INFO - 521
Introduction to Machine Learning
Final Project

German Bank Loan Defaulter Analysis

By

Abhay Kumara Sri Krishna Nandiraju

Table of Contents

1. Introduction:

The main objective of this project is to develop a machine learning model for the bank to predict whether the customer will default or not based on the historical data of various customers that did and did not default on their loan. It is crucial to predict loan defaulters as they can cause financial losses and affect the bank's ability to lend loans to other customers. Identifying loan defaulters helps banks perform risk management in an optimized manner. The German bank is facing issues with loan defaulters and wants to build a machine learning model to address the problem using a dataset that contains historical data of the customers who have taken loans from it. The dataset has the data of a thousand(1000) customers consisting of various attributes along with their respective default status indicating whether they defaulted on their loan or not.

The dataset has 17 columns and 1000 rows where the columns represent various attributes or features of the customers. Each row represents a customer along with their corresponding feature variables. Each customer is represented by the following 17 attributes:

a. 'checking_balance': It represents the amount of money available in the customer's checking account.(DM- Deutsche Mark)
b. 'months_loan_duration': It represents the duration since the loan was taken in months.
c. 'credit_history': It represents the credit history of the customer.
d. 'purpose': It represents the reason for which the loan was taken.
e. 'amount': It represents the amount of loan taken.
f. 'savings_balance': It represents the savings account balance..
g. 'employment_duration': It represents the duration of employment.
h. 'percent_of_income': It represents the loan installment amount as the percent of customer's income.
i. 'years_at_residence': It represents the number of years the applicant has lived at their current residence.
j. 'age': It represents the customer's age.

k. ‘other_credit’: It represents the other existing credits.
l. ‘housing’: It represents the type of housing owned by the customer.
m. ‘existing_loans_count’: It represents the existing count of each customer’s loans.
n. ‘job’: It represents the customer’s job type.
o. ‘dependents’:It represents the number of dependents the customer has.
p. ‘phone’: It represents whether the customer has a phone or not.
q. ‘default’: It indicates whether the customer has defaulted on their loan or not.

The dataset has 17 columns out of which the ‘default’ column is the target or the response that needs to be predicted using the remaining variables or features. It is essential to check the number of categorical and numerical variables present in the dataset and in addition to this, each categorical and numerical variable’s distribution should be computed to get an overview of each feature’s distribution. It is also crucial to check the relationship between variables as it may help us in reducing the number of predictors required to train the machine learning models. Therefore, it is interesting to check the distribution of different variables along with the relationship between them.

Interesting questions that need to be explored in this project are:

Q1. How are the categorical and numerical variables distributed?
Q2. Is there any correlation among the variables?
Q3. Which model is best suitable for the given task of predicting loan defaulters?
Q4. What are the top three important features of the dataset in the best suitable model?

2. Methods and Materials:

a. Exploratory Data Analysis(EDA):

