**Analytics\_Vidhya\_India\_ML\_Hiring\_Hackathon\_2019**

Loan Defaulting is the failure to repay a debt including interest or principal on a loan or security. A default can occur when a borrower is unable to make timely payments, misses payments, or avoids or stops making payments. Individuals, businesses, and even countries can fall prey to default if they cannot keep up their debt obligations. So it crucial for the businesses to predict this behaviour in advance and address it sooner before it leads to a loss for the company.

Here we are given the task of creating a model to predict the customers who are more likely to default.

We are given a train dataset with the following features.

* loan\_id
* source
* financial\_institution
* interest\_rate
* unpaid\_principal\_bal
* loan\_term
* origination\_date
* first\_payment\_date
* loan\_to\_value
* number\_of\_borrowers
* debt\_to\_income\_ratio
* borrower\_credit\_score
* loan\_purpose
* insurance\_percent
* co-borrower\_credit\_score
* insurance\_type 0 - Premium paid by borrower, 1 - Premium paid by Lender
* m1 to m12 Month-wise loan performance (deliquency in months)
* m13 target, loan deliquency status (0 = non deliquent, 1 = deliquent)

**Data Pre-Processing:**

Initially we identify the categorical variables present in the data . We could see that columns “source”, “loan\_purpose” and “financial\_institution” have categorical data. Hence we replace them by numerical unique values.

**Feature Engineering:**

The first step of Feature Engineering is to remove the features which show high collinearity. These cannot be included in the model as it can lead to more noise in the output. As we need to test the data for multicollinearity. We first filter out the numerical features but filtering based on datatypes “int” and “float”.

Then in-order to remove the correlated features. We first calculate a correlation matrix and drop the columns which have correlation value more than 0.9. (0.9 is chosen after a trail and error process . Having tried values 0.8, 0.9 and 0.95. I could find 0.9 yielding the best results).

**Resampling:**

As we can see from above that the data is highly unbalanced. Here there are two ways in which we can resample the dataset. We either under sample the class with the higher count or we over sample the class with the lower count. oversampling of the minor class might add lot of noise to the data while undersampling the major class might remove most the data. A better way in this case is to do both oversampling of minor class and undersampling of major class by a factor of 3 or 4 to get better accuracy while modelling. ( The value 3-4 is chosen based on trial and error ).

The sampled data is later split as train and test for test the machine learning models.

**Scaling:**

Another Important step of feature Engineering is to scale the data inorder to normalize it and speed up the algorithm. Choosing the right scalar method also makes a significant difference in terms of accuracy. Here we use the Robust Scalar as the values are not influenced by large outliers.

**Modelling Approach:**

Initially I tried the logistic regression model considering the binary output which we are predicting. I had a decent score of accuracy and ROC-AUC score. Considering the fact that ensemble models usually perform better than logistic regression in most cases I decided to try random forest model. The results were better than the logistic regression model.

Since the ensemble methods are performing better. I tried one of the most efficient gradient boosting ensemble models which is LightGBM model. Surprisingly there was significant improvement in my accuracy and ROC\_AUC score. At this point I know that I have finalised he machine learning model.

Then I was performing some trial and error with the resampling. When I did just undersampling or just over sampling, the accuracy was not that great. But when I did both undersampling and oversampling and concat the output into a single dataframe. The results the so great I suddenly jumped to 1st position.

**Key Takeaways:**

1. Just because the target class is highly unbalanced doesn’t mean we should either do under sampling or over sampling on one of the classes. Because we can combine both and get a better model.
2. Its always better to use scaling and normalize the data. As it gives a stable model. And its always better to experiment with various types of scaling methods.
3. I had build models which had higher ROC-AUC score than the score in my final code but their predictions were not as good. So I was confused which

**Things to Focus:**

1. First most important thing is dropping the features with high multicollinearity.
2. Then comes figuring out the ideal machine learning model. An ideal model could change based on the data and the problem which we are trying to solve.
3. After figuring out the ideal model its always advisable to tune the hyperparameters and analyse which values provide high accuracy.
4. Sampling must be done with care since using both over sampling and under sampling for the same target class would provide better accuracy than using just one.
5. If the data has some large outlier its better to use robust scalar method so the values are not influenced by the outliers.