Credit Risk Prediction

An Application of Data Mining on Bank Customer Dataset

Team 6

Lokesh Vaddi, Nikhil Kumar Kanisetty, Shanmukha Yaswanth Reddy Kallam, Mareedu Mahesh Chandra

May 15, 2022

CMPE-255 Data Mining,San José State University

Abstract

In this work, we explore the development of a tool that banks may use to determine whether or not to extend credit to a customer based on his personal and financial characteristics. Banks lose a significant amount of money as a result of credit defaults, and it is ultimately ordinary customers who bear the brunt of this error. Credit Risk Analysis is used by banks to ensure that credit is supplied to a trustworthy consumer. Credit risk is defined as the risk of defaulting on a loan as a result of the borrower's failure to make mandatory debt payments on time. The lender assumes this risk because the lender loses both the capital and the interest on the loan. In this paper we discuss on Machine Learning-based credit risk analysis that eliminates the time-consuming human process of assessing numerous criteria and conditions on which credit can be granted. In the process, it also eliminates the human factor of mathematical mistake and corruption. From this project we aim to build a model to predict whether a person is eligible to get a credit or not.

1 Introduction

Interest on loans is an important source of revenue for banks. Banks assess various characteristics before granting a loan to a client because they must be certain that the consumer will be able to repay the loan within the loan term. This carries a high level of risk; any error in evaluating client history might result in the bank losing credit. As a result, it is critical to do a customer analysis before extending credit to them.

This Project explains the strategy to developing a machine learning model that can predict whether a consumer will be approved for a loan or not. We investigated an open source dataset containing data on consumers who paid their debts and those who did not in order to solve this challenge. We delegated this task to a machine learning model.

2 Methods

To develop the model, we have designed a pipeline of steps to reach the end goal. The steps involved in the pipeline are :

- Data Collection

- Exploratory Data Analysis
- Pre-processing Data
- Feature Selection
- Model Training and Parameter tuning
- Model Evaluation
- Deployment

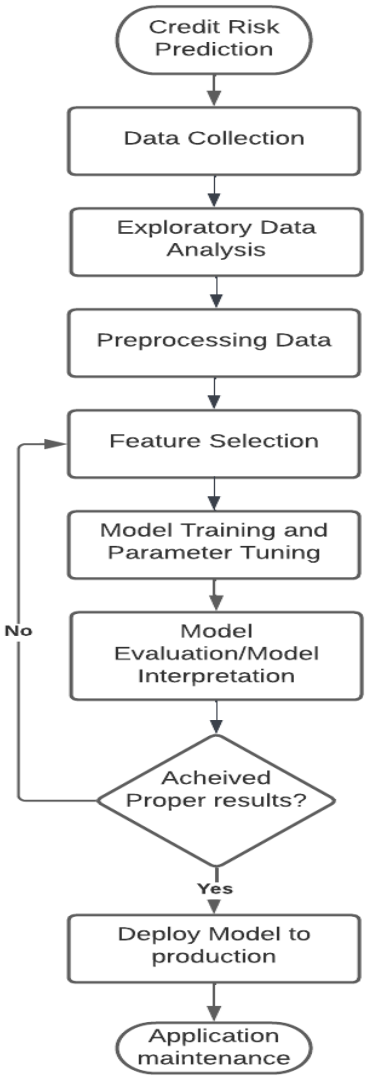


Figure 1: Model Development Pipeline

2.1 Data Collection

For this purpose we have used an open source dataset for solving the problem.Dataset contains 32581 records and 12 columns(11 features and 1 target varible)

2.2 Exploratory Data Analysis and Pre-processing

Before providing loan to any customer, bank analyzes few parameters of customers. These are general parameters, once passing these through additional set of customer data is analyzed. We are trying to build a model based on the initial set of parameters that help decide whether to give loan or not.

About the dataset, it is a combination of both numerical and categorical data. Below are the list of features used to predict whether a customer would be eligible for loan or not.