The dataset has 1000 entries with 17 columns with no null values in any of the columns. Out of the 17 columns, 10 are categorical and the remaining 7 are numerical. Each of 10 columns consist of information regarding a categorical variable and each of the 7 columns consist of information about a numerical variable. The ten categorical variables are phone, housing, other_credit, job, savings_balance, purpose, checking_balance, credit_history, employment_duration and default. The seven numerical variables are months_loan_duration, amount, percent_of_income, years_at_residence, age, existing_loans_count and dependents. The categorical variable default is the response that needs to be predicted and the remaining variables are treated as predictors. The dataset has no missing values, null values and duplicate rows in it.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 17 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   checking_balance      1000 non-null   object
 1   months_loan_duration  1000 non-null   int64
 2   credit_history        1000 non-null   object
 3   purpose               1000 non-null   object
 4   amount                1000 non-null   int64
 5   savings_balance       1000 non-null   object
 6   employment_duration   1000 non-null   object
 7   percent_of_income     1000 non-null   int64
 8   years_at_residence    1000 non-null   int64
 9   age                   1000 non-null   int64
 10  other_credit          1000 non-null   object
 11  housing               1000 non-null   object
 12  existing_loans_count  1000 non-null   int64
 13  job                   1000 non-null   object
 14  dependents            1000 non-null   int64
 15  phone                 1000 non-null   object
 16  default               1000 non-null   object
dtypes: int64(7), object(10)
memory usage: 132.9+ KB
```

Fig-1: Dataset Information

Exploring Categorical Variables:

In this section, each categorical variable is explored by plotting pie-charts and count-plots. There are 700 non-defaulters and 300 defaulters in the dataset indicating the presence of class imbalance in the dataset. The dataset is biased towards customers that did not default. Out of all the customers, 596 possess a phone whereas 404 do not. Based on the job type there are 630 skilled workers, 200 unskilled workers, 148 management workers and 22 unemployed customers. 713 customers own a house whereas 179 customers live in a rented home. Surprisingly, 108 customers neither own a house nor live in a rented home. Around 814 customers did not take credit from anywhere but 139 customers took other credit from a bank and 47 customers took credit from a store.

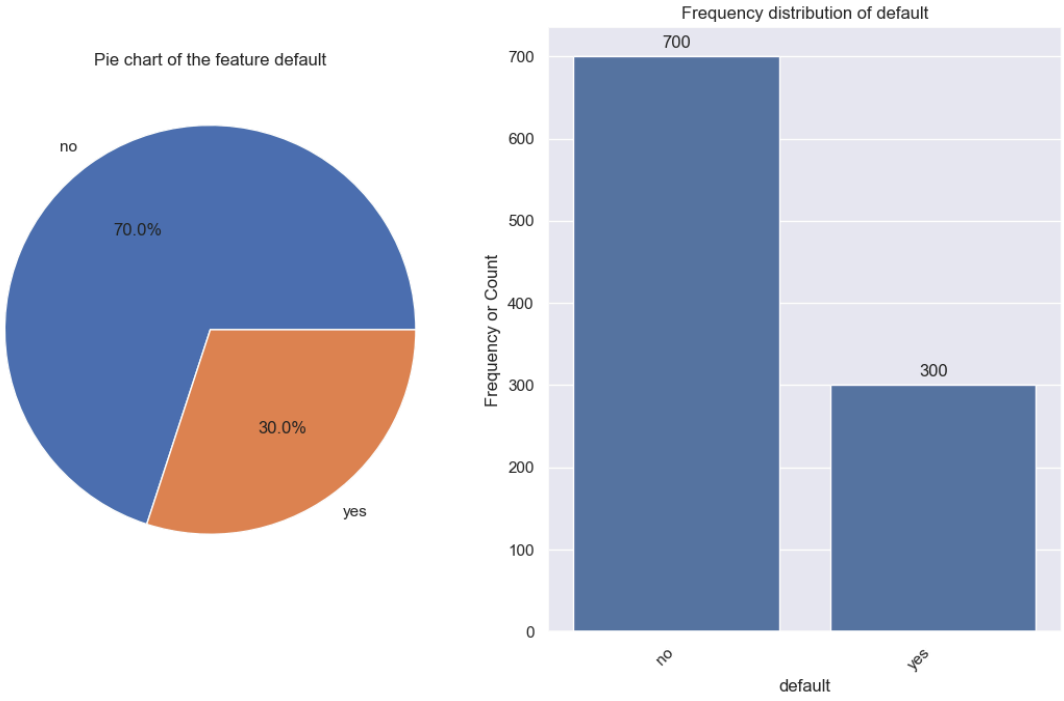

Fig-2: Countplot of the target or response default

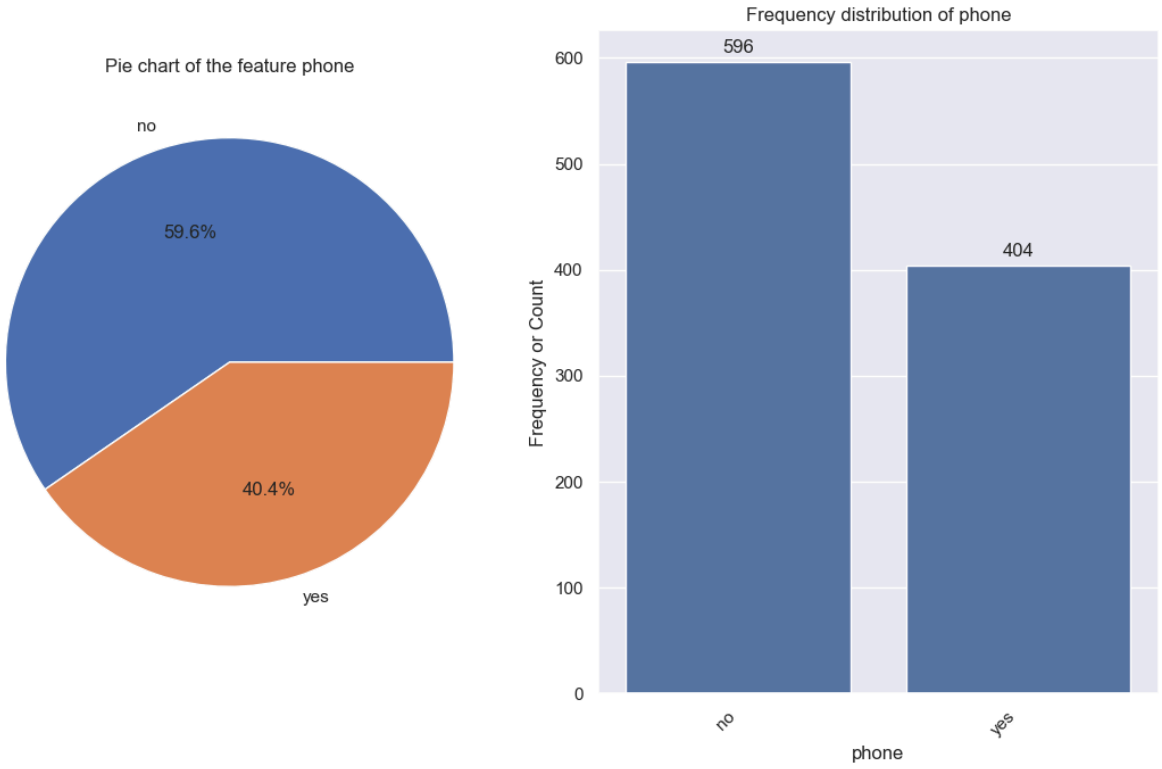

Fig-3: Countplot of predictor phone

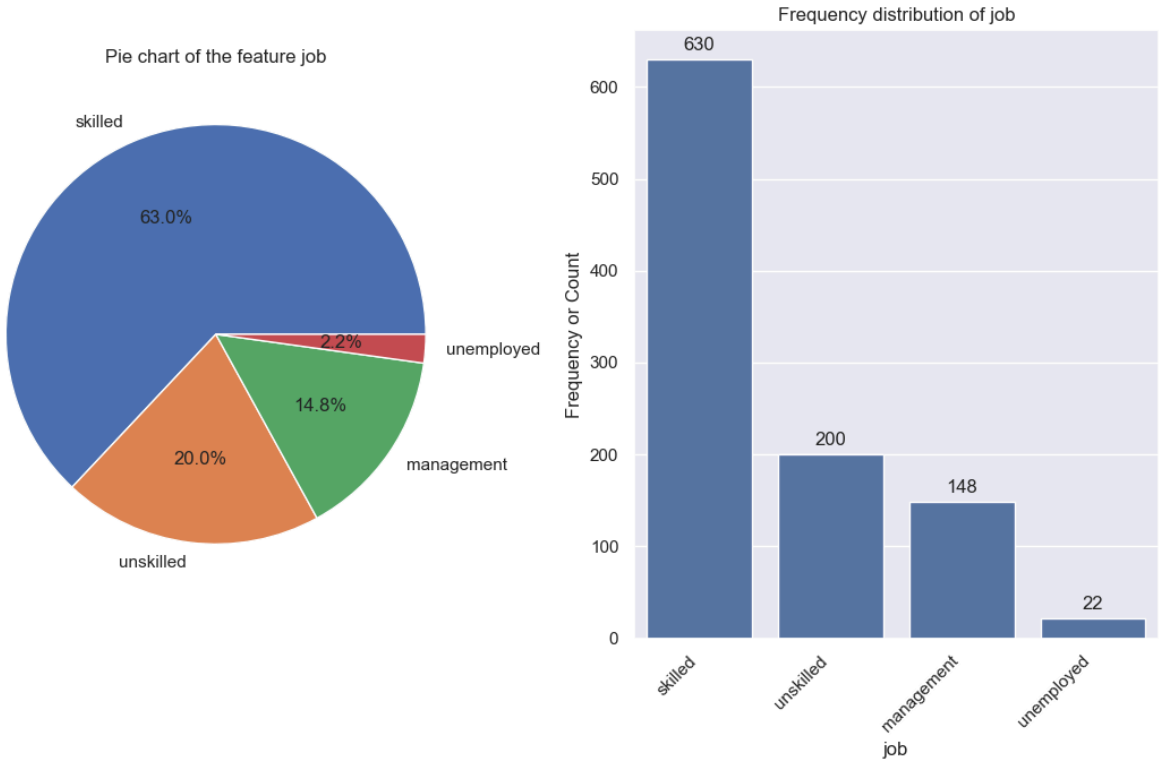

Fig-4: Countplot of predictor job

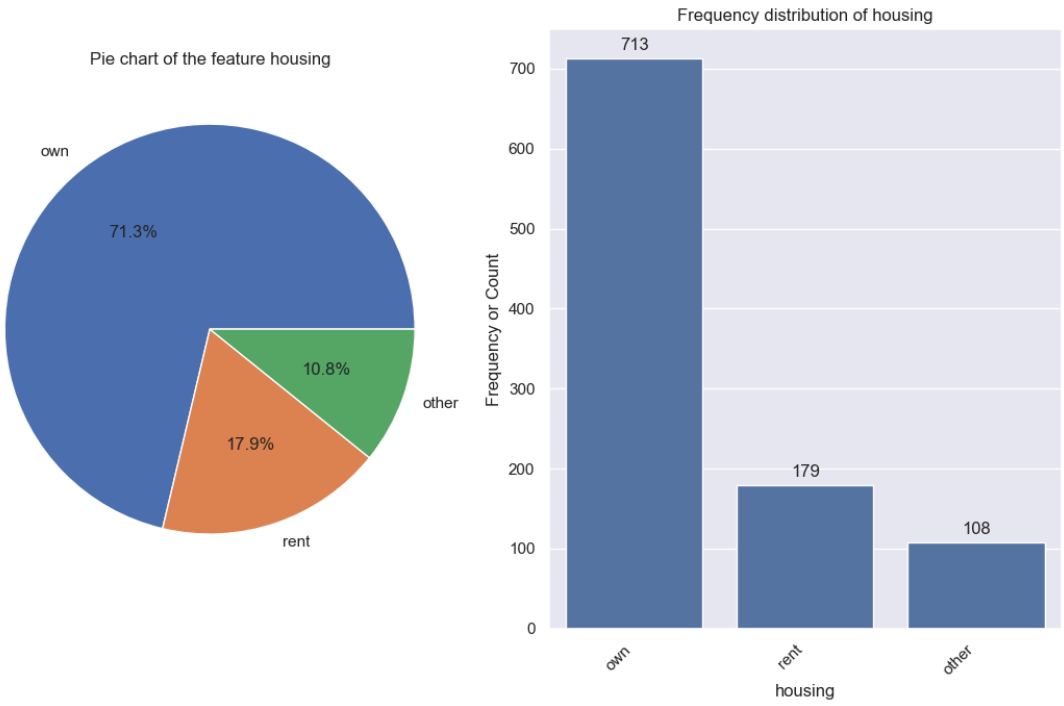

Fig-5: Countplot of predictor housing

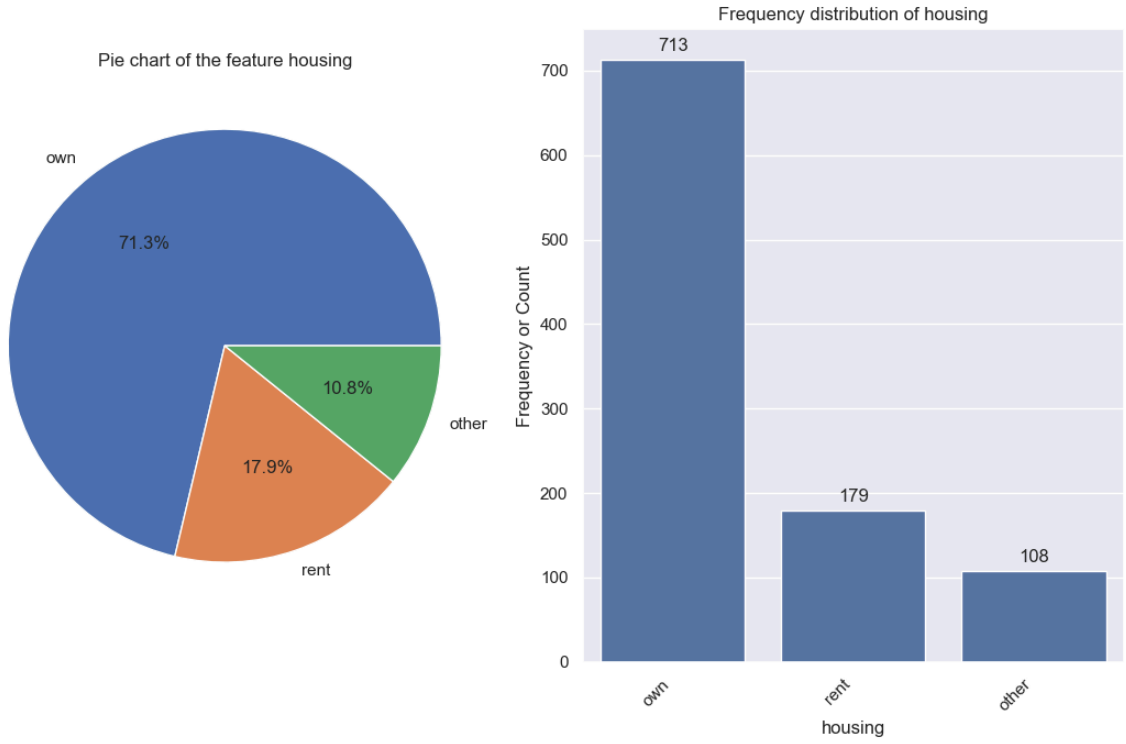

Fig-6: Countplot of predictor other_credit

