AML Final Project Report

Predicting Loan Defaulters in Banking Sector using ML Algorithms

Prepared By

Group 03

Anagha Bhole - AXB210061

Surabhi Chavan – SSC200012

Shashanka S Ranade – SSR200013

Srishti Vinaik - SXV210063

under the supervision of

Prof. Zixuan Meng

Table of Contents

1. EXECUTIVE SUMMARY

1.1 Introduction

Banks are the major source of funding in every economy. They provide wealth management such that its functioning is maintained all time. All the transactions in the form of deposits and lending are continuously taking place, and therefore lending activities need better monitoring. As banks cannot be sure whether a particular customer would default on his/her loan, therefore they collect information from the customers in relation to its lending policies and procedures and sanction loans. But the problem arises when the loan is not paid. Therefore, to save banks from the risk of loan defaulters, we are using the Machine learning Classification algorithm to classify if any new borrower is likely to default or not. We have a huge dataset with many different features like loan_limit, loan_type, rate_of_interest, upfront charges, Credit_Score, etc. which are considered by banks while approving loans to its customers. In our dataset we have 'Status' as the target variable which is binary (if the borrower will default on the loan or not).

1.2 Objective

We are concerned about how accurately our classification models could predict if any new borrower is likely to default or not and therefore develop a robust ML Model for future predictions.

1.3 Research

To answer the questions, we need to analyze the role of each attribute in the dataset. By performing various visualization and preprocessing techniques, obtain a dataset with no null values. We also need to find relations between all the attributes and the target variable. Then by running different classification models, how does the performance vary for the target variable across all models. Lastly, it is necessary to know which models predict bank defaulters more accurately. After all the analysis, we will conclude what we observe from the outputs and recommendations on how to improve the model for better predictions.

2. DATA DESCRIPTION

The dataset is taken from Kaggle.com. There are 34 attributes in the dataset that covers most of the features that banks usually consider for analyzing whether to approve the loan of their bank's customers. The number of observations contained in the dataset is 148670. Based on all these features, the target variable is binary as we are using a supervised learning-classification algorithm to classify if any new borrower is likely to default.

There are 21 categorical variables and 13 numerical variables. Below are features present in our dataset:

```
(148670, 34)
Index(['ID', 'year', 'loan_limit', 'Gender', 'approv_in_adv', 'loan_type',
       'loan_purpose', 'Credit_Worthiness', 'open_credit',
       'business_or_commercial', 'loan_amount', 'rate_of_interest',
       'Interest_rate_spread', 'Upfront_charges', 'term', 'Neg_ammortization',
       'interest_only', 'lump_sum_payment', 'property_value',
       'construction_type', 'occupancy_type', 'Secured_by', 'total_units',
       'income', 'credit_type', 'Credit_Score', 'co-applicant_credit_type',
       'age', 'submission_of_application', 'LTV', 'Region', 'Security_Type',
       'Status', 'dtir1'],
      dtype='object')
```

ID	year	loan_limit	Gender	approv_in_adv	loan_type	loan_purpose	Credit_Worthiness	open_credit	business_or_commercial	...	credit_type
24890	2019	cf	Sex Not Available	nopre	type1	p1	l1	nopc	nob/c	...	EXP
24891	2019	cf	Male	nopre	type2	p1	l1	nopc	b/c	...	EQUI
24892	2019	cf	Male	pre	type1	p1	l1	nopc	nob/c	...	EXP

credit_type	Credit_Score	co-applicant_credit_type	age	submission_of_application	LTV	Region	Security_Type	Status	dtir1
EXP	758	CIB	25-34	to_inst	98.728814	south	direct	1	45.0
EQUI	552	EXP	55-64	to_inst	NaN	North	direct	1	NaN
EXP	834	CIB	35-44	to_inst	80.019685	south	direct	0	46.0

3. DATA EXPLORATION AND PREPROCESSING

3.1 Exploratory Data Analysis

After exploring the number of rows and columns of the dataset,we analyzed the dataset.