Feature Name	Description
person_age	Age
person_income	Annual Income
person_home_ownership	Home ownership
person_emp_length	Employment length (in years)
loan_intent	Loan intent
loan_grade	Loan grade
loan_amnt	Loan amount
loan_int_rate	Interest rate
loan_percent_income	Percent income
cb_person_default_on_file	Historical default
cb_person_cred_hist_length	Credit history length
loan_status	Loan status (**0 is non default 1 is default**)

Table 1: Dataset features and target variable information

loan-status is the target variable up for prediction.Default is the failure to repay a loan according to the terms agreed to in the promissory note.

As a part of EDA and pre-processing we have performed the below methods to clean the data and visualize the insights:

2.2.1 Dropping Duplicate records and Handling missing value

From the given source dataset, we have dropped all the duplicate records. The dataset is checked for find NULL(NA) values. After analyzing the dataset we have found that there are no NULL values. Instead we have a placeholder **?** where ever the records are missing.

loan-int-rate and loan-intent are two columns found to have missing values. The values are imputed with their corresponding mean values. The reason for imputing mean is that mean and median of these columns are in very close range indicating no outliers.

2.2.2 Removing Outliers

We have looked at the box plots of all the features. Out of all the features we have analyzed that there are few customer records whose age(person-age) is greater than 80. Considering normal expected age as 80 yrs. We have discarded data of customers whose age is greater than 80.

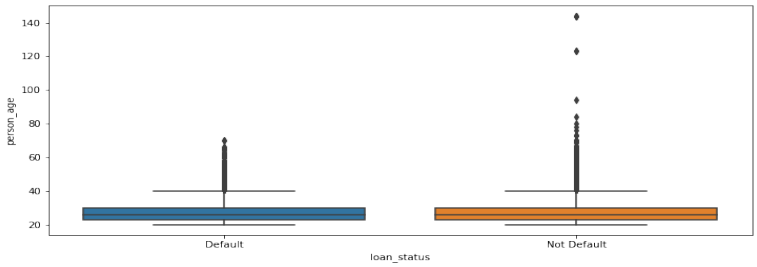


Figure 2: Box plot of person_age feature

For feature person-emp-length, considering the the retirement period is 60 years, max employment for a person would be 40-45 yrs if he/she starts working around 15-20. So we discarded data of persons where employment period is greater than 41.

2.2.3 Correlation among features

Given data, there is a possibility that multiple features might express the same behavior. To eliminate this correlation, we have produced heat maps and pair plots on numerical data. Analyzing the heat maps gives us a picture of how correlated two features are with respective to each other. By visualizing the pair plots we can analyze the alignment of data points.

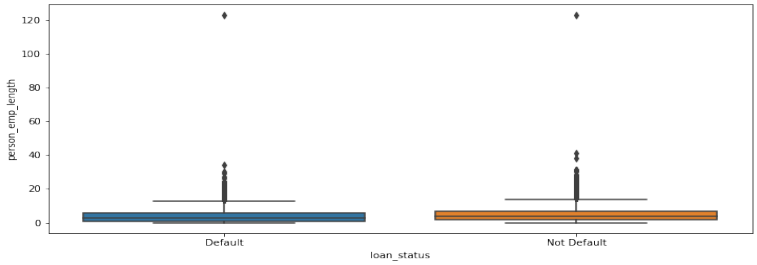


Figure 3: Box plot of person_emp_length feature

The distribution direction of data points explains us whether the correlation is negative or positive.

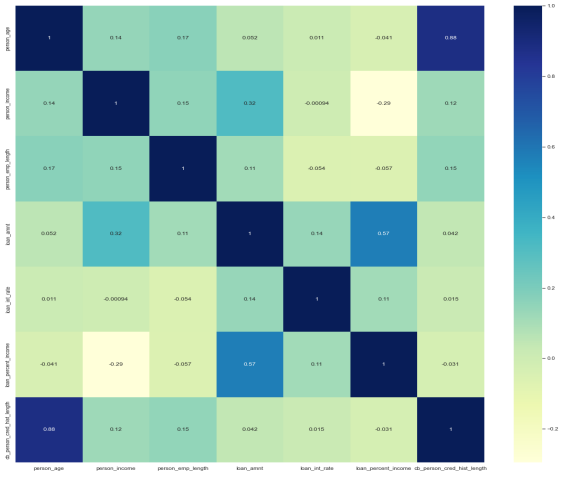


Figure 4: Co-relation Heat map

From Figure 4 and Figure 5, we can observe that there is high positive correlation between the features cb_person_cred_hist_length and person_age haveing high collinearity of 88%. As collinearity is high we are dropping the feature cb_person_cred_hist_length in the further processes.