There are 172 customers whose employment duration is less than 1 year, 339 customers whose employment duration is between 1 to 4 years, 174 customers whose employment duration is between 4 to 7 years, 253 customers whose employment duration is greater than 7 years and 62 customers who are unemployed. Large number of customers(603) have a savings balance less than 100 DM and few customers(63) have a savings balance between 500-1000 DM and only 48 customers have greater than 1000 DM. Moreover there are 183 customers with unknown savings balance and 103 customers with saving balance between 100 to 500 DM. The purpose column represents the purpose for which the loan was taken. There are two categories representing cars- one is named as 'car' whereas the other column is named as 'car0'. Both represent that the loan was taken to purchase a car but there seems to be a spelling mistake. Majority of customers(473) took loan to buy furniture/ appliances, whereas very few(22) took out loans for renovations. Around 349 customers(car + car0) took loan to buy a car and 97 customers took loan for business purpose but only 59 took loan for education purpose.

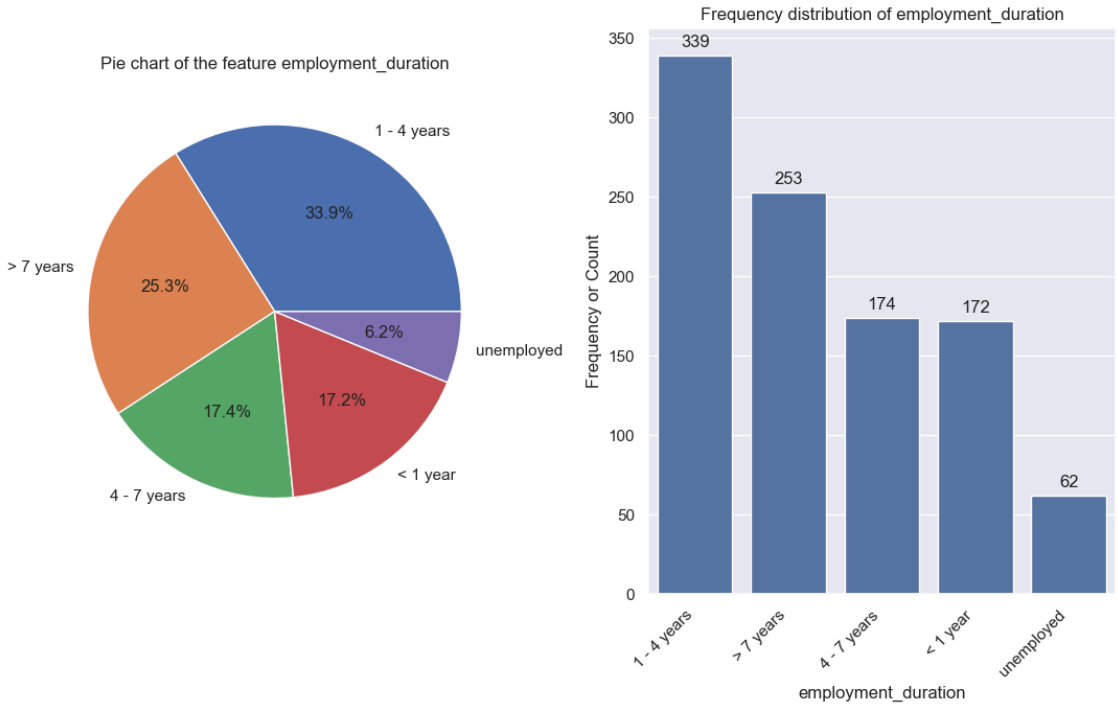

Fig-7: Countplot of predictor employment duration

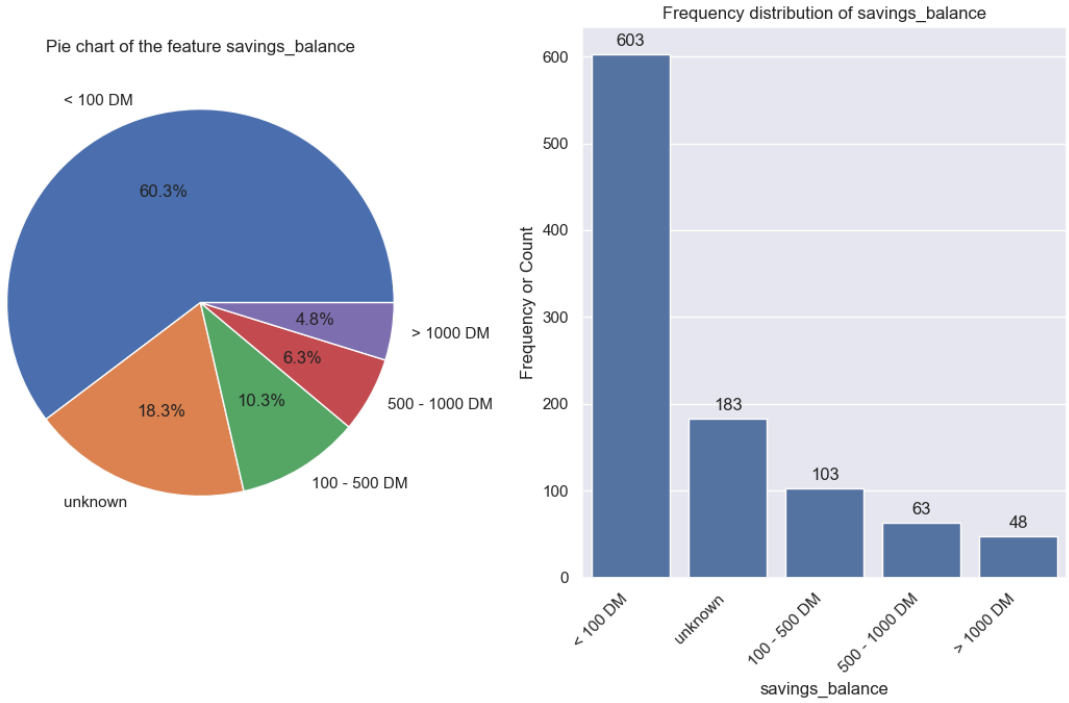

Fig-8: Countplot of predictor savings_balance

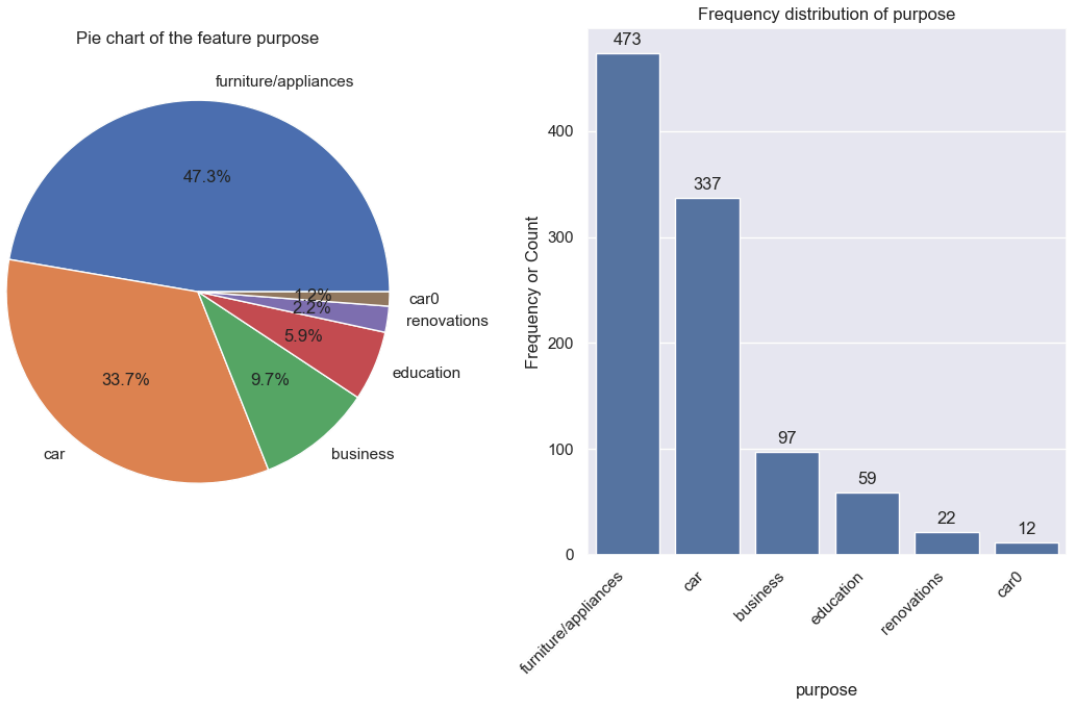

Fig-9: Countplot of predictor purpose

Majority of the customers(530) have a good credit history and only 40 have a perfect credit history. There are a significant number of customers(293) with critical credit history. Additionally, there are 88 customers with a poor credit history but only 49 with a very good credit history. There are 394 customers with unknown checkings balance, 274 customers with checking balance less than 0DM, 269 customers with checking balance between 1-200 DM and only 63 customers with checking balance greater than 200 DM.

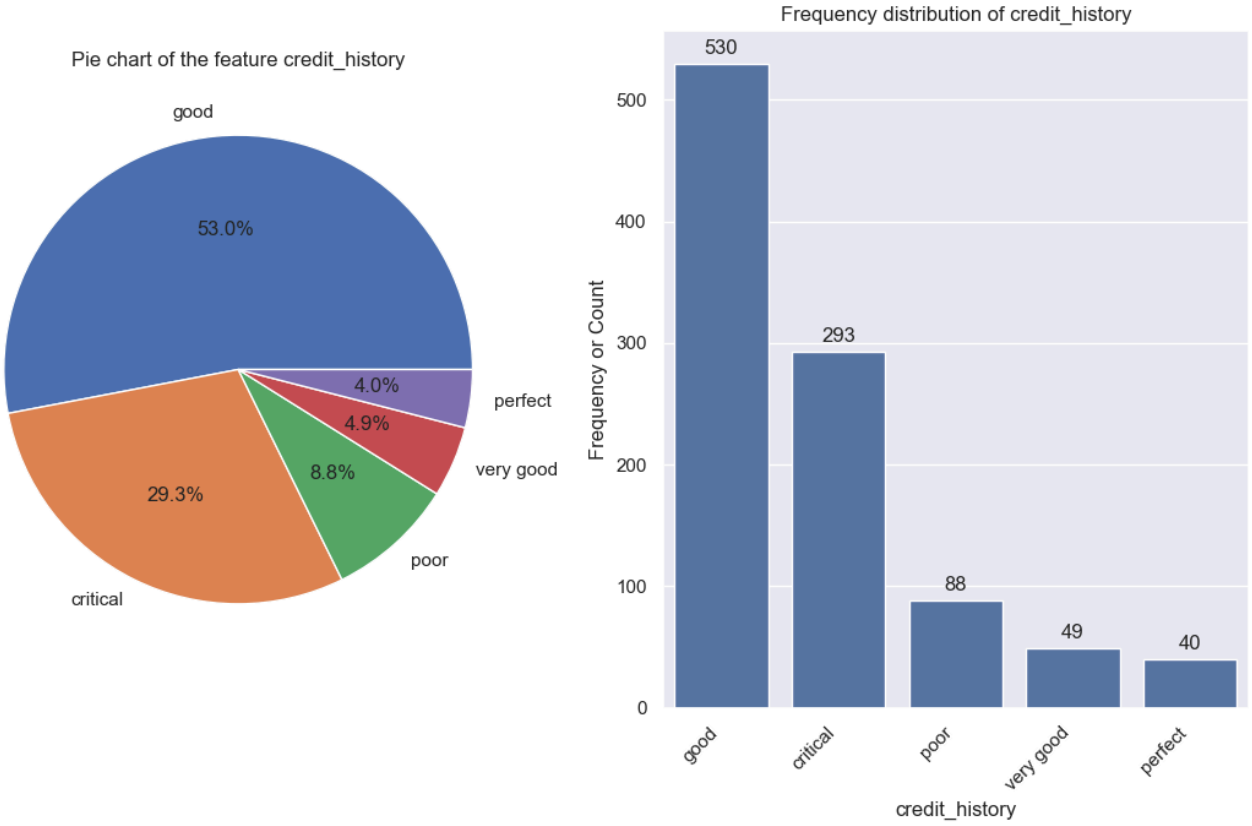

Fig-10: Countplot of predictor credit_history

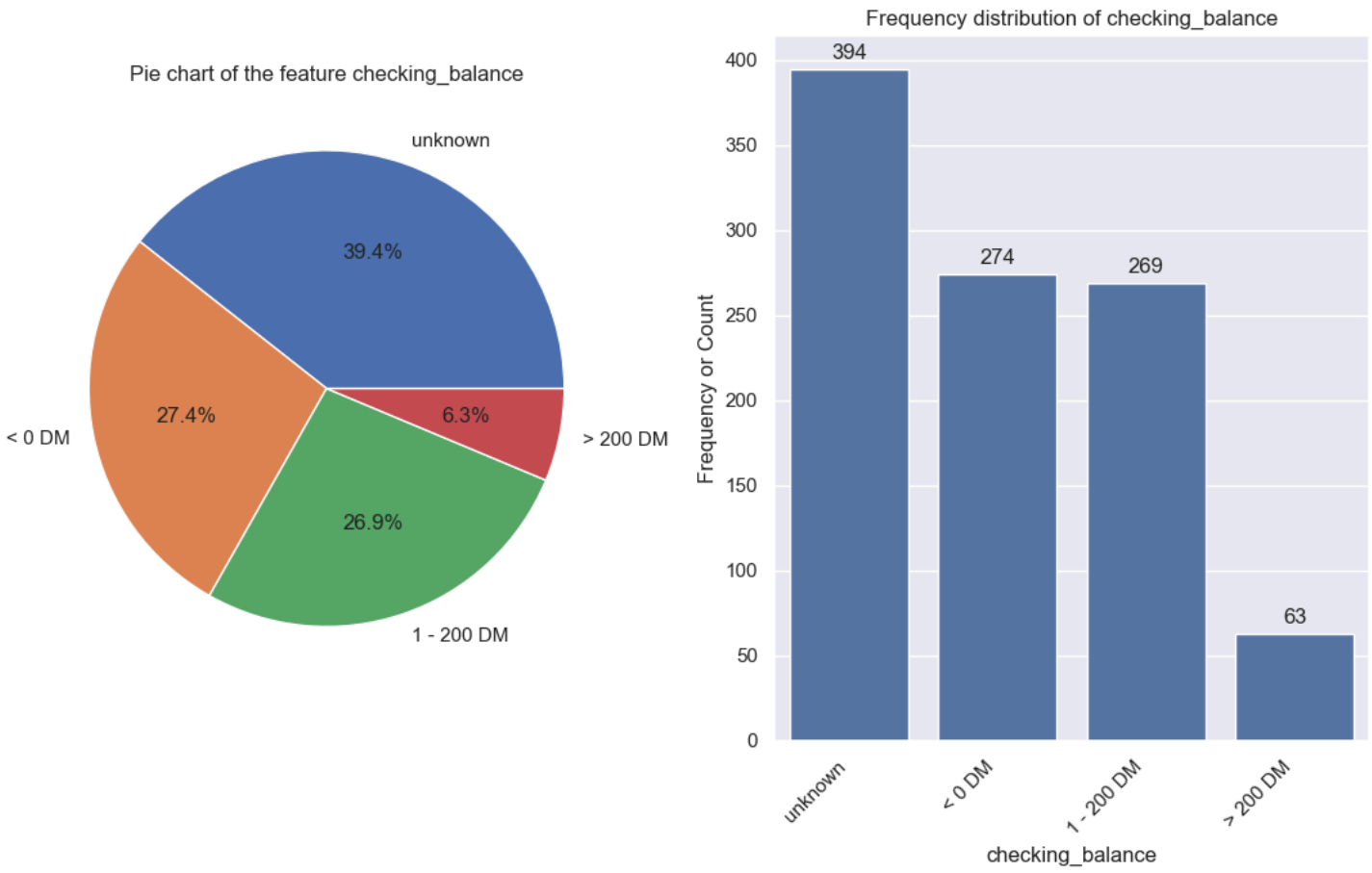

Fig-11: Countplot of predictor checking_balance

The above graphs illustrate the frequency distribution of each categorical variable whereas the graphs displayed below depict the category-wise percentage and category-wise number of defaulters and non-defaulters in each categorical variable or predictor.

In the below displayed category-wise plots, it is clear from the checking_balance graph that the percentage of defaulters increases as the checking_balance amount reduces. It is clear that customers belonging to the category '<0 DM' in the variable checking_balance have the highest number of defaulters whereas the customers belonging to the category 'unknown' have the least number of defaulters.

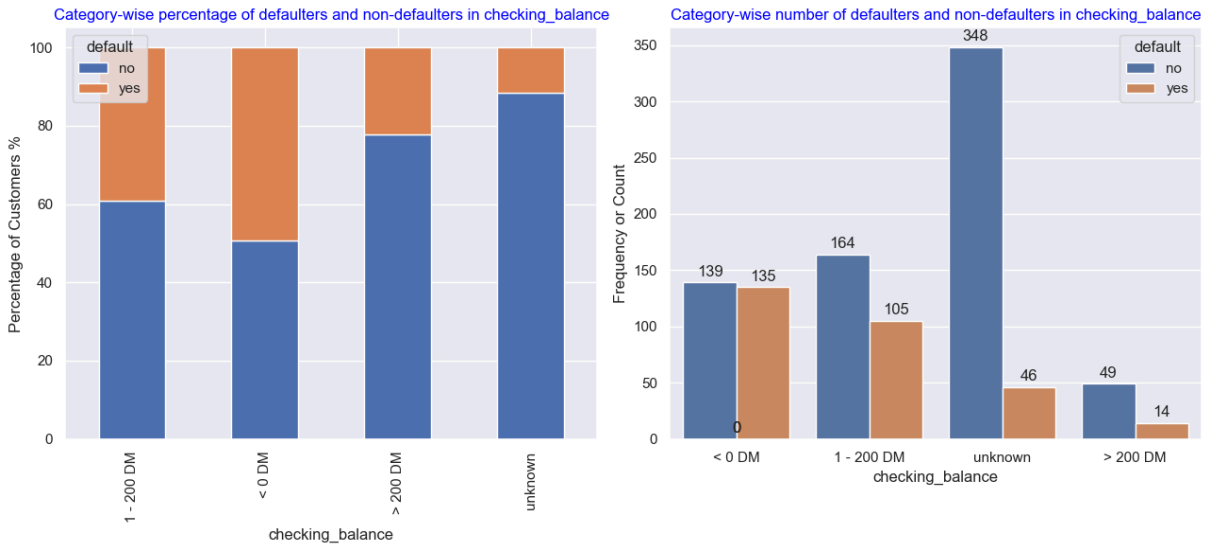

Fig-12: Category-wise plot of checking_balance

From the credit_history graph given below, it is surprising to see that the customers belonging to the categories 'perfect' and 'very good' default on their credit. This might have happened due to lack of adequate data about customers with 'perfect' and 'very good' credit history. Out of all the categories, the category 'good' has the highest number of non-defaulters(361) and defaulters(169). The category 'critical' has the lowest percentage of defaulters.Unemployed customers and customers whose employment duration is less than 1 year (<1 year) are more likely to default on their credit. Customers with more employment duration are less likely to default on their loan . Customers who took loan for the purpose of education and car(car + car0) are more likely to default.