3.1.1 Checking for null values in the data

To get rid of the null values, we began distinguishing each of the variables and imputed values with the help of the scikit-learn package using a simple imputer. We imputed the mean value of observations present in the respective feature for the missing numerical variables. The web imputed the most frequent category of words present in the respective variable for categorical variables. We dropped the missing value records of age and term as they had just 241 missing values.

```
data.isnull().sum()
```

```
ID                            0
year                          0
loan_limit                 3344
Gender                        0
approv_in_adv               908
loan_type                     0
loan_purpose                134
Credit_Worthiness             0
open_credit                   0
business_or_commercial        0
loan_amount                   0
rate_of_interest          36439
Interest_rate_spread      36639
Upfront_charges           39642
term                         41
Neg_ammortization           121
interest_only                 0
lump_sum_payment              0
property_value            15098
construction_type             0
occupancy_type                0
Secured_by                    0
total_units                   0
income                     9150
credit_type                   0
Credit_Score                  0
co-applicant_credit_type      0
age                         200
submission_of_application   200
LTV                       15098
Region                        0
Security_Type                 0
Status                        0
dtir1                     24121
dtype: int64
```

3.1.2 Checking for duplicate records

We checked the data for duplicate records, and there were no duplicate records in the data.

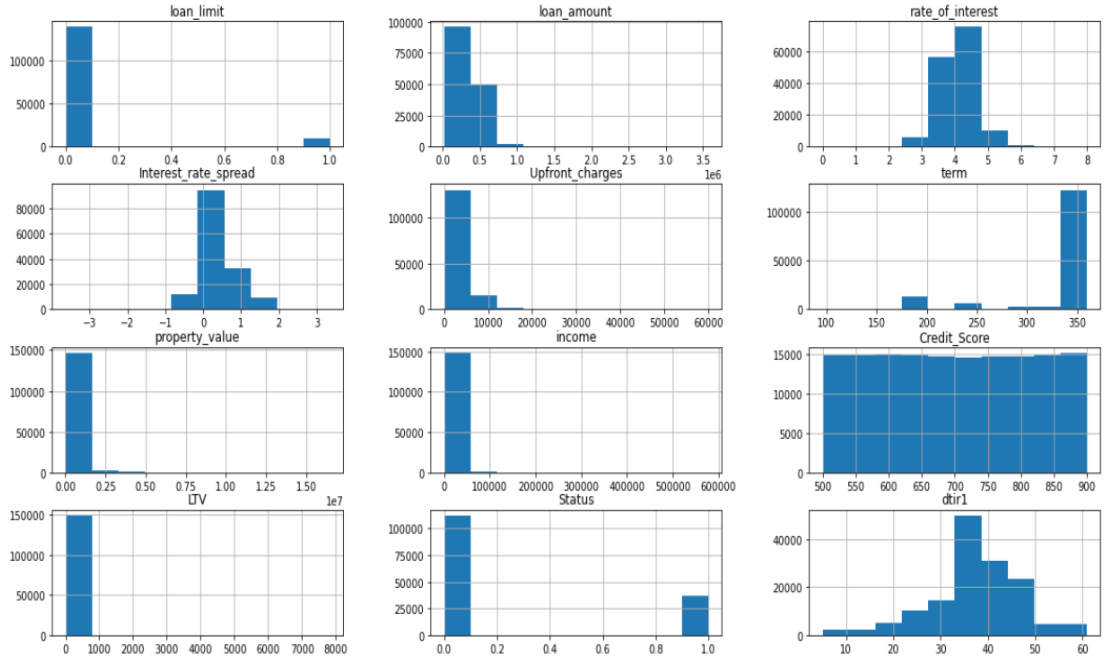
```
In [17]: data.shape
Out[17]: (148429, 32)

In [18]: # drop duplicates
         data= data.drop_duplicates()

In [19]: data.shape
Out[19]: (148429, 32)
```

3.1.3 Exploring each feature by plotting histograms

```
In [34]: data.hist()
         plt.rcParams["figure.figsize"] = (250,10)
```