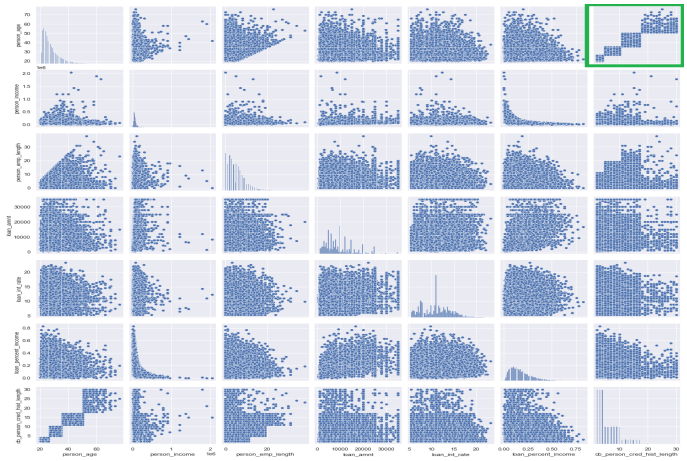


Figure 5: Pair plot among numerical features

2.2.4 Findings from EDA

3 Model Training and Testing

3.1 Data Processing before training

3.1.1 One hot encoding

Since we have a few categorical columns in the dataset which are ordinal and containing few categories, we can use one hot encoding on those features to make use of these features on models like Logistic Regression and XGBoost. We used OneHotEncoder() from the scikit-learn library on the dataset. From the below image we can see the one hot encoded data.

3.1.2 Train Test split

Train-Test split is a strategy for measuring the performance of a machine learning algorithm. As we are using supervised learning technique for classification or regression task train and test split would be helpful. We have divided dataset into two subsets as part of the train-test. The training dataset is the initial subset, which is used to fit the model. The second subset is not used to train the model; instead, the dataset's input element is given to the model, which then makes predictions and compares them to the predicted values.