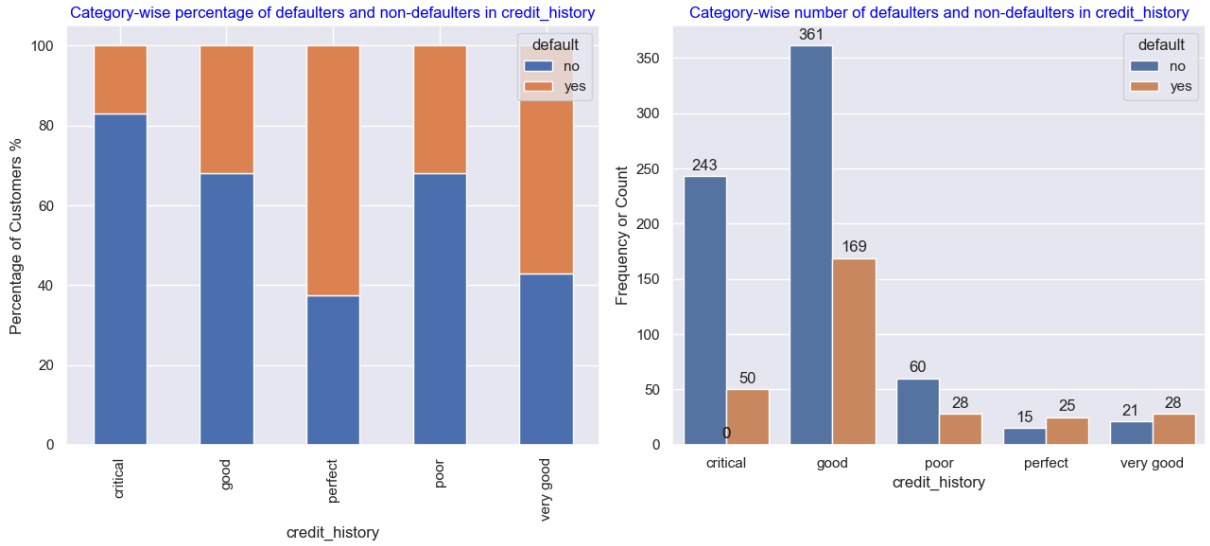

Fig-13: Category-wise plot of credit_history

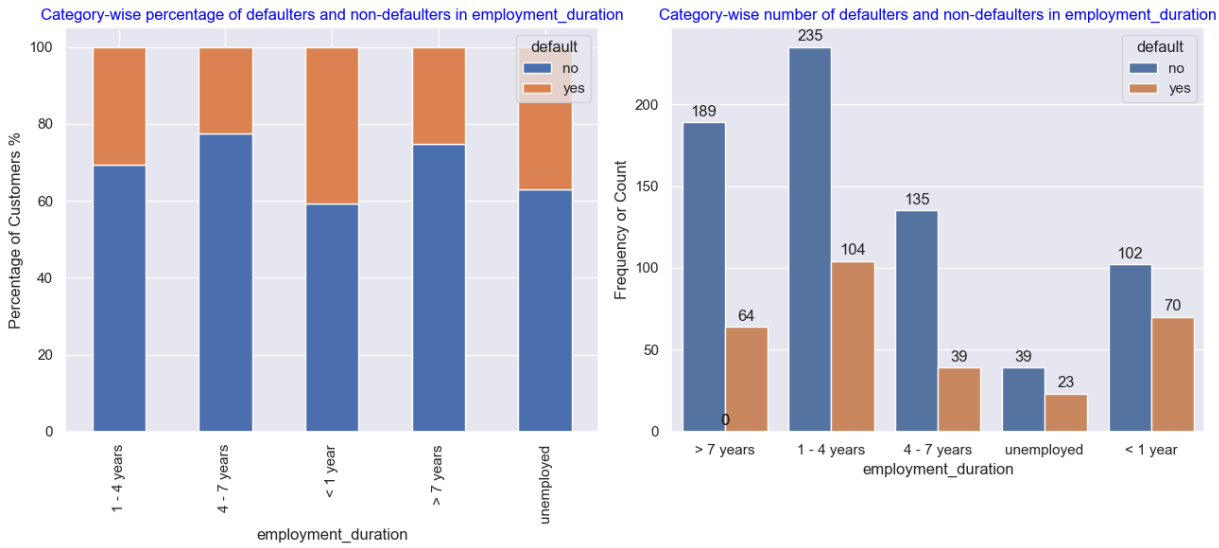

Fig-14: Category-wise plot of employment_duration

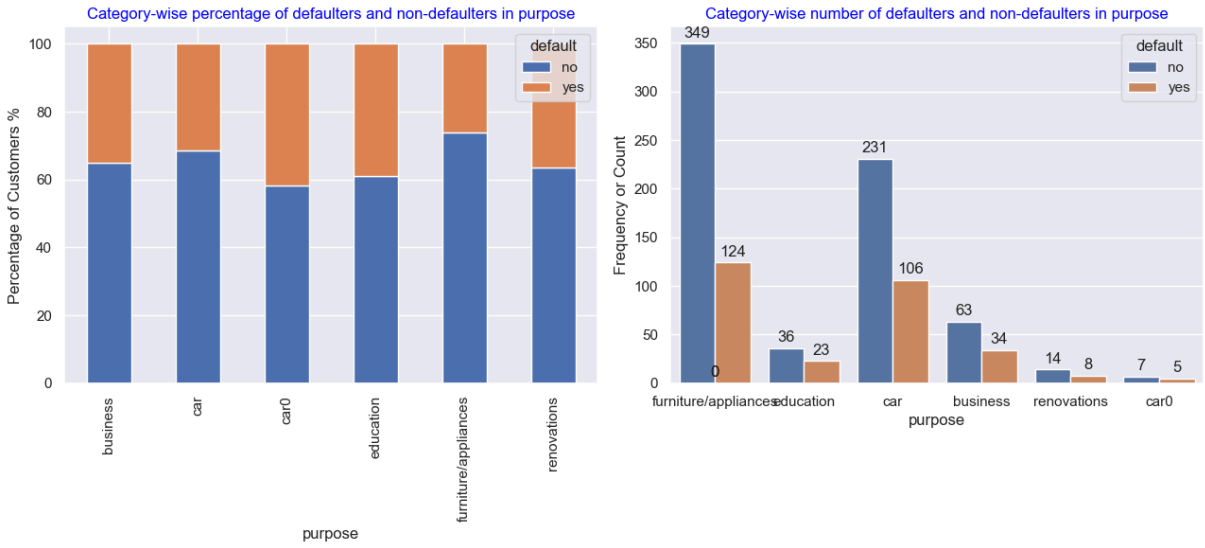

Fig-15: Category-wise plot of purpose

In the savings_balance figure, it is clear that the percentage of defaulters increases as the savings balance amount decreases. In addition to this, customers that own a house are less likely to default on their loan. Moreover, customers with no sources of other credit are less likely to default on their loan too. The percentage of defaulters in each of the categories of the predictors are approximately the same for the predictors phone and job.

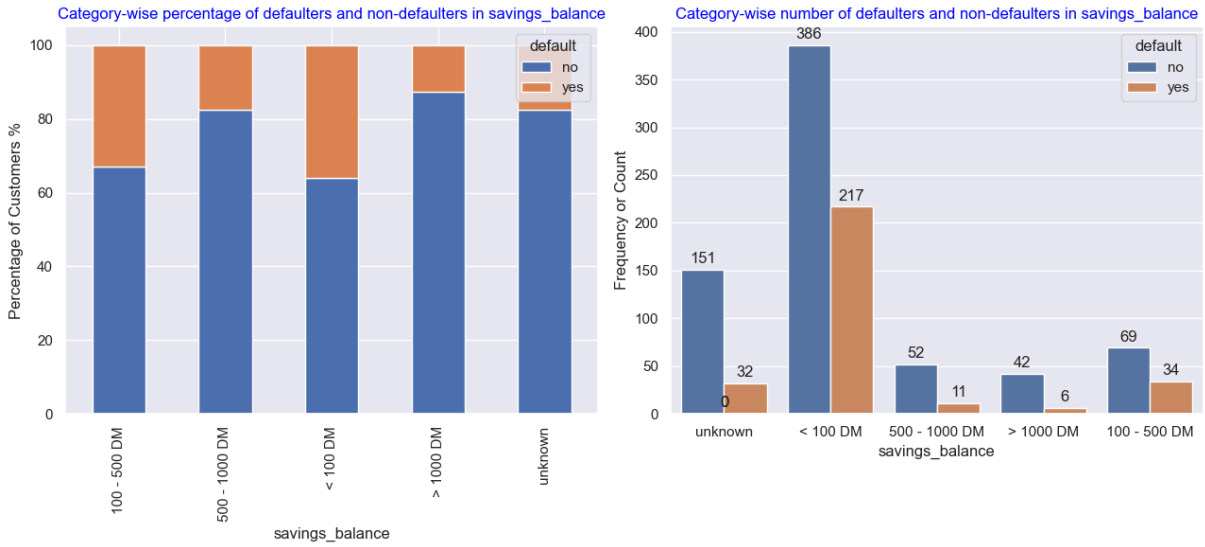

Fig-16: Category-wise plot of savings_balance

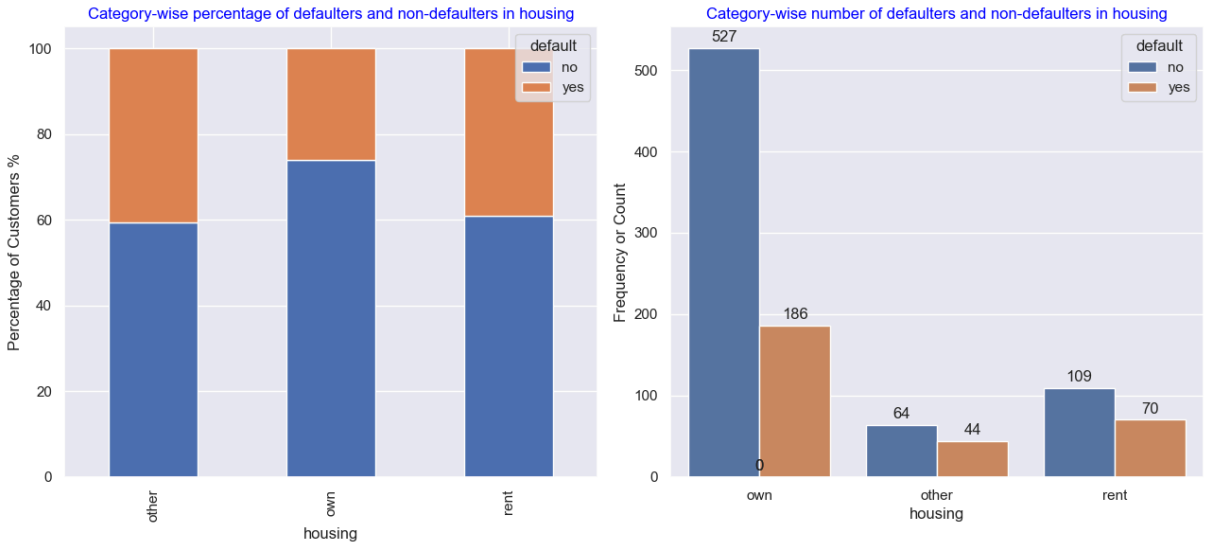

Fig-17: Category-wise plot of housing

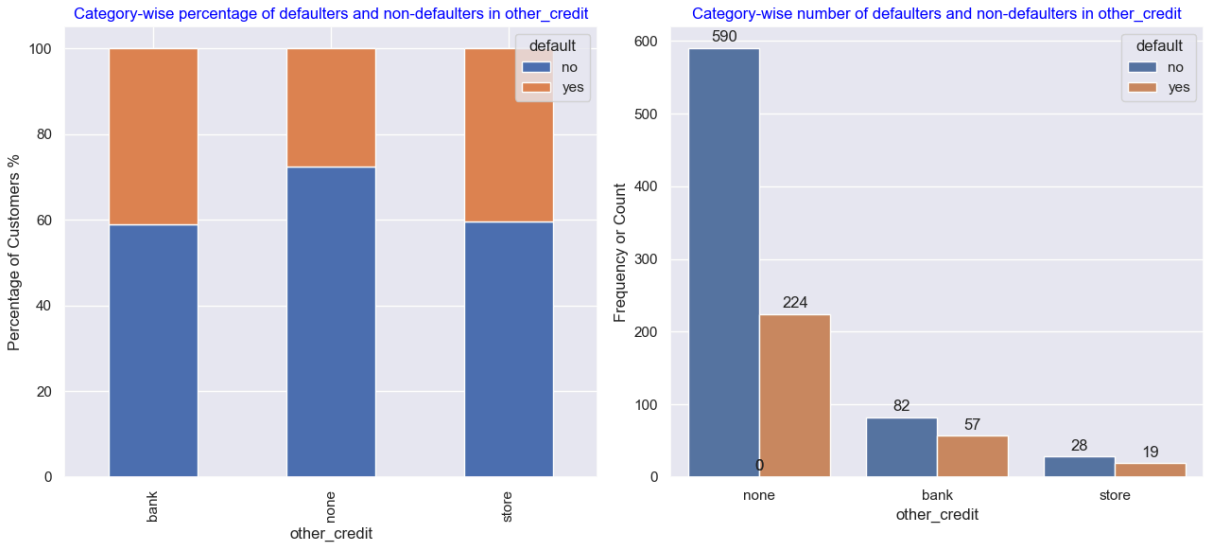

Fig-18: Category-wise plot of other_credit

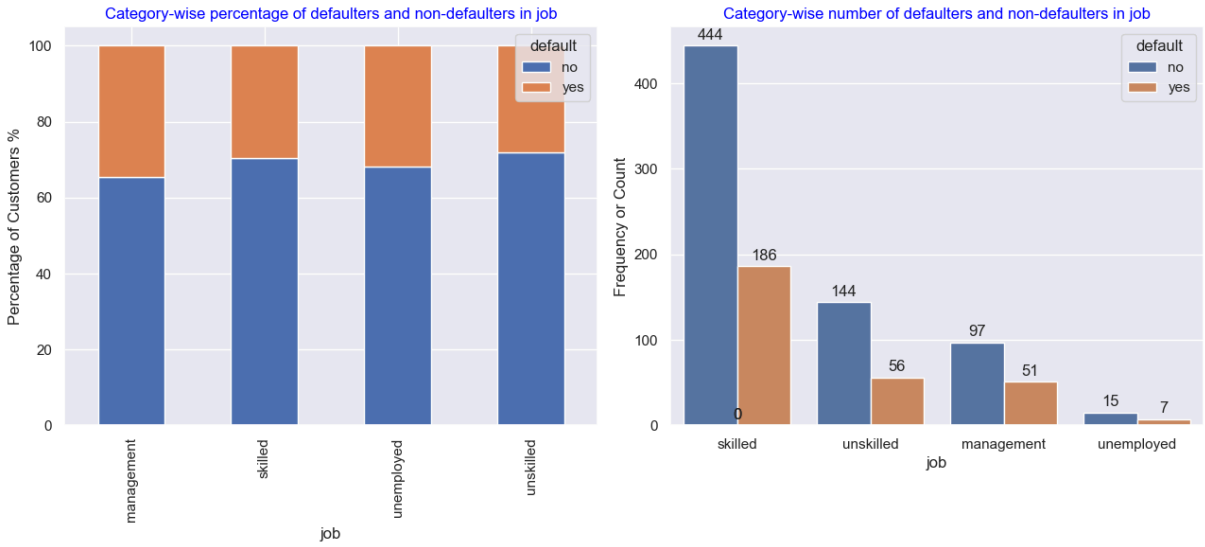

Fig-19: Category-wise plot of job

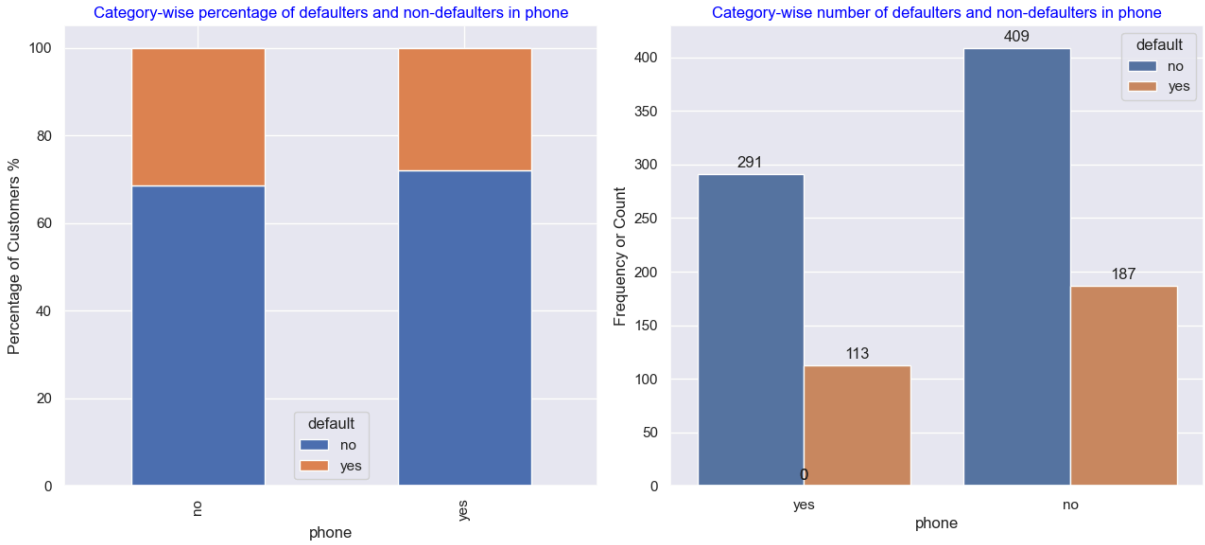

Fig-20: Category-wise plot of phone

Exploring Numerical Variables:

Numerical variables can be understood by plotting pair-plots and correlation matrix. Correlation matrix gives us a metric using which we can estimate the relationship between the variables. With the help of correlation value between two variables, we can determine if they are positively or negatively related to each other. The pairplots give us a visual depiction about how each variable is related to the other variables. From the correlation matrix and pairplot, it can be seen that the variables amount and months_loan_duration are moderately correlated with each other and moreover they are positively correlated. Remaining variables do not exhibit any correlation among them. The correlation coefficient between amount and months_loan_duration is 0.62 indicating they are positively correlated to each other.

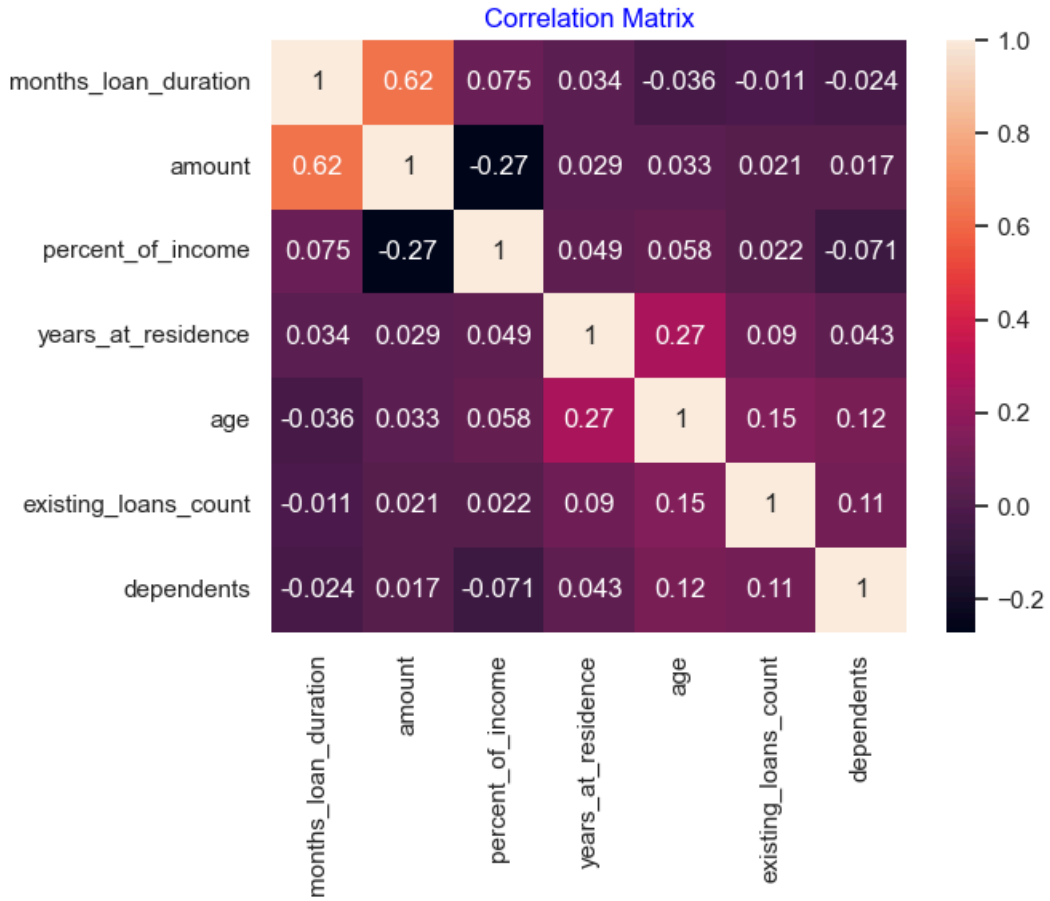

Fig-21:Correlation Matrix

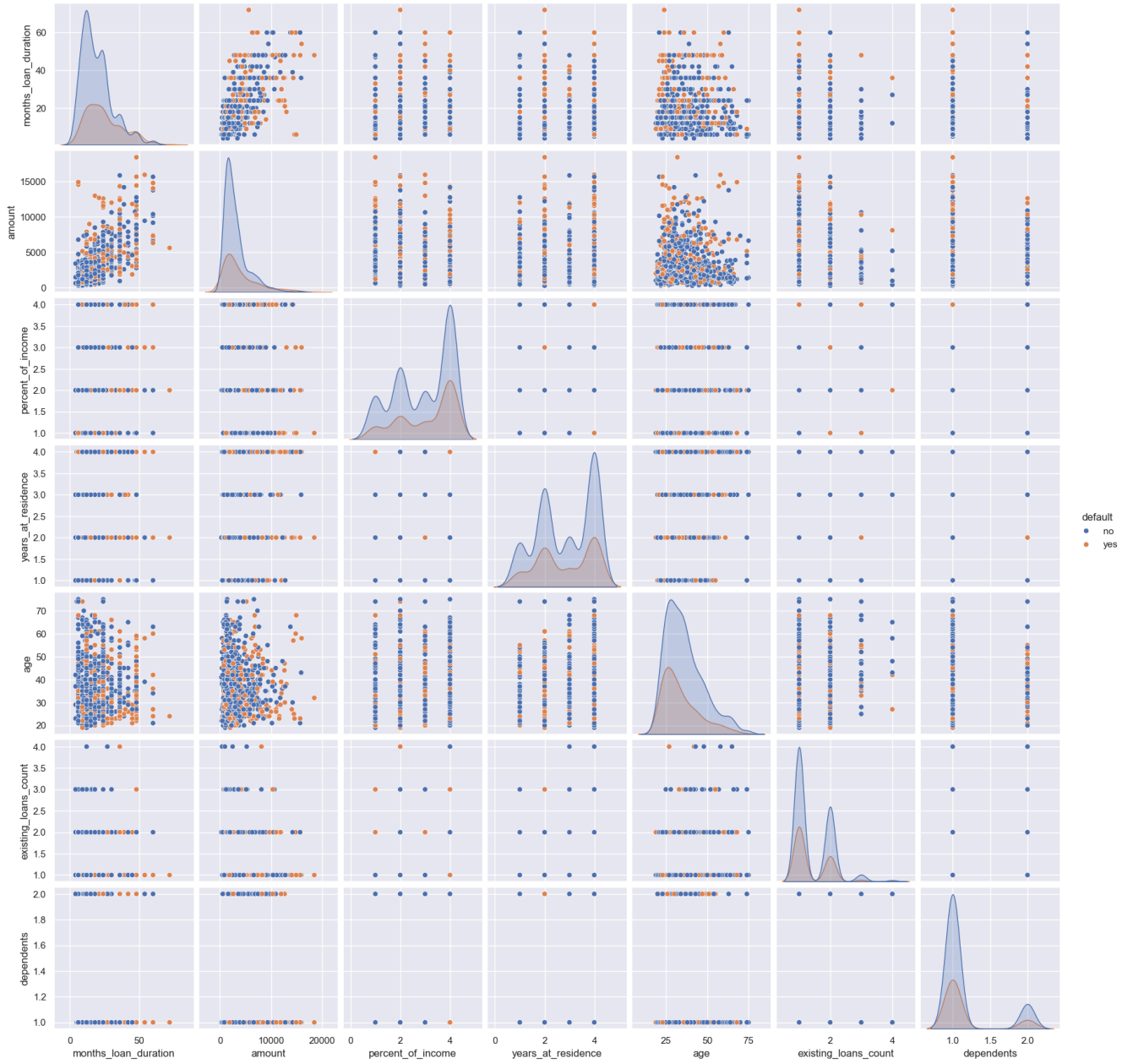

Fig-22:Pairplot between the variables

b. Pre-processing data for model training:

In the pre-processing step, firstly the categorical variable purpose has ‘car’ and ‘car0’ representing that the loan was taken to purchase a car. They both represent the same purpose but are misspelled. Hence, the category ‘car0’ in the predictor is renamed to ‘car’. The numerical variable amount has a large range from 250 to 18424, hence it is normalized using the MinMaxScaler() function. The remaining numerical variables did not undergo any transformations. The categorical variables contain both ordinal and nominal variables. The nominal variables in the dataset are checking_balance, purpose, savings_balance, job, other_credit, housing, phone and the target default. Actually, checking_balance and savings_balance should be ordinal variables but they both have category 'unknown' in large numbers. Hence, it cannot be deleted and cannot be given any order. So, these two variables are considered nominal variables in the pre-processing step. The ordinal variables in the dataset are credit_history and employment_duration. The categories in variables credit_history and employment_duration are ordered as 'critical' < 'poor' < 'good' < 'very good' < 'perfect' and 'unemployed' < '<1 year' < '1-4 years' < '4-7 years' < '> 7 years' respectively.

