Approach

I first visualized the dataset to check if there is any correlation between any columns. Then I plotted various columns to check the behaviour of the dataset. After plotting the graphs, I used **LabelEncoder** to encode the categorical variables. Then I used **Feature Scaling** to rescale the dataset. Then I used **XGBoost Classifier** to train the dataset. I also used GridSearch to select the best parameters of the model.

Quality checks performed / Errors found:

- No Errors were found.

Data Visualization:

- No. of unique elements for each month

Column	No. of unique elements
m1	4
m2	5
m3	6
m4	7
m5	8
m6	9
m7	10
m8	10
m9	11
m10	12
m11	13
m12	13

Number of unique elements increases as month increases.

- number_of_borrowers and co-borrowers_credit_score are highly correlated to each other.
- Majority of source is “X” with a count of 63858 (55.02%).
- insurance_type has value “0” with a percentage of 99.67%.

Data Preprocessing:

- Used LabelEncoder from scikit-learn library to encode the columns which has type as object.
- Used Feature Scaling to rescale the training and testing dataset.

Model Choice Explanation:

- I tried different ML models like Random Forest, Logistic Regression, SVM, KNN, Decision Tree. All the models were not giving the desired results. Some models were underfitting like Random Forest and Logistic Regression whereas some models were overfitting like Decision Tree.
- Then Finally I tried XGBoost and tuned it with different values for parameters.
- Also I used GridSearch to find out the best parameters.

Final XGBClassifier parameters:

```
classifier = XGBClassifier( learning_rate =0.1,
                            n_estimators=112,
                            max_depth=9,
                            min_child_weight=5,
                            gamma=0,
                            subsample=0.8,
                            colsample_bytree=0.6,
                            objective= 'binary:logistic',
                            nthread=4,
                            scale_pos_weight=13,
                            reg_lambda=5,
                            alpha=0,
                            base_score=0.5,
                            seed=1029   )
```

The most important features are loan_term and insurance_type.

Key Takeaways from the challenge:

- Always visualize the dataset to get the insights before applying any ML model to it.
- Try different models. Never stick to one model.
- For classification problems, examine and understand the percentage of positive labels in the target variable.
- Do one submission to see the score on public leaderboard. Avoid making submission every 5 mins even in case there is no limits per day.

5 things a participant must focus on while solving such problems:

- Start with slowly and thoroughly reading the problem statement. Do a quick inspection of the data by using pandas builtin functions and note down my observations. By quick inspection I mean loading into a variable df, doing df.info(), df.nunique(), df.describe() and so on.
- Understand each of the predictors (attributes) and look for biases in the data. This is the key and it is not related to algorithms. It is mainly commonsense and business knowledge. For classification problems, examine and understand the percentage of positive labels in the target variable. For regression problems, examine the distribution of the target variable.
- Do standard scaling on continuous variables to bring them all to mean 0 and standard deviation 1. Next do a correlation plot. Drop the correlated columns (typically correlations > 0.7) to prevent model being biased.
- Split the Training Set into Train, Development and hold out sets. I normally keep 20%, 10% as dev and hold out set if the training data is more than 100k, if not something like 10%, 5%, because I want my training to be robust. This sounds contradictory to Prof Andrew's advice on keeping a smaller percentage for dev and hold out, when the training data is large. But the training data size there is much larger.
- Try training with Logistic Regression model, if it is a classification problem, and Linear Regression for regression problem. Now start with hyperparameter tuning, if you know a bit of the theory. If not, make an attempt to read up and try. There is nothing to lose. Do K-fold cross validation with sci-kit learn libraries.

Aakash Jhawar
+91 9413564182 aakash.16.jhawar@gmail.com
aakashjhawar.github.io 16uec001@lnmiit.ac.in

https://github.com/aakashjhawar/AnalyticsVidhya-India-ML-Hiring-Hackathon-2019