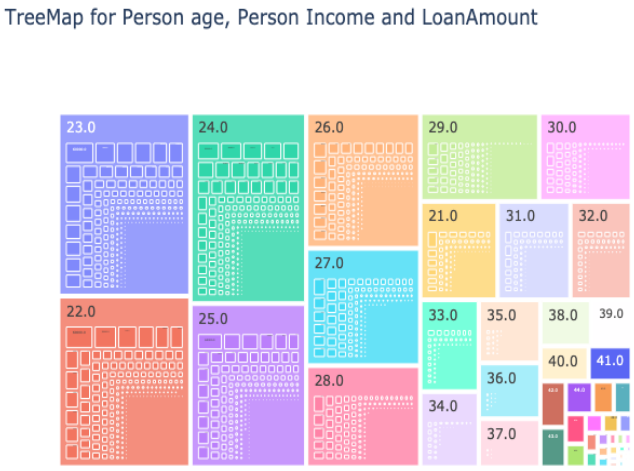


Figure 6: TreeMap for Person Age, Person Income and Loan Amount

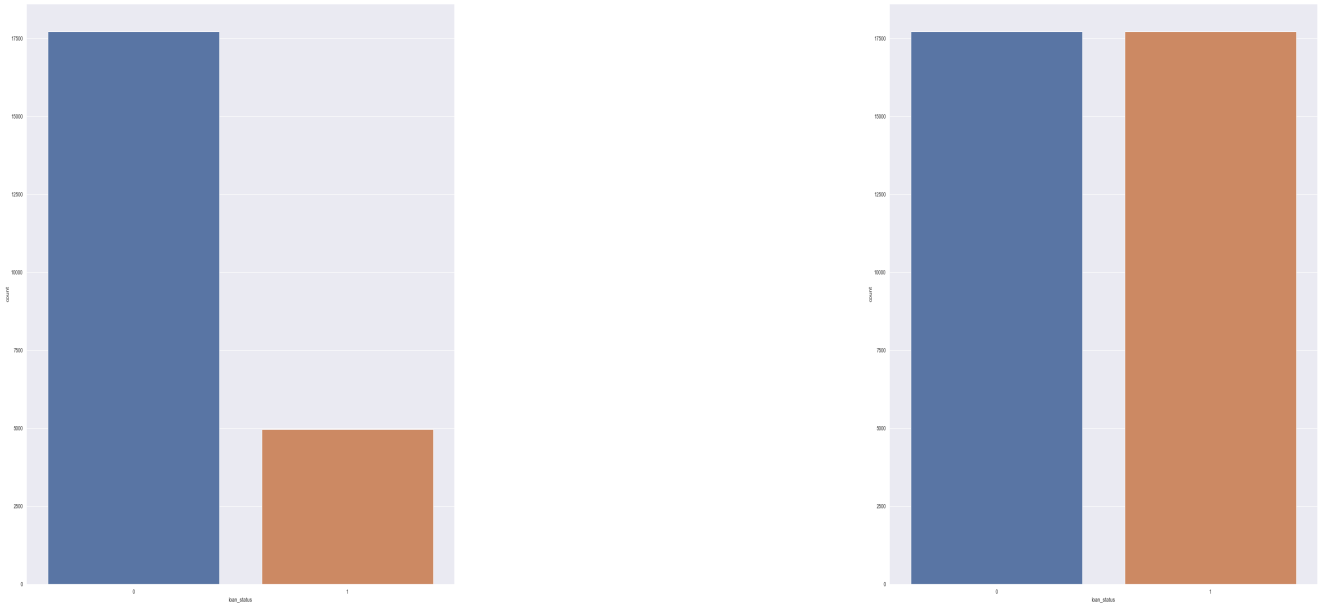
loan_amnt	loan_int_rate	loan_percent_income	person_home_ownership_MORTGAGE	person_home_ownership_OTHER	person_home_ownership_OWN	person_hom
1000.0	11.14	0.10	0.0	0.0	1.0	
5500.0	12.87	0.57	1.0	0.0	0.0	
35000.0	15.23	0.53	0.0	0.0	0.0	
35000.0	14.27	0.55	0.0	0.0	0.0	
2500.0	7.14	0.25	0.0	0.0	1.0	
...	...	...	...	...	...	...
5800.0	13.16	0.11	1.0	0.0	0.0	
17625.0	7.49	0.15	1.0	0.0	0.0	
35000.0	10.99	0.46	0.0	0.0	0.0	
15000.0	11.48	0.10	1.0	0.0	0.0	
6475.0	9.99	0.15	0.0	0.0	0.0	

Figure 7: Dataset after One hot encoding

3.1.3 Upsampling the Training data

When working with unbalanced datasets, the problem is that most machine learning techniques approaches will overlook the minority class, resulting in poor performance, even though it is often the minority class that is most relevant. One of the approaches to address an imbalanced dataset is to oversample the minority class. Upsampling is a procedure where the synthetically generated data points are introduced into the dataset. Following this procedure results in counts of both the labels being almost identical. This procedure prevents the model from overlooking the minority class and inclining towards the majority class.

After dividing the dataset into two subsets as training set and test set, we performed upsampling only on the training subset using SMOTE (Synthetic Minority Over-sampling Tech-



(a) Before Upsampling *(b) After Upsampling*

Figure 8: Countplot before and after Upsampling

nique) from imblearn.

3.2 Models Training and Parameter Tuning

For solving the problem we have applied multiple models on the dataset. Below are the few models we applied and the processes involved.

3.2.1 Random Forest Classifier

In this module, we will create several random forest models for our dataset and see which of them works best. We will create a simple random forest, then tune its parameters using GridSearchCV, and then form a model using only the features selected through the SelectFromModel() method. As our dataset is imbalanced, we first upsample it and then use it to train our models.