The nominal variables in the dataset are encoded using a one-hot encoder and the ordinal variables are encoded using an ordinal encoder. The numerical variable amount is normalized using the min-max scaling method. The entire pre-processing steps are performed using the column-transformer pipeline to simplify the transformation process. The target column default has only two classes: yes and no. The class yes is mapped to 1 indicating loan defaulters whereas no is mapped to 0 indicating non-defaulters. The above said pre-processing steps are performed after splitting the data into train and test data in a 70:30 ratio. This is done to avoid data leakage problems and this method resembles the actual and practical scenario where the new data is not seen by the model. While training the XGBoost model on the transformed data, it was throwing an error because of the presence of ‘<’ in some of the feature names. Hence, the required feature names were renamed in the transformed dataset obtained from pre-processing the raw, original dataset.

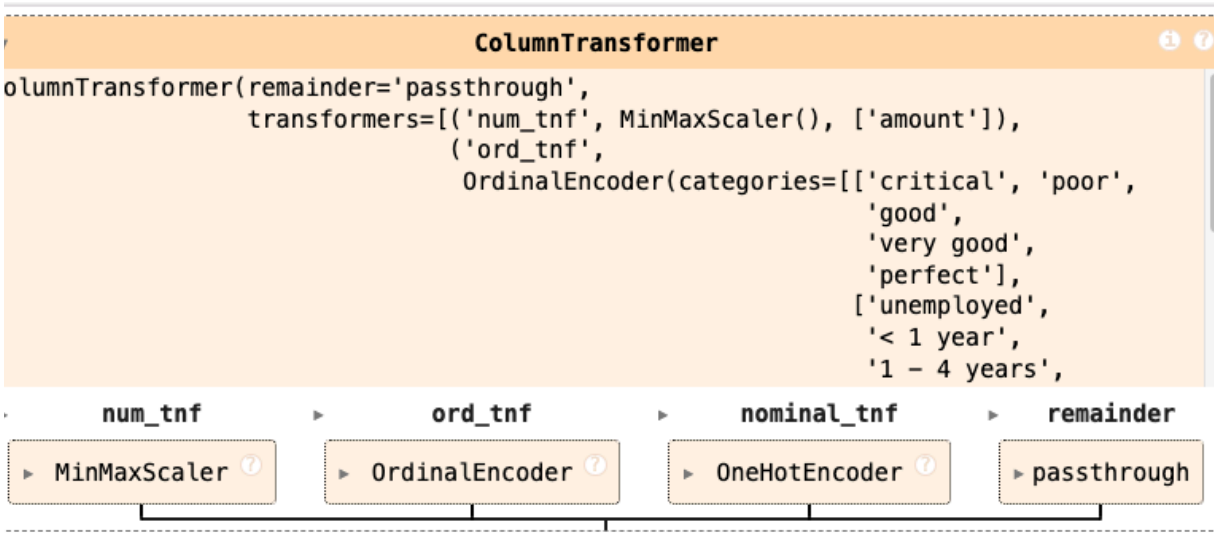

Fig-23: Pre-processing

c. Model Tuning and Training :

The best model for this task is found out by carrying out cross-validation and hyperparameter tuning on supervised classifiers such as Bagging, Random Forest, Gradient Boosting Machine(GBM), AdaBoost and XGBoost. Additionally, Catboost and LightGBM models were also trained on the dataset to build advanced classifiers. The cross-validation scores were calculated on the basis of 5-fold cross-validation and recall was used as the scoring metric. The models are fine-tuned using the grid search cross validation approach. In the grid-search cross-validation approach, the user specifies a list of parameter values specific to the model and the algorithm performs a cross validation task on each and every combination of parameter values to find the best parameter setting for the given model.

After finding the best parameter values for a given model, the model is once again trained on the training set using the best parameters and is evaluated on a separate test set to check its performance. The metrics and results are visualized using the custom VisualizeMetrics class defined in the notebook. The VisualizeMetrics class has three methods in it. The calculate_metrics()

method takes the model name, predictors and response as an input and computes the basic classification metrics such as recall, F1, accuracy, precision and AUC. The display_cm_roc() method takes the model name, predictors and response as arguments and returns the confusion matrix and ROC curve. The last method get_metrics() method is used to display the classification metrics of training set and test set, confusion matrix of the test set and the ROC curve of the test set. The class VisualizeMetrics was written to minimize the repetition of code. All the models are instantiated with random_state=96. Please use random_state=96 in model to reproduce the metrics.

3. Results:

Cross-Validation Scores:

```
Cross-Validation Performance scores:-

For Bagging: 0.381
For Random forest: 0.4
For GBM: 0.4619
For Adaboost: 0.4238
For Xgboost: 0.4714
```

```
Performance on training dataset:-

Recall for Bagging = 0.9333
Recall for Random forest = 1.0
Recall for GBM = 0.7524
Recall for Adaboost = 0.5429
Recall for Xgboost = 1.0
```

Fig-24: Cross-validation and Recall scores

The cross-validation scores and the recall values of the base models are shown in the above figure. The recall values on the training dataset for the bagging, random forest, GBM, AdaBoost and XGBoost models are 0.9333, 1.0, 0.7524, 0.5429 and 1 respectively. The cross-validation scores for the bagging, random forest, GBM, AdaBoost and XGBoost models are 0.381, 0.4, 0.4619, 0.4238 and 0.4714 respectively. The cross-validation scores are not up to the mark. They can be further improved by hyperparameter tuning.

a. Bagging hyperparameter Tuning and Model Training:

After tuning the parameters, the optimal value of the parameter 'estimator' is found to be 'DecisionTreeClassifier()' and the value of the parameter 'n_estimators' is found to be 95. These parameters resulted in a cv score of 0.4952. The following results are obtained when the model is once again trained on the training dataset using the optimal parameters and is evaluated on the test set. The recall, accuracy, AUC, precision and F1-score on the test dataset are 0.5,0.75,0.68,0.6 and 0.55 respectively whereas on the training set all the metrics are equal to 1.

```
Model used = BaggingClassifier(estimator=DecisionTreeClassifier(), n_estimators=95,
                 random_state=96)


Performance on Training data:

    Recall  Accuracy  AUC  Precision  F1-score
0      1.0       1.0  1.0        1.0       1.0
--------------------------------------------------

Performance on Test data:

    Recall  Accuracy   AUC  Precision  F1-score
0      0.5      0.75  0.68        0.6      0.55
--------------------------------------------------

Confusion Matrix and ROC Curve on test data:
```

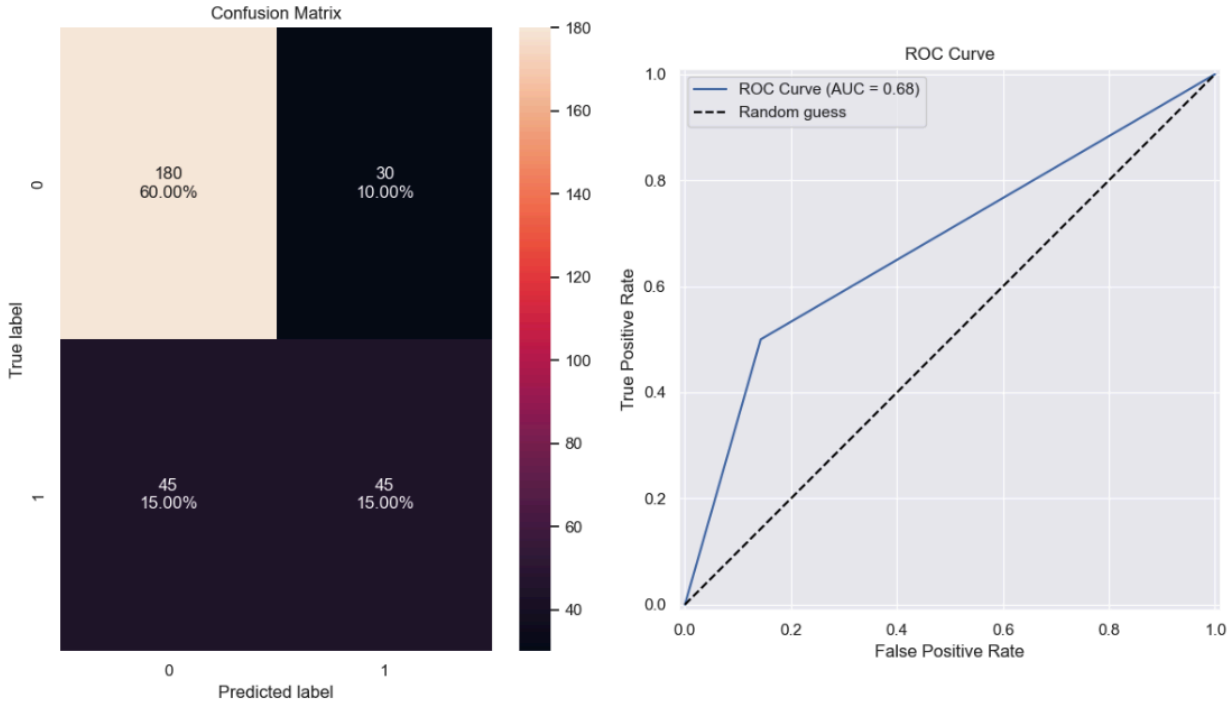

Fig-25: Bagging Results

b. Random Forest hyperparameter Tuning and Model Training:

The optimal values of the parameters are {'criterion': 'entropy', 'n_estimators': 15} with a cv score = 0.4. The metrics, confusion matrix and AOC curve obtained are:-

```
Model used = RandomForestClassifier(criterion='entropy', n_estimators=15, random_state=96)

Performance on Training data:

    Recall  Accuracy   AUC  Precision  F1-score
0    0.96      0.99  0.98        1.0      0.98
--------------------------------------------------

Performance on Test data:

    Recall  Accuracy   AUC  Precision  F1-score
0    0.42      0.75  0.65       0.61       0.5
--------------------------------------------------

Confusion Matrix and ROC Curve on test data:
```

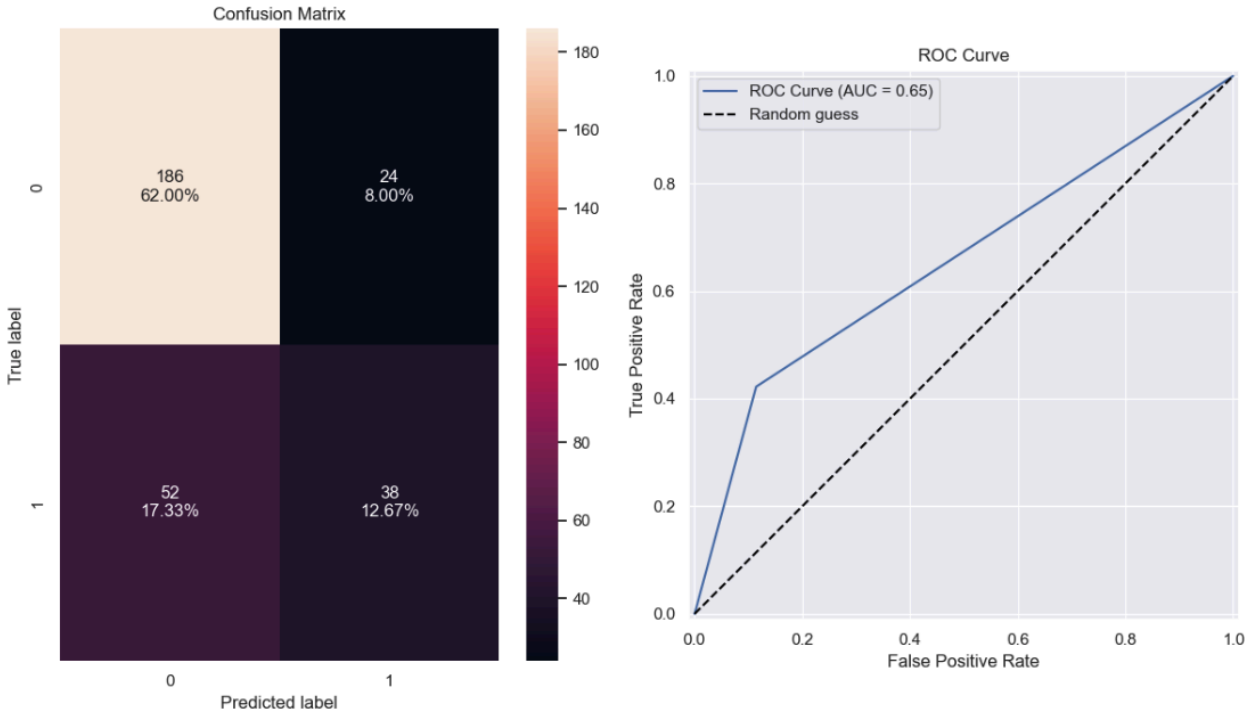

Fig-26: Random Forest Results

c. GBM hyperparameter Tuning and Model Training:

The optimal values of the parameters are {'learning_rate': 1, 'max_depth': 3, 'n_estimators': 70} with a cv score = 0.5286. The metrics, confusion matrix and AOC curve obtained are:-

