



CREDIT CARD DEFAULT PREDICTION

Using Machine Learning Techniques

Submitted by : Chinta Krishna Mourya



Objective :

Financial threats are displaying a trend about the credit risk of commercial banks as the incredible improvement in the financial industry has arisen. In this way, one of the biggest threats faces by commercial banks is the risk prediction of credit clients. The goal is to predict the probability of credit default based on credit card owner's characteristics and payment history

Benefits :

1. Improved accuracy - By using ML algorithms, it is possible to analyze a wider range of variables and identify patterns that can predict credit card default with greater accuracy.
2. Faster decision-making.
3. Risk Reduction.
4. Enhanced customer experience - help credit card issuers to personalize their offerings based on a customer's financial behavior and credit history.
5. Cost Saving - can reduce the cost of manual analysis and increase efficiency.



Dataset and Cleaning :


- This project used the UCI CreditCardFraud data.
- Dataset consists of 25 columns, removed “ID” column from it as it is unnecessary.
- The attribute name 'PAY_o', 'default.payment.next.month' were renamed to 'PAY_1' and 'Default ' respectively.
- Some unspecified values in 'Marriage', Education' and 'PAY_1,2,3,4,5,6' columns have been processed as per dataset description.
- No Null values in dataset.

Insights from data:

- Defaults have a higher proportion of Lower LIMIT_BAL values
- NonDefaults have a higher proportion of Females (Sex=2)
- NonDefaults have a higher proportion of MoreEducated (EDUCATION=1 or 2)
- NonDefaults have a higher proportion of Singles (MARRIAGE=2)
- NonDefaults have a higher proportion of people 30-40years
- NonDefaults have a MUCH higher proportion of zero or negative PAY_X variables (this means that being current or ahead of payments is associated with not defaulting in the following month). *This is a strong relationship as the distribution are more separated - so we expect the PAY_X to be important!*
- Dataset is unbalanced.

Data Preprocessing :

- Since, dataset is highly imbalanced {NonDefault: 23335, Default: 6630}, balanced it using oversampling technique – SMOTE.
- Data is in different scales, so used StandardScaler to scale the data for better results and saved this scaler object for scaling the user input in future.



Model Building :

- Divided the dataset with test size 0.25.
- Trained the trainset with different models SVM, XGBoost, GradientBoostin, DecisionTree, LogisticRegression, naiveBayes and RandomForest.
- RandomForest has performed better and hypertuned it to improve the results.
- Finally, model is performing maximum at its best with Accuracy 0.86.

RANDOM FOREST MODEL:

- The term “Random Forest Classifier” refers to the classification algorithm made up of several decision trees. The algorithm uses randomness to build each individual tree to promote uncorrelated forests, which then uses the forest’s predictive powers to make accurate decisions.
- The random forest algorithm is used in a lot of different fields, like banking, the stock market, medicine and e-commerce.

Tuning :

- Trained with different values of "n_estimators", "criterion", "max_depth", "ccp_alpha".
- These are some hyperparameters of the Random Forest model that are used to tune using grid search with 5-fold cross-validation to find the best combination of values for the model.

Hyperparameters briefly :

- `'n_estimators'`: the number of decision trees in the forest.
- `'Criterion'`: the function to measure the quality of a split. The two options used in this code are "entropy" and "gini".
- `'max_depth'`: the maximum depth of each decision tree in the forest. If set to None, nodes are expanded until all leaves are pure or until all leaves contain less than `min_samples_split` samples.
- `'ccp_alpha'`: parameter for Minimal Cost-Complexity Pruning. It adds a cost complexity penalty to the impurity value of each node in the tree. Smaller values of `ccp_alpha` increase the amount of pruning, resulting in smaller trees.



Deployment:

- Created a flask app to take the user input and pre process it to return the prediction.
- Created front end using HTML, CSS.
- Deployed it in AWS instance using EC2 instance.



Conclusion :

- Final dataset is cleaned, scaled, and balanced.
- Making clusters of train data and training the clusters with different models to find the best model is also giving same results as training on RandomForest.
- So, chose single model and trained it and tested with test set and built the model to save it.
- Final model is good with 86% accuracy and this would help issuer of creditcard , to decide whom to give and on what factors.
- Deployed it on cloud platform.



THANK YOU