Initially, we create a simple random forest classifier, with max_depth set to 3. Even though the f1 scores are good for both classes in the training dataset, the f1 score for class 1 isn't the best in the test dataset. As a next step, we will use the GridSearchCV() method to

tune the hyper-parameters. In the random forest classifier, we set appropriate ranges for all parameters and tuned them to get the best results. The best_params_ variable is used to fetch the parameter values, which are then used to form a random forest classifier. The f1 score has increased significantly for class 1 in the test dataset as well as for both classes in the training and test datasets. There is slight overfitting in the dataset with 97% for the training data and 93% for the testing data. We will examine if we can reduce this overfitting by using only the top features needed for the model. We use the SelectFromModel() method to select our previous model. SelectFromModel() extracts the most significant features from the given data, based on their weights. Fit the train data into the SelectFromModel() with our model then use the get_support() function to get the top features. These top_features are then used to train the previously tuned model. We can see from the results that we did not achieve what we expected, but rather the f1 score of class 1 decreased. The overfitting was also not reduced. Out of these 3 models, the second, hyper-parameter-tuned model performs the best.

3.2.2 Decision Tree Classifier

As the decision tree's are widely used in banking sector, we have created Decision Tree model for our dataset. Decision Tress are easy to interpret. After a little explanation, we can grasp decision tree models, but decision-tree learners can produce too complicated trees that do not generalize data well. For this overfitting problem we tried to solve using techniques like pruning. Results of the decision tree algorithm on the dataset are much more interesting than LogisticRegression as we got 89% accuracy.

3.2.3 Logistic Regression

Logistic regression is one of the popular machine learning techniques which comes under the category of supervised learning techniques. It's used for predicting categorically dependent variables using a set of independent variables. It is widely used for binary classification like yes or no , default or not default. Logistic Regression is relatively faster than other supervised classification techniques or ensemble methods, but it is less accurate. With logistic regression we got an accuracy of 81% and precision of 55%.

3.2.4 XGBoost Classifier

For solving this problem we implemented XGBoost agorithm.This algorithm uses Gradient boosting technique and trees to understand the data patterns. About model, it is an ensemble method, meaning it's a way of combining predictions from several models into one. It does that by taking each predictor sequentially and modelling it based on its predecessor's error.

Because of its accuracy and simplicity, it has become one of the most used machine learning techniques. We can use this model for both regression and classification.

After applying up-sampling and encoding techniques,we have trained the model on the resulting dataset.The reason for considering this algorithm is that,we have converted the entire dataset to numerical,and tree based algorithms create the best splits on numerical data.Instead of creating complex equations making the model complicated these tree base models create simple yet powerful models.

We received 93% accuracy on the test dataset,and AUC metric is 0.95 which tells us that model is able to differntiate classes very well.

4 Comparisions

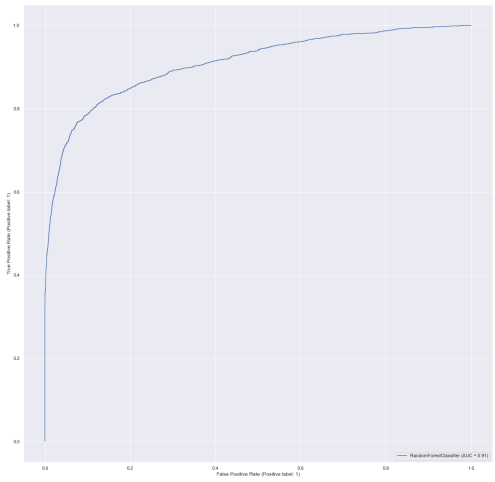
After applying multiple models on the dataset we have monitored results from each model. We compare these results and come to a conclusion on the best model suited to solve this problem. The metrics we have used to find the generalized model are:

- Accuracy
- Precision
- Re-Call
- F-1 Score
- ROC-AUC

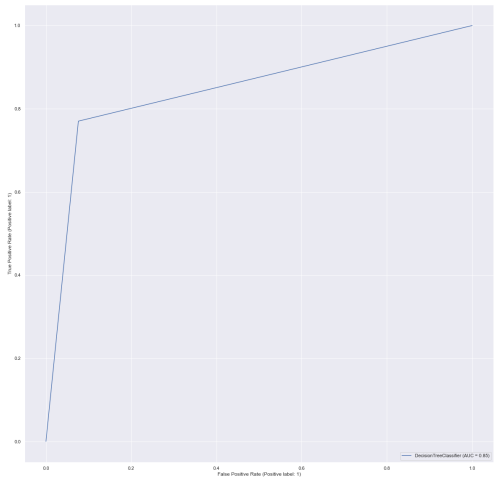
Below are the evaluation metrics across all the models.

