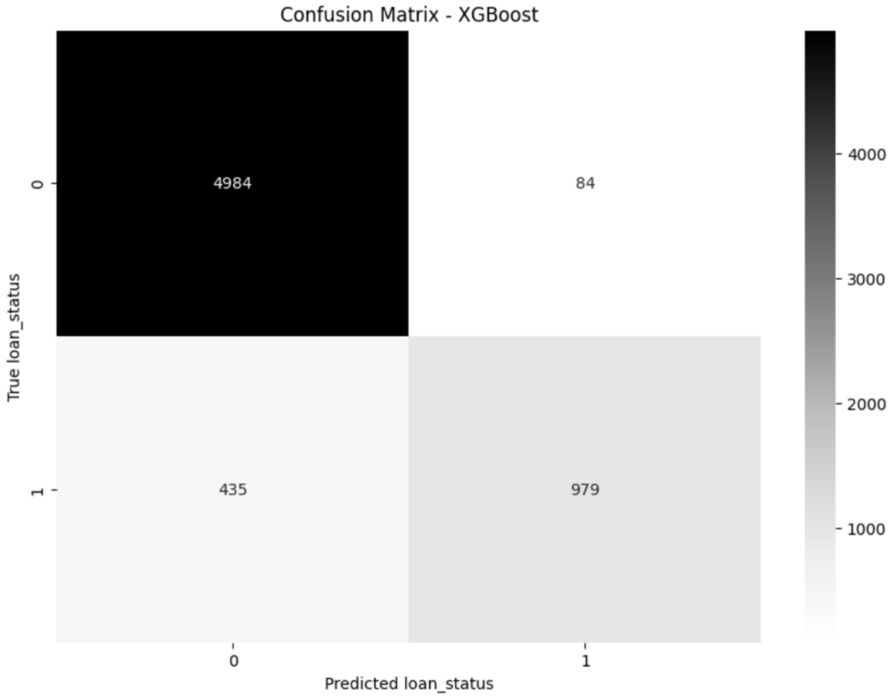
```
xgboost_y_pred = xgboost_sklearn(X_train, Y_train, X_test)
```

```
get_metrics(Y_test, xgboost_y_pred)
```

```
Acuracy of the model: 0.919932119716137
Precision of the model: 0.9209783631232361
Recall of the model: 0.6923620933521923
F1 score of the model: 0.7904723455793297
```

```
cm_xgbooxt = confusion_matrix(Y_test, xgboost_y_pred)
```

```
plt.figure(figsize=(10,7))
sns.heatmap(cm_xgbooxt, annot=True, fmt='g', cmap='Greys')
plt.xlabel('Predicted loan_status')
plt.ylabel('True loan_status')
plt.title('Confusion Matrix - XGBoost')
plt.show()
```