3.2 Data Preprocessing

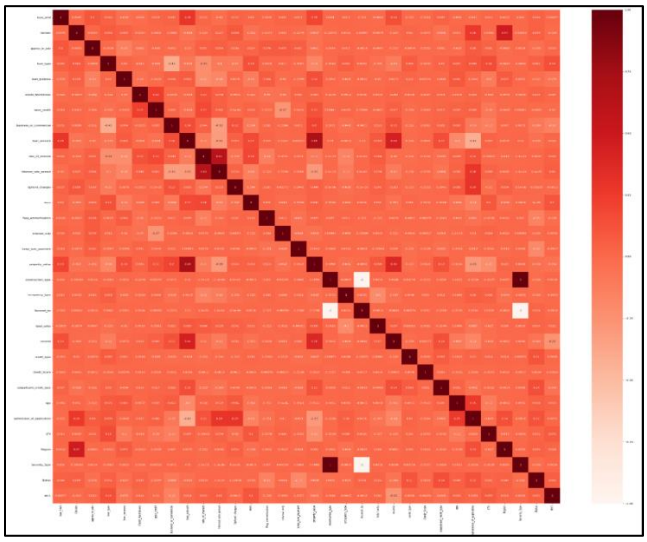
3.2.1 Encoding categorical data

We performed label encoding using Label Encoder from Scikit learn library to deal with categorical variables.

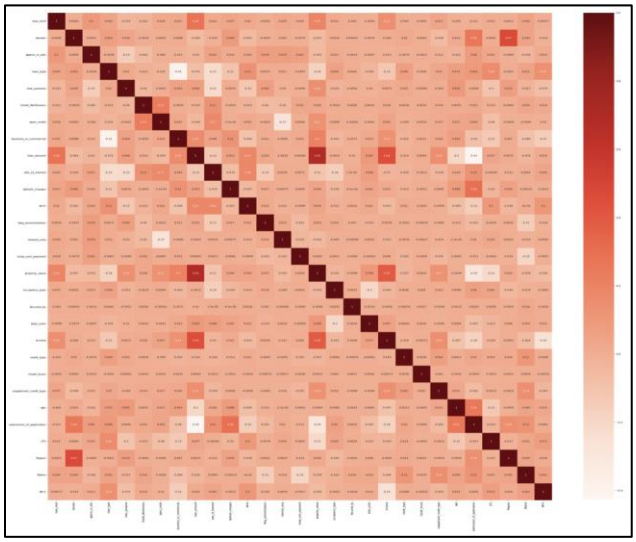
3.2.2 Checking for multicollinearity

Using Pearson's correlation, we tried to understand the collinearity between the explanatory variables. We created a heat map where darker colors signified more collinearity between the explanatory variables and the lighter colors signified less collinearity. Three variables were highly correlated, so we dropped 2 of those variables from the dataset.

Before dropping variables:



After dropping the highly correlated variables:



Before running the models, we split data by randomly selecting observations from the whole dataset as a training set and a test set.

4. APPROACHES

4.1 Approach 1: Running models with all the variables

After dropping variables with multicollinearity, we ran the model directly using the 32 variables. This gave us an accuracy of 100% for the decision tree, random forest, and XGBoost.

4.2 Approach 2: Lasso regression for feature selection

We ran Lasso regression for feature selection of the variables. We got the following results:

```
 The Best Parameter: {'model__alpha': 0.01}
[ 0.01274752  0.00285002 -0.00077532  0.         -0.          0.00279805
 -0.         -0.02118905  0.         -0.         -0.          0.
 -0.05610807 -0.         -0.07070102 -0.         -0.00089517  0.00050244
  0.0030504  -0.01398297  0.03074372  0.          0.05121445  0.
  0.03551916  0.          0.0015348   0.00986332]
[0.01274752 0.00285002 0.00077532 0.         0.         0.00279805
 0.         0.02118905 0.         0.         0.         0.
 0.05610807 0.         0.07070102 0.         0.00089517 0.00050244
 0.0030504  0.01398297 0.03074372 0.         0.05121445 0.
 0.03551916 0.         0.0015348  0.00986332]

 The Features Selected: ['loan_limit' 'Gender' 'approv_in_adv' 'Credit_Worthiness'
 'business_or_commercial' 'Neg_ammortization' 'lump_sum_payment'
 'occupancy_type' 'Secured_by' 'total_units' 'income' 'credit_type'
 'coapplicant_credit_type' 'submission_of_application' 'Region' 'dtir1']

 The Features that can be ignored: ['loan_type' 'loan_purpose' 'open_credit' 'loan_amount' 'rate_of_interest'
 'Upfront_charges' 'term' 'interest_only' 'property_value' 'Credit_Score'
 'age' 'LTV']
```

Here, we can see that Lasso gave some variables like “Credit score,” which should logically be a good predictor of the target variables, which is incorrect.