```
Model used = GradientBoostingClassifier(learning_rate=1, n_estimators=70, random_state=96)

Performance on Training data:

    Recall  Accuracy  AUC  Precision  F1-score
0      1.0       1.0  1.0        1.0       1.0
--------------------------------------------------

Performance on Test data:

    Recall  Accuracy   AUC  Precision  F1-score
0     0.52      0.73  0.67       0.56      0.54
--------------------------------------------------

Confusion Matrix and ROC Curve on test data:
```

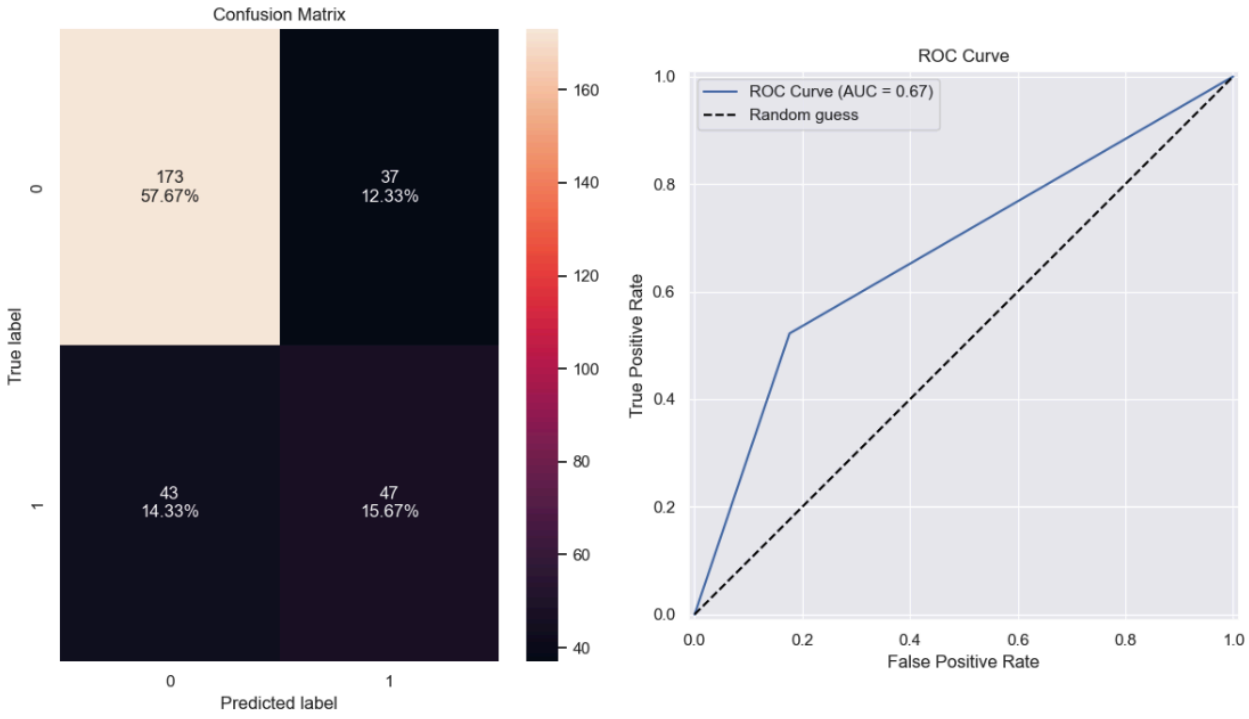

Fig-27: Gradient Boosting Machine Results

d. AdaBoost hyperparameter Tuning and Model Training:

The optimal values of the parameters are {'learning_rate': 2, 'n_estimators': 10} with a cv score = 0.8381. The metrics, confusion matrix and AOC curve obtained are:-

```
Model used = AdaBoostClassifier(algorithm='SAMME', learning_rate=2, n_estimators=10)

Performance on Training data:

    Recall  Accuracy   AUC  Precision  F1-score
0    0.84       0.6  0.67       0.42      0.56
--------------------------------------------------

Performance on Test data:

    Recall  Accuracy   AUC  Precision  F1-score
0    0.87      0.61  0.69       0.43      0.57
--------------------------------------------------

Confusion Matrix and ROC Curve on test data:
```

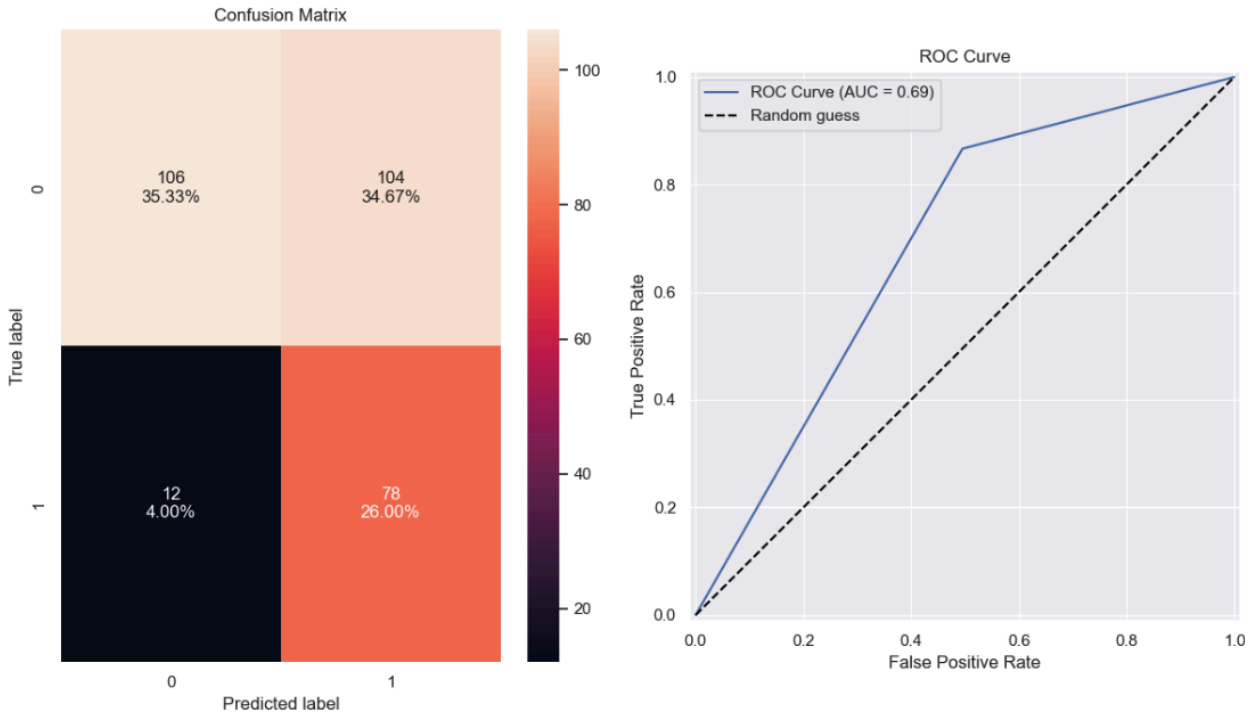

Fig-28: AdaBoost Results

e. XGBoost hyperparameter Tuning and Model Training:

The optimal values of the parameters are {'gamma': 0, 'learning_rate': 0.01, 'max_depth': 1, 'n_estimators': 10, 'scale_pos_weight': 5, 'subsample': 0.8} with a cv score = 1.0. The metrics, confusion matrix and AOC curve obtained are:-

```
Model used = XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=0, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.01, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=1, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=10, n_jobs=None,
              num_parallel_tree=None, random_state=None, ...)

Performance on Training data:

    Recall  Accuracy  AUC  Precision  F1-score
0     1.0       0.3  0.5        0.3      0.46
--------------------------------------------------

Performance on Test data:

    Recall  Accuracy  AUC  Precision  F1-score
0     1.0       0.3  0.5        0.3      0.46
--------------------------------------------------

Confusion Matrix and ROC Curve on test data:
```

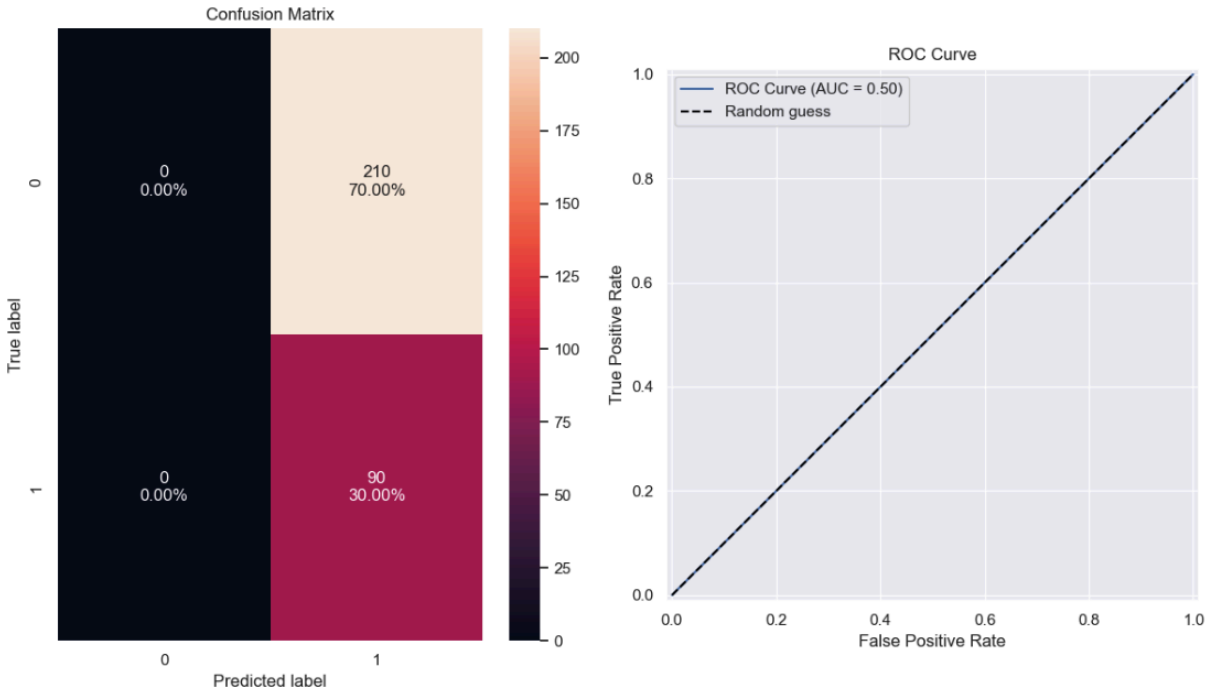

Fig-29: XGBoost Results

f. CatBoost Model Training:

The arguments of the CatBoost model used for training are random_state=96, verbose=0, iterations=500, learning_rate=0.15, depth=6, loss_function='MultiClass'. The metrics, confusion matrix and AOC curve obtained are:-

```
Model used = <catboost.core.CatBoostClassifier object at 0x1768ebd50>

Performance on Training data:

    Recall  Accuracy  AUC  Precision  F1-score
0      1.0       1.0  1.0        1.0       1.0
--------------------------------------------------

Performance on Test data:

    Recall  Accuracy   AUC  Precision  F1-score
0     0.57      0.77  0.71       0.64       0.6
--------------------------------------------------