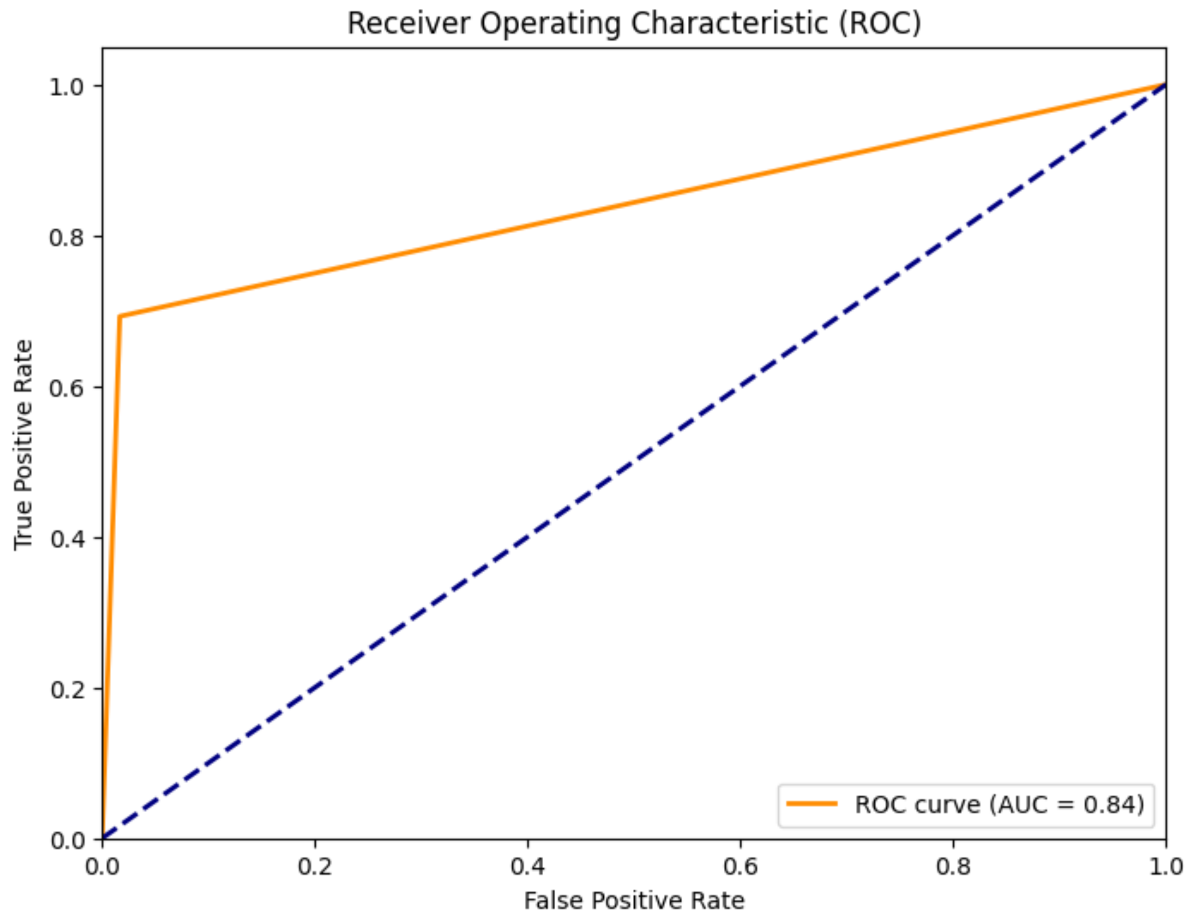



```
print(classification_report(Y_test, xgboost_y_pred))
```

```
              precision    recall  f1-score   support

           0       0.92      0.98      0.95      5068
           1       0.92      0.69      0.79      1414

    accuracy                           0.92      6482
   macro avg       0.92      0.84      0.87      6482
weighted avg       0.92      0.92      0.92      6482
```



Explanation and Analysis:

- 91% of accuracy rate shows that XGBoost is performing well on the overall dataset.
- Precision with (92%) indicates a high level of reliability in the positive predictions the model makes.
- Recall (69%) suggests a good ability to identify most of the actual positive cases (defaults) in the dataset.
- F1 Score (79%) reflects a strong balance between precision and recall, which is particularly important in credit risk modelling.
- TN – 4984, The model accurately identified the majority of non-defaulting loans.
- FP – 84, A low number of loans were incorrectly predicted to default, which is preferable in risk-averse scenarios.

- TP – 979, The model correctly identified a high number of defaults.
- FN – 435, The number of defaults that were missed by the model, though substantial, is less than what was seen with the Decision Tree model.
- With a high precision rate, XGBoost is effective at predicting loan defaults, making it a valuable tool for credit risk management.
- The recall has improved compared to some previous models, indicating a better identification of actual defaults, which is crucial for reducing the risk of unexpected credit losses.
- Although not as interpretable as simpler models like Decision Trees, XGBoost's ability to rank features can still provide actionable insights.
- This model has demonstrated an impressive ability to discern between default and non-default loans with a high degree of accuracy and precision.
- XGBoost includes built-in regularization, which helps prevent overfitting, a common issue in tree-based models.
- The XGBoost model has shown a strong performance in predicting loan defaults, achieving high marks in both precision and recall.
- Its balanced F1 score suggests that it can reliably differentiate between defaulting and non-defaulting loans, making it a solid choice for credit risk analysis.
- The high level of accuracy and the ability to handle complex patterns in the data may also enable more nuanced risk assessment and decision-making processes.