Model	Accuracy	Precision	Recall	F1-Score
RandomForestClassifier	0.900432	0.825745	0.690644	0.752176
DecisionTreeClassifier	0.890249	0.73917	0.770099	0.754317
XGBClassifier	0.933244	0.941986	0.74048	0.829166
LogisticRegression	0.811767	0.550084	0.766808	0.640613

Table 2: Evaluation metrics across different models



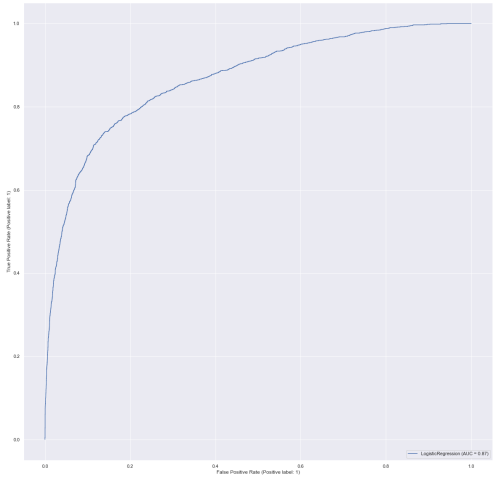
(a) Random Forest ROC-AUC(0.91)



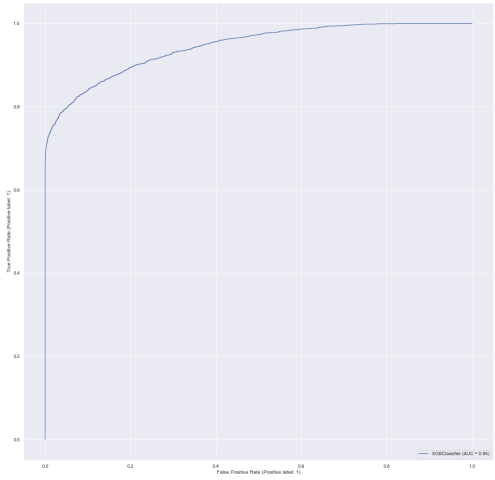
(b) DecisionTreeClassifier ROC-AUC(0.85)

Figure 9: ROC AUC Curves for Random Forest and Decision Tree classifiers

Out of all the models tested and evaluated, XGBoost and Random Forest delivered high performance in identifying the classes Default and Not Default.



(a) LogisticRegression ROC-AUC(0.87)



(b) XGBClassifier ROC-AUC(0.94)

Figure 10: ROC AUC Curves for LogisticRegression and XGBoost classifiers

5 Conclusion

After evaluating the performance of all the models,we understand that tree based models are best suited for this dataset.Bagging models like Random Forest and Boosting models like XGBoost delivered similar results.

Though the dataset is balanced by duplicating the minority class,adding additional data points in future might make the model much robust.To add strength to the model,in future we plan to add additional features to the dataset and testing it with neural networks.

References

[1] Farzad, Shahbazi (2019), *Using decision tree classification algorithm to design and construct the credit rating model for banking customers.*

[2] Tzu-Tsung Wong, Shang-Jung Yeh(2019), *Weighted Random Forests for Evaluating Financial Credit Risk*

[3] Ali Al-Aradi(2014), *Credit Scoring via Logistic Regression*

[4] Oliver Takawira, John W.Muteba Mwamba (2022), *Sovereign Credit Ratings Analysis Using the Logistic Regression Model*

[5] Kui Wanga, Meixuan Li, Jingyi Cheng, Xiaomeng Zhou, Gang Li (2022), *Research on personal credit risk evaluation based on XGBoost*