Problem Statement:

Loan default prediction is one of the most critical and crucial problems faced by financial institutions and organizations as it has a noteworthy effect on the profitability of these institutions. In recent years, there has been a tremendous increase in the volume of non – performing loans which results in a jeopardizing effect on the growth of these institutions.

Therefore, to maintain a healthy portfolio, the banks put stringent monitoring and evaluation measures in place to ensure timely repayment of loans by borrowers. Despite these measures, a major proportion of loans become delinquent. Delinquency occurs when a borrower misses a payment against his/her loan.

Given the information like mortgage details, borrowers related details and payment details, our objective is to identify the delinquency status of loans for the next month given the delinquency status for the previous 12 months (in number of months).

Approach:

Data preprocessing:

The first step I did after loading the data is checking the percentage of each class(Target Variable Analysis) in the data set to get a sense of how imbalanced the data is. By the way this is binary classification problem so there are only two classes and I surprisingly found out that the data is highly highly imbalanced data. It's just only 0.5 percent of loans(data points) are delinquent and the remaining 99.5 percent of data points are non-delinquent. So we should be really careful while feeding the data into the model and choosing the metric for hyperparameter tuning because if all points are predicted as non-delinquent we still get an accuracy of 99.5 percent which is very high but its misleading.

The next step is I checked whether there are any duplicates in the majority class and I found out that there are 16 duplicates and I removed them because they may mislead the model. The next part of data cleaning is to fill null values if there are any but fortunately the data set does not have any null values. The next step is handling categorical features and date features in the dataset. I found out that there are two categorical features in the dataset “source” and “financial_instituition” . After checking these features I converted them to numerical features. Similarly there are different date features I splitted them to date, month and year and created three separate features for each feature. As there is no text data in the dataset no text preprocessing required at all. As the data is highly imbalanced in nature we need to either upsample the minority class or downsample the majority class but the downsampling is not preferred as we may lose a lot of information so in turn model may not be able to learn from the training data. But I tried both methods and obviously found out that upsampling outperforms downsampling. For upsampling I used the SMOTE algorithm and for downsampling I used the NearMiss algorithm.

Data Visualisation:

Here comes the interesting part data visualisation. First I started with uni-variate analysis where I tried multiple features and plotted probability density function(pdf) with target variable whether these features can be very important for classification. But I found out that pdfs of almost all features are overlapping. Then I plotted a correlation matrix between all features and found out some correlated features. I also visualised data using PCA(dimensionality reduction) but not able to get any insights from the plot.

Model Selection:

Data is highly imbalanced. So the first type of algorithm came to my mind are bagging and boosting algorithms. And I choose the metric as f1-score for hyperparameter tuning as it works well for imbalanced data and also it is the metric used in the competition to evaluate the results. I started with XGBoost with multiple learning rates,multiple max_depth,multiple n_estimators and applied Grid Search and found out the best hyperparameters as max_depth=10 and n_estimators=1000 and learning_rate=0.01.Next algorithm is Random Forest the results of random forest are slightly improved compared to XGBoost. But what I have observed from these two algorithms is they tend to overfit on the train data(getting a f1-score of 0.99 with train data and 0.48 with validation data). So I thought to introduce some randomization in the model so that it introduces regularization which helps to avoid overfitting the training data. So I used the “Extremely Randomized Trees” algorithm which has more randomization compared to Random Forest and as expected, it gave me good results compared to all the previous results. Now as generally in the competitions people use stacking classifiers as they help us to give good results at least a small improvement from the the base classifiers so that we can move up at least some positions on the leaderboard. So I used stacking classifier with RF,XGBoost and Extremely Randomized Trees with meta classifier as XGBoost. But when I changed the meta classifier to RF results are slightly improved. I also tried after standardizing the data but it did not improve any results as expected because tree based models does not get affected by the scale of the data.

Note: Rerunning the same code may give you slightly different results due to different random seed.

My F1-score on the public leaderboard is 0.31351351 hope it will be increased when private leaderboard is released.

Thank you.