6. Naïve Bayes:

Using the Bayes theorem and strong (naive) independence assumptions between the features is the foundation of naive Bayes classifiers. Even though they are some of the most basic Bayesian network models, their simplicity, speed, and efficacy in some situations have made them a popular option despite their feature independence assumption.

In the context of credit risk, binary classification using Naive Bayes involves predicting whether or not a person will default on a loan (positive class) or not (negative class). Usually, this prediction is based on characteristics taken from the person's transactional behaviour, demographic information, and credit history.

Reasons to choose Naïve Bayes:

- Based on the Bayes theorem, naive Bayes classifiers strongly assume the independence of the features.
- They work especially well in classification tasks where the conditional independence assumption holds reasonably well or where the features are independent.
- Naive Bayes models are known for being fast and efficient to train, especially on large datasets.
- This model is known for being fast and efficient to train, especially on large datasets.

- They can perform remarkably well, especially in cases where the assumption of independence holds and when the dimensionality of the inputs is high.

```
def naive_bayes_sklearn(X_train, Y_train, X_test):
    naive_bayes_model = GaussianNB()
    naive_bayes_model.fit(X_train, Y_train)
    save_model(naive_bayes_model, 'xgboost_model.pkl')
    Y_pred = naive_bayes_model.predict(X_test)
    return Y_pred
```

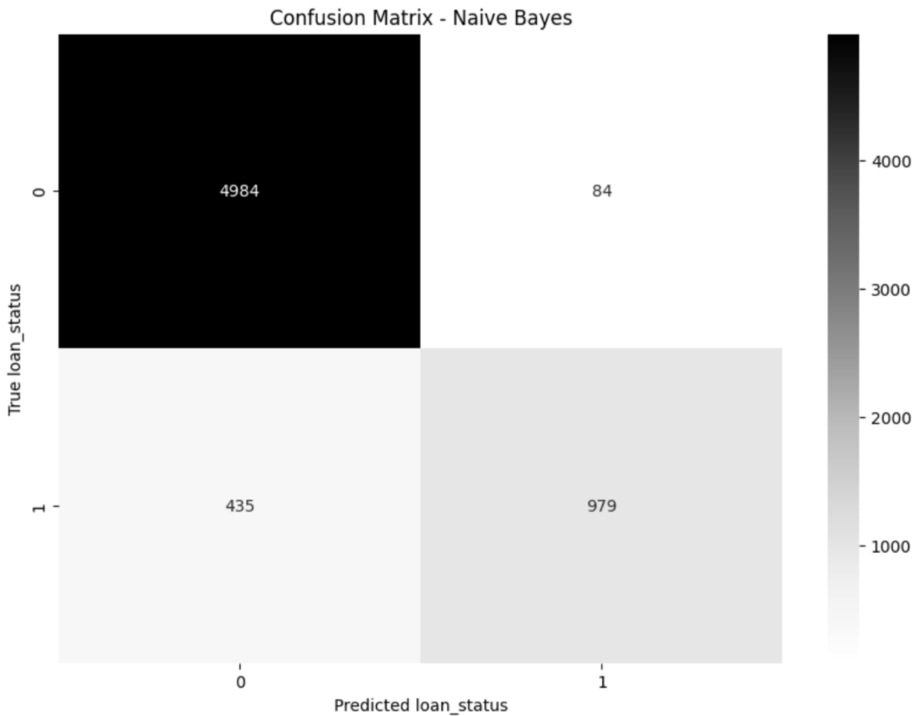
```
nb_y_pred = naive_bayes_sklearn(X_train, Y_train, X_test)
```

```
get_metrics(Y_test, nb_y_pred)
```

```
Acuracy of the model: 0.8171860536871336
Precision of the model: 0.5739186571981924
Recall of the model: 0.6287128712871287
F1 score of the model: 0.6000674991562605
```

```
cm_bayes = confusion_matrix(Y_test, nb_y_pred)
```

```
plt.figure(figsize=(10,7))
sns.heatmap(cm_xgbooxt, annot=True, fmt='g', cmap='Greys')
plt.xlabel('Predicted loan_status')
plt.ylabel('True loan_status')
plt.title('Confusion Matrix - Naive Bayes')
plt.show()
```