4.3 Approach 3: Dropping the problematic variable

We checked the rules for the decision tree and found that we get a pure split on the rate of interest, which is highly unlikely as we had multiple attributes in our data. Therefore, we dropped the complex variables and thus got good results for the models.

5. MODELS & PERFORMANCE EVALUATION

5.1 Model and performance metrics

After splitting the dataset into training and test set, we ran multiple classification models like Logistic Regression, Naïve Bayes, KNN, Decision Tree, Random Forest, and XgBoost Classifier on the data.

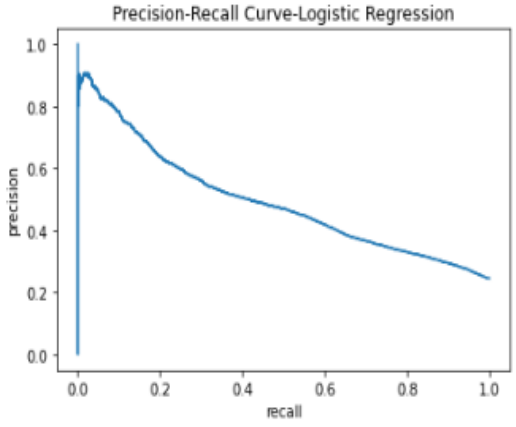
We used different performance metrics like Accuracy, F1 Score, Precision score, Recall Score, ROC curve, Precision-Recall curve, and AUC.

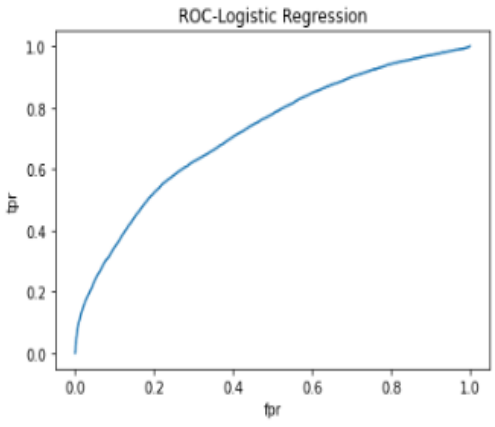
5.2 Performance evaluation results

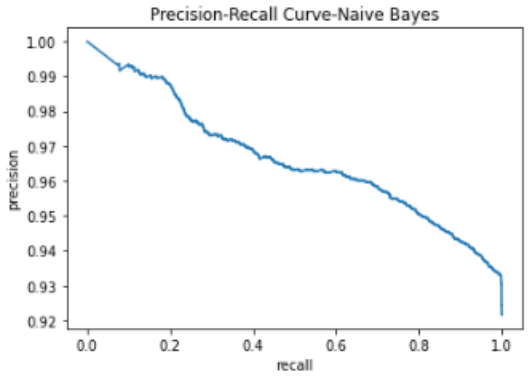
Below is the performance evaluation using six different metric

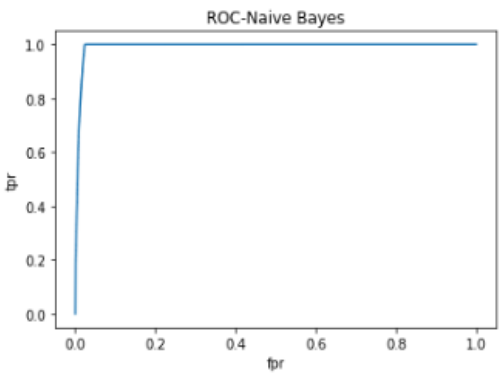
Classification algorithm	**Accuracy**	**F1 score**	**Precision score**	**Recall score**	**AUC**
Logistic Regression	77.59%	0.289	0.655	0.185	0.7200
Naive Bayes	82.23%	0.440	0.974	0.284	0.993
KNN	84.17%	0.652	0.708	0.605	0.835
Basic Decision Tree	97.14%	0.942	0.945	0.939	0.960
Random forest algorithm	97.82%	0.956	0.941	0.973	0.995
XGBoost Classifier	98.14%	0.963	0.937	0.991	0.997
Decision Tree Grid Search+CV	98.23%	0.965	0.933	0.999	0.994