Confusion Matrix and ROC Curve on test data:
```

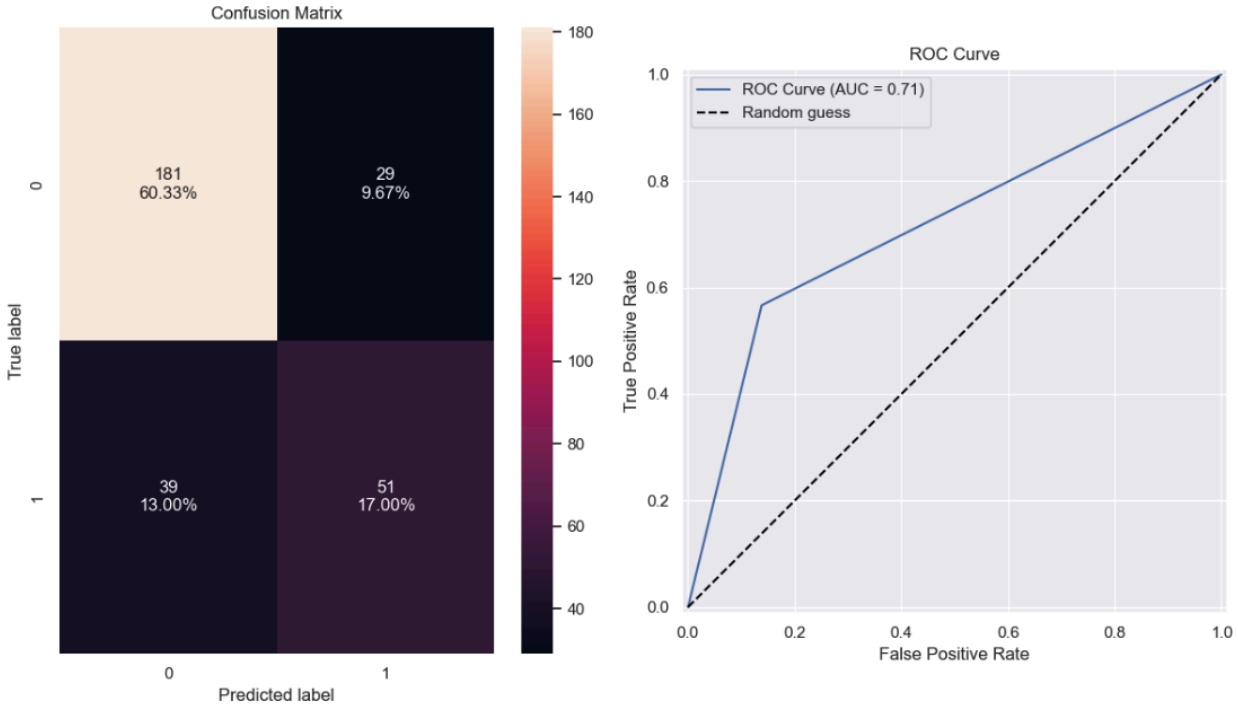

Fig-30: CatBoost Results

g. LightGBM Model Training:

The arguments of the LightGBM model used for training are {'num_leaves':25, 'objective':'binary','metric':'binary_logloss'}. The metrics, confusion matrix and AOC curve obtained are:-

```
Model used = LGBMClassifier(metric='binary_logloss', num_leaves=25, objective='binary',
               random_state=96)


Performance on Training data:

    Recall  Accuracy  AUC  Precision  F1-score
0      1.0       1.0  1.0        1.0       1.0
--------------------------------------------------


Performance on Test data:

    Recall  Accuracy   AUC  Precision  F1-score
0     0.56      0.77  0.71       0.63      0.59
--------------------------------------------------


Confusion Matrix and ROC Curve on test data:
```

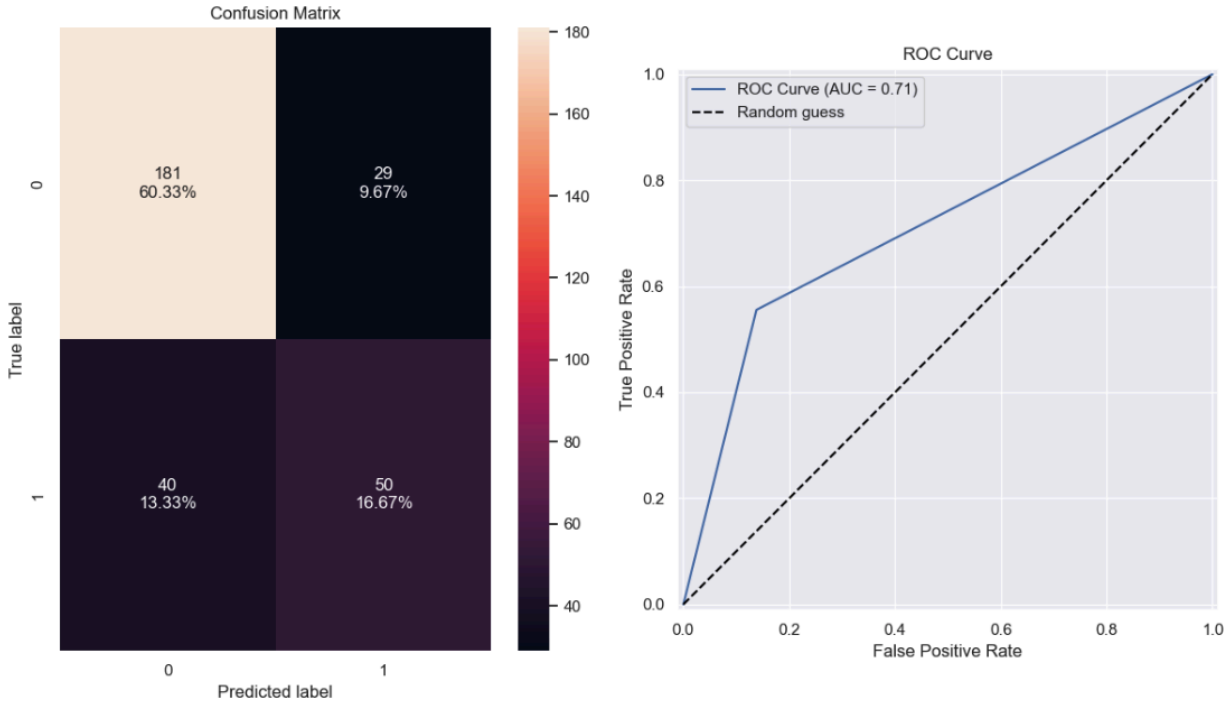

Fig-31: LightGBM Results

Consolidated Results Table:-

The consolidated results on both training and test sets are represented below.

Training Performance:

	Bagging Classifier	Random Forest Classifier	GBM Classifier	AdaBoost Classifier	XGBoost Classifier	CatBoost Classifier	LGBM Classifier
Recall	1.0	0.96	1.0	0.84	1.00	1.0	1.0
Accuracy	1.0	0.99	1.0	0.60	0.30	1.0	1.0
AUC	1.0	0.98	1.0	0.67	0.50	1.0	1.0
Precision	1.0	1.00	1.0	0.42	0.30	1.0	1.0
F1-score	1.0	0.98	1.0	0.56	0.46	1.0	1.0

Test Performance:

	Bagging Classifier	Random Forest Classifier	GBM Classifier	AdaBoost Classifier	XGBoost Classifier	CatBoost Classifier	LGBM Classifier
Recall	0.50	0.42	0.52	0.87	1.00	0.57	0.56
Accuracy	0.75	0.75	0.73	0.61	0.30	0.77	0.77
AUC	0.68	0.65	0.67	0.69	0.50	0.71	0.71
Precision	0.60	0.61	0.56	0.43	0.30	0.64	0.63
F1-score	0.55	0.50	0.54	0.57	0.46	0.60	0.59

Fig-32: Results Table

4. Discussion:

a. Major Findings and Interpretation of results:

From the Fig-24 it can be seen that the recall values for Random Forest and XGBoost are 1. This indicates that there are no false negatives. It is essential to avoid false negatives in this problem because predicting a customer will not default when he actually defaults incurs a great loss to the bank. Hence, our machine learning model should reduce false negatives as much as possible. Bagging model follows next in performance with a recall value of 0.9333. It indicates that the model correctly identified 93.33% of the customers who actually default. GBM has a recall of 0.7524 indicating that it has correctly identified 75.24% of the customers who actually default. Adaboost on the other hand was able to identify only 54.29% of the

customers who actually default. But the cross-validation scores are not up to the mark. They can be improved further by hyperparameter tuning.

From the Fig-25, it is clear that the Bagging classifier model has a recall value on test data is 0.5, that is the model is able to correctly identify 50% of the customers who actually default in the test data. AUC = 0.68 indicates that the model is better than the one obtained by random guessing(AUC=0.5). The metrics on training data indicate that the model is overfitting the training data. In the Fig-26, it can be seen that the recall value for random forest model on test data is 0.42 that is the model can correctly identify 42% of the customers who actually default in the test data. AUC=0.65 indicates that the model is better than that obtained by random guessing. The metrics obtained on training data indicate that the model is overfitting. Performance of the random forest classifier is worse than the bagging classifier.

In the Fig-27, it can be observed that the recall value for GBM model on test data is 0.52 which indicates that the GBM model is correctly predicting only 52% of the customers who actually default in the test data. GBM's AUC=0.67 indicates that this model is better than the random guess model. This model shows better performance compared to the above discussed models. From Fig-28, it is clear that the recall value for AdaBoost on test data is 0.87, that is the model is able to correctly predict 87% of the customers who actually default in the test data. Adaboost's AUC=0.69 indicating that it is better than the random guess model. This model is not overfitting and has similar accuracies on both train(0.61) and test(0.6) datasets but it is unable to predict more customers who do not actually default. The performance of AdaBoost classifier is better than the above discussed models in terms of predicting the customers that actually default.

From the Fig-29, it is surprising to see that the XGBoost model has a recall 1 on both training and test datasets. XGBoost model is only learning the customers that default but it is not the customers that do not. Its AUC is 0.5 which is same as that of random guess. XGBoost has an accuracy of 0.3 because of its inability to predict the non-defaulters. This XGBoost model

predicts every customer as a potential defaulter. It cannot distinguish between defaulters and non-defaulters. In the Fig-30, it can be observed that the Catboost model has a recall value of 0.57 on test data indicating that the model can correctly predict 57% of the customers that actually default in the test data. The LightGBM model shown in Fig-31 has a recall of 0.56 on test data indicating that it can correctly predict 56% of the customers that actually default on their loan in the test data. Both Catboost and LightGBM have AUC=0.71 which indicates that they are better than the random guess model.

Finally after analyzing all the models, AdaBoost was found suitable for the task as it has a high recall value of 0.87 indicating it can correctly predict 87% of the actual loan defaulters. In the present context, the bank needs a model that has high recall value as it is essential to avoid false negatives in this problem because predicting a customer will not default when he actually defaults incurs a great loss to the bank. Additionally, the AdaBoost model has an accuracy of 61% and a precision of 43% on the test data too. Even though XGBoost's recall value(1.0) is greater than AdaBoost's(0.87), XGBoost is not suitable for this task because of its inability to detect any non-defaulters in the dataset. The XGBoost model predicts all the customers as defaulters which is undesirable as the model needs to predict non-defaulters too. Therefore, AdaBoost is the best suited model for predicting customers that default on their loan. It is followed by CatBoost, LGBM and GBM classifiers.

b. Limitations and Future Improvements:

The given bank dataset has 700 non-defaulters and 300 defaulters indicating that the data is biased towards on-defaulters. This class imbalance issue can be resolved by collecting more data on customers that defaulted earlier. In addition to this the other limitation in the dataset is that the predictors such as checkings_balance, savings_balance contain the category 'unknown' in large numbers which forced the analysis to consider them as nominal variables instead of ordinal variables. This issue can be resolved by

extracting accurate and more precise data of the customers. The performance can be improved by gathering data on more customers.

The limitation on the model is that all the parameters of a model are not found by exhaustively searching through a grid consisting of all the possible parameter values. Some of the models can be fine-tuned exhaustively on all the model's parameters to get better performance on the test data. Bagging classifiers can be tuned to improve the recall that is the ability to predict the customers that actually default. Additionally, AdaBoost classifier can be tuned more to improve the precision inorder to reduce the false positives. Randomized search cross-validation approach can be used to find the best parameter values efficiently and quickly.

5. Conclusions:

Firstly, the problem statement and the context are analyzed and understood thoroughly and clearly. Then exploratory data analysis is carried on the dataset in a step-by-step manner to check for missing, duplicate and null values. Later in the EDA itself, the distributions of categorical and numerical variables are computed to find out insights from the data. From the correlation matrix it is found that the correlation coefficient between the variable amount and months_loan_duration is 0.62 indicating the presence of a positive and moderately strong correlation between them. Later pre-processing is performed on categorical and numerical variables to prepare them for model training. After training all the models, AdaBoost was found to perform better than all the other models because of high recall, good accuracy and decent precision values on the test dataset. Finally, the top 3 important features of the dataset for AdaBoost model are amount, months_loan_duration and credit_history.