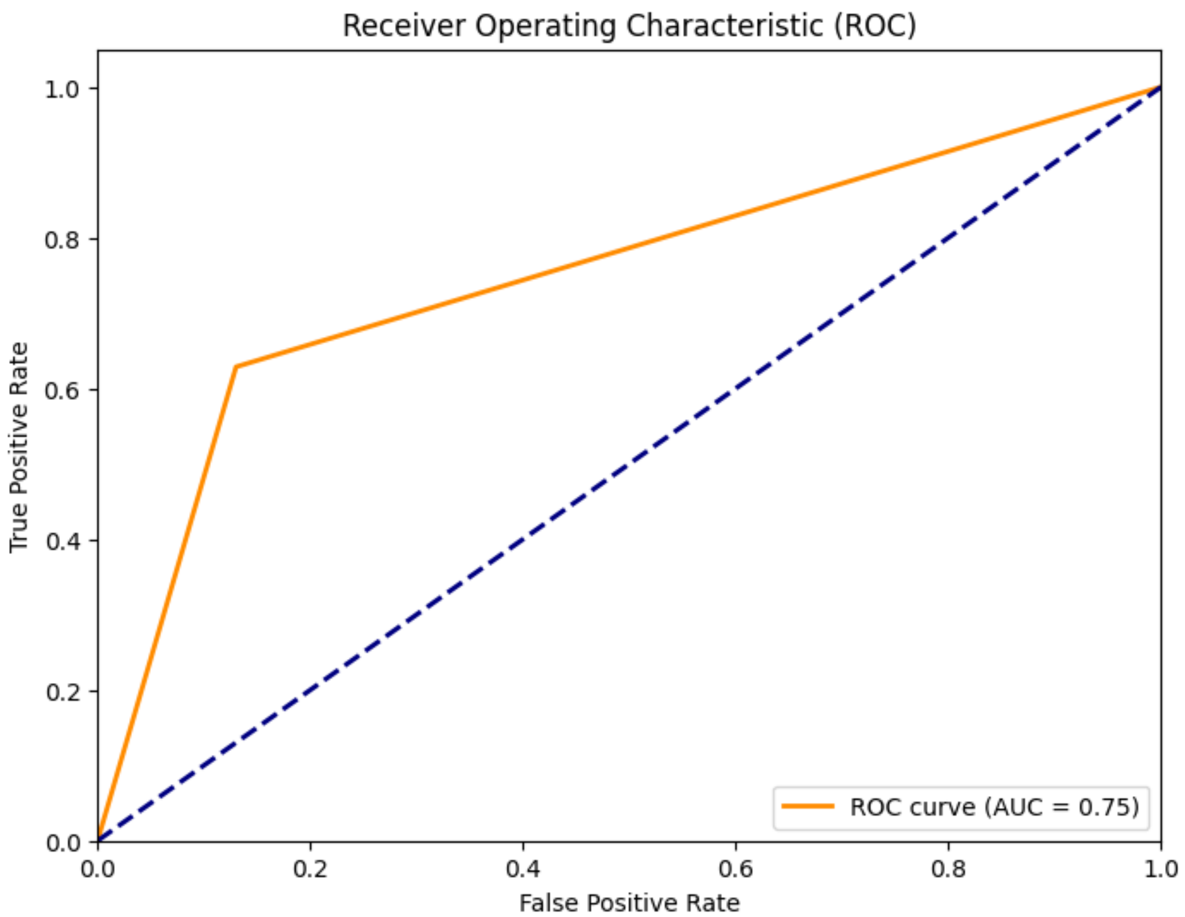


```
print(classification_report(Y_test, nb_y_pred))
```

```
              precision    recall  f1-score   support

           0       0.89      0.87      0.88      5068
           1       0.57      0.63      0.60      1414

    accuracy                           0.82      6482
   macro avg       0.73      0.75      0.74      6482
weighted avg       0.82      0.82      0.82      6482
```



Explanation and Analysis:

- Accuracy (81%), The model has a reasonable accuracy, indicating a fair number of predictions are correct but is lower compared to more complex models like XGBoost or Random Forest.
- Precision (57%), The precision is moderate, which suggests that when the model predicts a loan default, it is correct a little over half the time.
- Recall (62%), The recall indicates that the model is reasonably good at identifying actual defaults, though there is room for improvement.
- F1 Score (60%), The F1 score, which considers both precision and recall, shows that the model is moderately effective but may not balance the two as well as other models.
- TN - 4984, The model correctly identified a large number of non-defaulting loans, indicating good specificity.
- FP – 84, There are few loans incorrectly labelled as defaults, which is good for minimizing unnecessary follow-ups.
- TP – 979, A substantial number of defaults were correctly identified, which is important for risk detection.
- FN – 435, The model failed to catch a notable number of defaults, indicating that it may miss some high-risk loans.
- While Naive Bayes is a simpler model and easier to implement, it may not capture complex relationships as effectively as other models.
- Given its moderate precision and recall, Naive Bayes could be more suitable for applications where the cost of false positives and negatives is not too high, or as an initial screening model.
- Naive Bayes' performance can be sensitive to how the data is pre-processed, especially how categorical variables are handled and whether the features truly meet the independence assumption.
- The Naive Bayes model shows adequacy in differentiating between the defaulting and non-defaulting loans but is outperformed by more complex models like Random Forest and XGBoost.
- The model's performance may be limited by the assumption of feature independence, which is rarely the case in real-world data, especially in complex domains like credit risk.
- The model's strength lies in its rapid training and prediction times, making it suitable for applications where speed is a critical factor.
- The Naive Bayes classifier provides a passable but unexceptional solution for predicting loan defaults in the credit risk dataset thanks to its probabilistic methodology.
- Although its moderate precision and recall imply that it can be applied to preliminary credit risk assessments, further models or techniques may be required to improve its predictive power for more precise financial risk management decision-making.

7. Gradient Boosting Machines (GBM):

Gradient Boosting Machines (GBM) are a class of machine learning algorithms that optimize predictive models by applying boosting techniques.

An ensemble method called "boosting" turns a number of weak classifiers into a single, powerful classifier. GBM constructs trees one by one, with each new tree aiding in the correction of mistakes made by trees that have already been trained.

A weighted combination of the predictions made by the weak learners, each with a different accuracy, is the model's output.

Reasons to choose GBM:

- Using GBM for binary classification in credit risk analysis, one can predict two possible outcomes: whether or not a borrower will default on a loan (positive class) or not (negative class). Many input features are used to train the predictive model, including the borrower's loan characteristics, credit history, and demographics.
- GBM is a powerful ensemble learning technique known for its predictive accuracy, especially in classification problems.
- GBM can uncover complex relationships in the data by building trees in a sequential manner that correct the mistakes of previous ones.
- GBM provides insights into feature importance, allowing analysts to identify which factors are most influential in predicting defaults.

```
def gbm_sklearn(X_train, Y_train, X_test):
    gbm_model = GradientBoostingClassifier(n_estimators=100, learning_rate=1.0, max_depth=1, random_state=42)
    gbm_model.fit(X_train, Y_train)
    save_model(gbm_model, 'xgboost_model.pkl')
    Y_pred = gbm_model.predict(X_test)
    return Y_pred
```