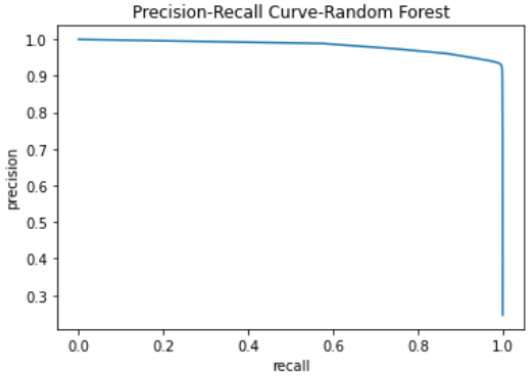
Below are the Precision-Recall and ROC Curve

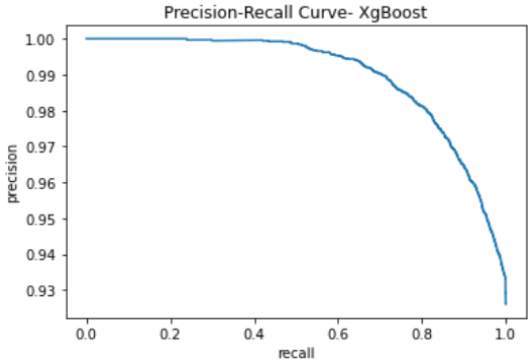


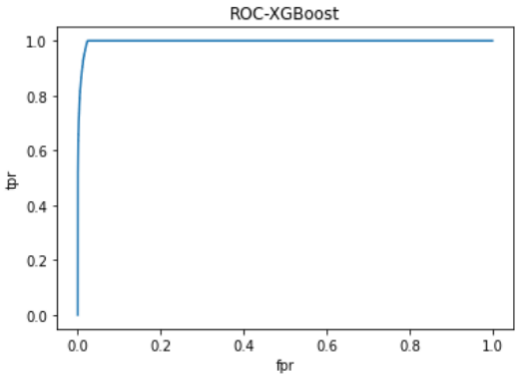


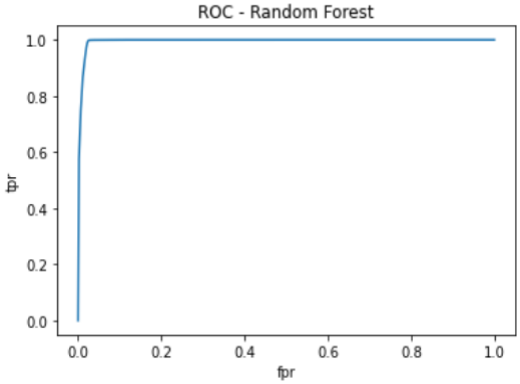

Precision-Recall Curve-Naive Bayes
precision
recall

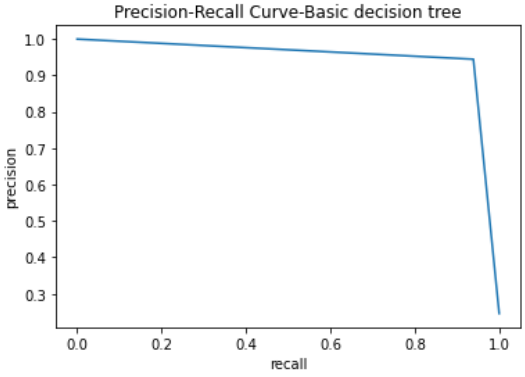

ROC-Naive Bayes
tpr
fpr

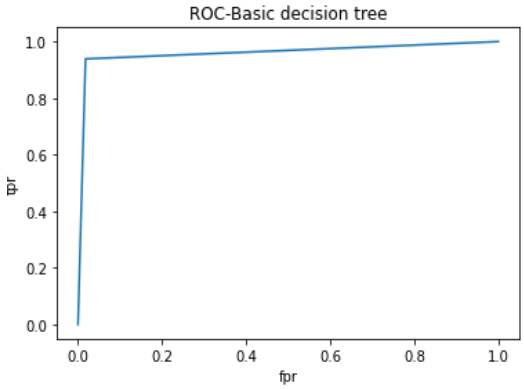

Precision-Recall Curve-Random Forest
precision
recall

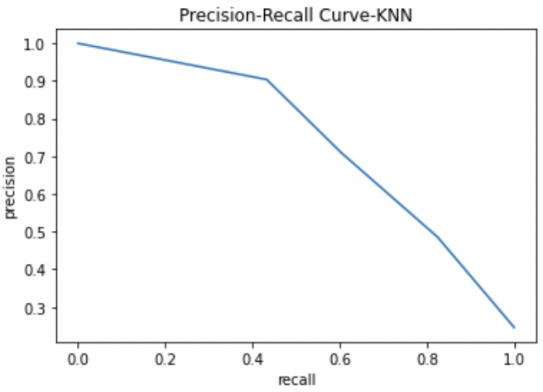

Precision-Recall Curve- XgBoost
precision
recall

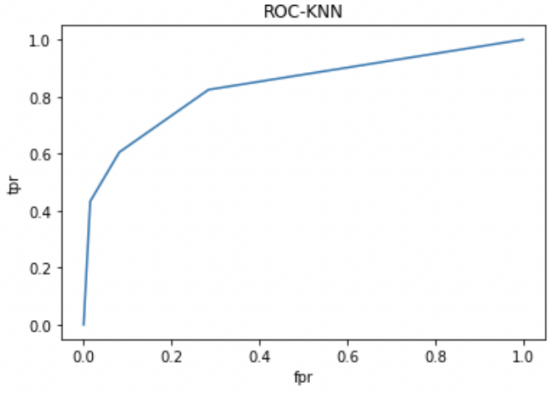

ROC-XGBoost
tpr
fpr

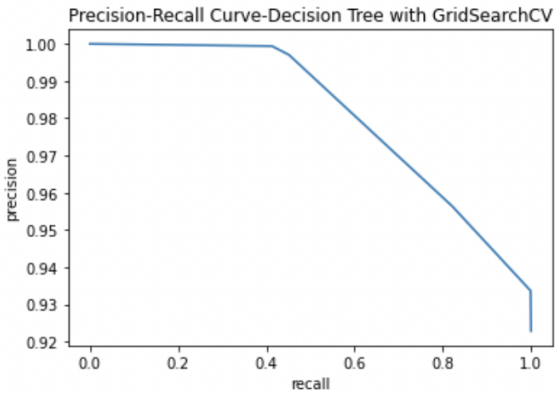

ROC - Random Forest
tpr
fpr

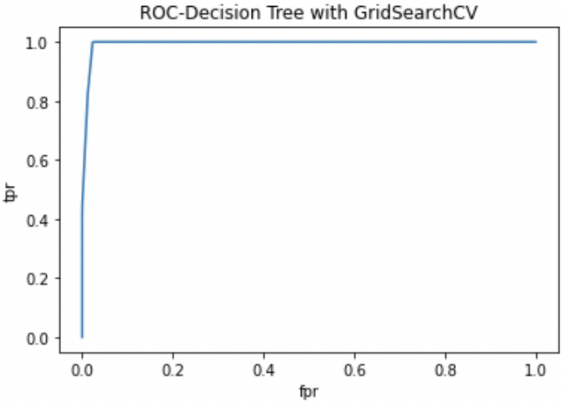

Precision-Recall Curve-Basic decision tree
precision
recall


ROC-Basic decision tree
tpr
fpr


Precision-Recall Curve-KNN
precision
recall
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.0
0.2
0.4
0.6
0.8
1.0


ROC-KNN
tpr
fpr
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Precision-Recall Curve-Decision Tree with GridSearchCV
precision
recall
1.00
0.99
0.98
0.97
0.96
0.95
0.94
0.93
0.92
0.0
0.2
0.4
0.6
0.8
1.0


ROC-Decision Tree with GridSearchCV
tpr
fpr
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

6. CONCLUSION & FUTURE SCOPE

- The Decision tree with GridSearch+CV and XGBoost Classifier gave better accuracy than other algorithms for this dataset.
- In conclusion, machine learning algorithms can predict loan defaulters in the Banking Sector and mitigate other risks.
- In the future, if time series data is included in this project, we can get a better understanding of consumers, and the ML algorithms would predict more accurately.