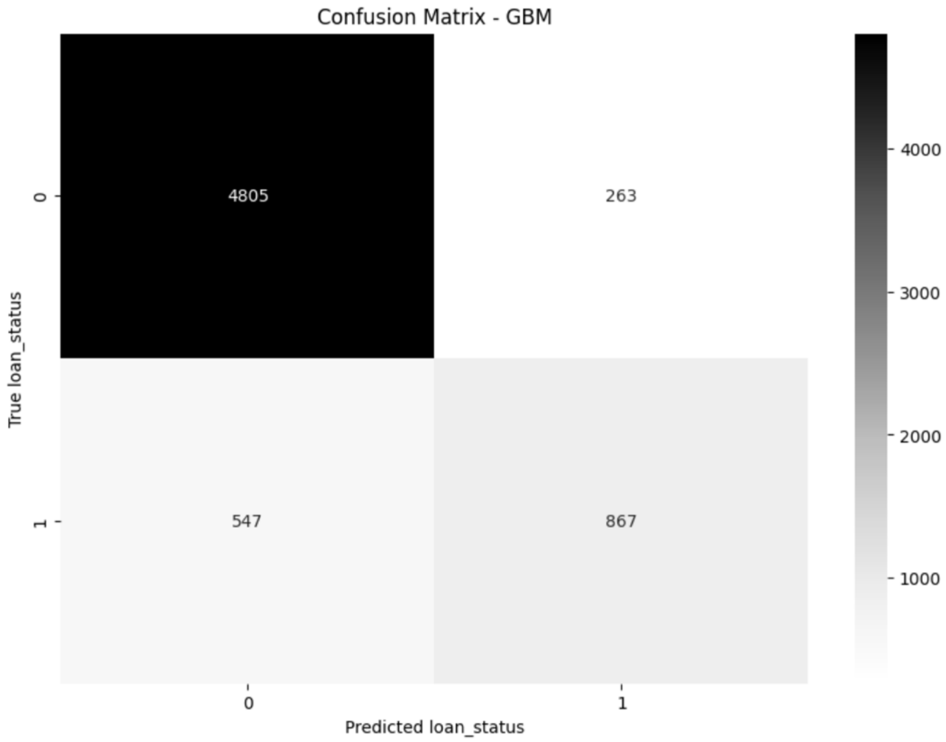
```
gbm_y_pred = gbm_sklearn(X_train, Y_train, X_test)
```

```
get_metrics(Y_test, gbm_y_pred)
```

```
Acuracy of the model: 0.875038568343104
Precision of the model: 0.7672566371681416
Recall of the model: 0.6131541725601132
F1 score of the model: 0.6816037735849058
```

```python
cm_gbm = confusion_matrix(Y_test, gbm_y_pred)
```

```python
plt.figure(figsize=(10,7))
sns.heatmap(cm_gbm, annot=True, fmt='g', cmap='Greys')
plt.xlabel('Predicted loan_status')
plt.ylabel('True loan_status')
plt.title('Confusion Matrix - GBM')
plt.show()
```

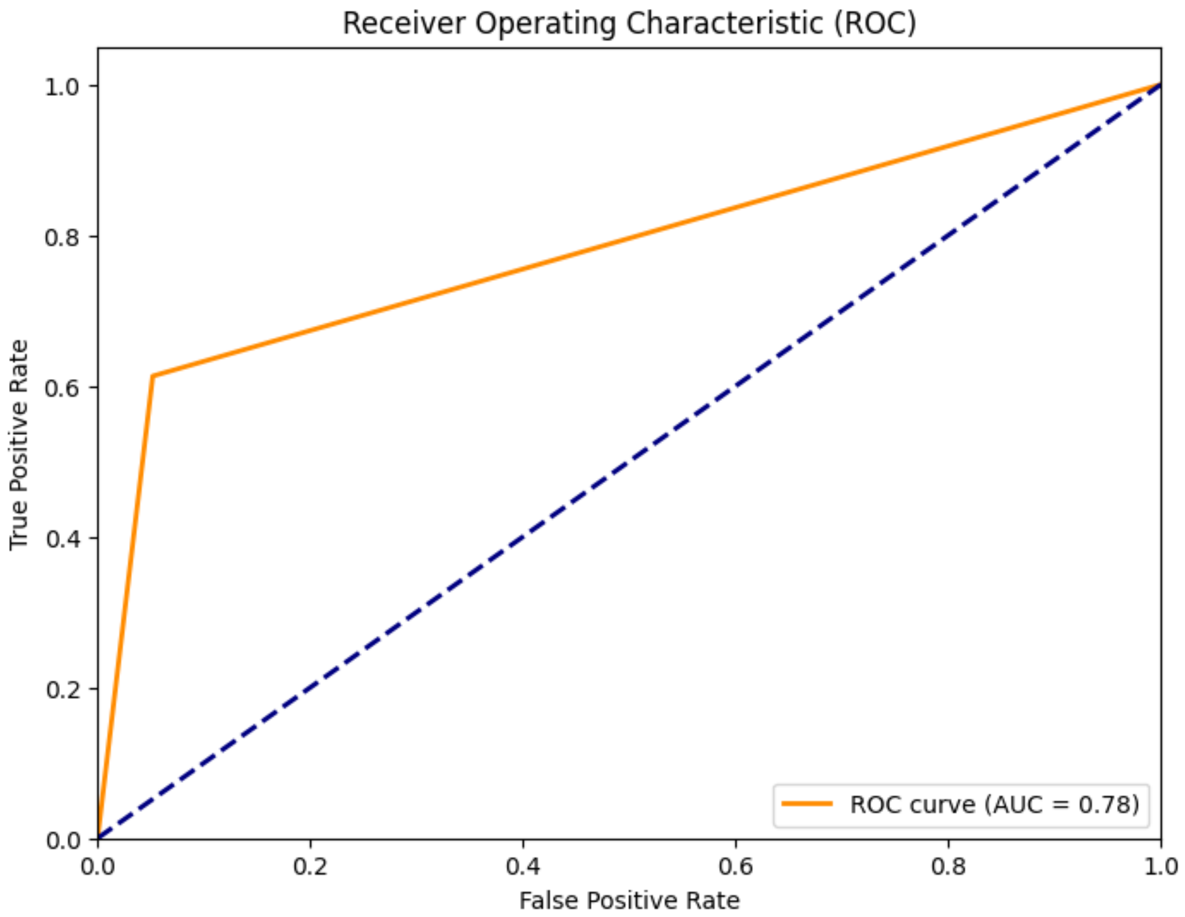



```python
print(classification_report(Y_test, gbm_y_pred))
```

```
              precision    recall  f1-score   support

           0       0.90      0.95      0.92      5068
           1       0.77      0.61      0.68      1414

    accuracy                           0.88      6482
   macro avg       0.83      0.78      0.80      6482
weighted avg       0.87      0.88      0.87      6482
```



Explanation and Analysis:

- Accuracy (87%): GBM model has a high accuracy, indicating that a significant majority of predictions are correct.
- Precision (76 %): A relatively high precision implies that the model has a strong ability to correctly predict loan defaults when it does so.
- Recall (61%): The recall is moderate, showing that the model can identify a good proportion of the actual defaults.
- F1 Score (68%): A robust F1 score indicates a balance between precision and recall, suggesting that the model is well-calibrated for both identifying defaults and minimizing false positives.

- True Negatives (TN - 4805): The model successfully identified a large number of non-defaulting loans, which is crucial for not denying loans to creditworthy applicants.
- False Positives (FP - 263): The number of loans incorrectly labelled as defaults is moderate, indicating some room for improvement in the model's specificity.
- True Positives (TP - 867): The model correctly predicted a substantial number of defaults, which is key for lenders to mitigate risk.
- False Negatives (FN - 547): The model missed some defaults, suggesting that some high-risk loans might not be flagged.
- The GBM model demonstrates a strong capability to distinguish between defaults and non-defaults, with high accuracy and a good balance between precision and recall.
- While more complex than simpler models like Decision Trees, GBM's performance justifies its use, especially in capturing the complex patterns often found in financial data.
- GBM includes parameters that help control overfitting, such as tree depth and learning rate, which need careful tuning to optimize model performance.
- Effective Classification: GBM is effective at classifying loan defaults, which is crucial in credit risk management to minimize potential losses.
- Trade-offs: The model presents a trade-off between false positives and false negatives, which financial institutions need to evaluate based on the cost associated with each.
- Data Pre-processing and Feature Engineering: The performance of GBM can be sensitive to the quality of data pre-processing and feature engineering, emphasizing the importance of these steps.

References:

https://xgboost.readthedocs.io/en/stable/python/python_api.html

https://www.datacamp.com/tutorial/xgboost-in-python

https://machinelearningmastery.com/gradient-boosting-with-scikit-learn-xgboost-lightgbm-and-catboost/

https://www.datacamp.com/tutorial/random-forests-classifier-python

https://www.tutorialspoint.com/scikit_learn/scikit_learn_decision_trees.html