FOML Hackathon

Ahmik Virani
ES22BTECH11001

1

1. Introduction

I have tried several models on the given dataset, however the model which I finally used was Adaboost with base models as random forest.

2. Explaining the dataset

- The first thing that I observed in the dataset was that many of the columns contained majorly null values.
- I also noticed that majority of the columns were not at all related, so I decided to combine some variables which I felt could be merged to form better attributes.
- To handle the null values I did the following steps:
  - First I identified that some columns were boolean and their only values were 1, so the null values I replaced with 0.
  - I also observed that one column namely, OtherZoningCode, has codes of 'other zones', so I decided to make it boolean such that it indicates if the person has any other zone or not.
  - Next any remaining column with > 50% null values, I decided to ignore them.
  - Finally the remaining null values were filled with the medians of the columns.
- Finally I made a corelation matrix with the remaining columns (combination of old columns and new formed columns), and only keep those whose corelation matrix gave a value >= 0.01

3. Handling the categorical data

As I observed from the description that there were some categorical data given to us in the form of numbers, so with all the filtered data from the above set I ensured to do one hot encoding on them.

The ones remaining till the end were

- HarvestProcessingType
- NationalRegionCode
- DistrictId

We also had some boolean values till the end:

- TaxOverdueStatus
- NaturalLakePresence
- OtherZoningCode

4. Handling numerical data

There was no additioal step for handling the numerical data once the new columns were created.

5. Handling data imbalance

The most important thing that I realised was that there were too many points of the target='medium'. In particular the ratio of low:medium:high point was 1:3:1.

To handle this, I though of implementing the following strategies

- Undersampling
- Oversampling

5.1. Undersampling

I tried doing 2 variants of under sampling. The first was directly using the inbuilt library provided by imblearn.under_sampling. However this was not very effective and was not giving me a good score in the validation set that I had created using train test split (Note that I had first done train test split and then applied the under sampling on the train data so that the original test data i.e. validation set is not tampered with).

The next model which I thought would work was the I could divide the dataset first into 3 parts with labels low, medium, high. Then we know that medium has extra entries, so divide this randomly into 3 groups and form 3 models. Here every subset of 'medium' I combined the entire 'low' and 'high' data. Finally I took the most frequent score. Incase of ties, I used 1-nn classifier. However this approach too failed.

5.2. Oversampling

Here also I tried two differnt approaches. First I will mention the one which did not work, finally the one which did work.

One thing that I tried was SMOTE. However this did not work and one reason I guess is because the data was too complex, so finding synthetic instances was not really helping generalize well.

The next approach I used was directly duplicating the low and high labels. This was actually helpful. First I though it would lead to overfitting, however on validation set both on my own and kaggle, it outperformed the others and also by a good margin. So I continued with this.

5.3. Other approaches

It was given in the slides that MLPs and SVMs also are less prone to class imbalance, however on my preprocessed they actually turned out not performing very well.

6. Using Various Models

I tried a plenty of models, most of them and their performance is mentioned below:

- 1-NN: This performed well at the start, but as soon as I used ensemble methods this was quickly discarded.
- Neural Networks: Very average performance
- SVM: Not a good time v/s output tradeoff, ran for very long but did not give any better results that neural networks.

- XGBoost: This was performing very poorly.
- Adaboost: This was working very nicely. And I ended up with combining this and random forest for my final model.
- Random Forest: The performance of this was quite similar to Adaboost.
- Staking: I tried stacking adaboost and random forest, was giving good result, but not as good as below.
- Adaboost with base-estimator as Random-forest: This is final model which I trained and was satisfied with the output, and used this for my submission

7. Conclusion

Just to conclude, the model I am using is described below:

- The model I am using is Adaboost with base estimator as Random forest.
- The hyper-parameters of the base estimator i.e. random forest are:
  - max_depth = 5
  - n_estimators = 28
  - random_state = 42
- The hyper-parameters of the Adaboost model are:
  - estimator = random_forest
  - n_estimators = 150
  - learning_rate = 1
  - random_state = 42

Please Note that on my laptop (Macbook air M1 chip), it takes roughly 10-15 minutes to train the model, so please wait for the model to finish and ignore any warning message.

I have printed line : "Training model... Please wait..." and "Started prediction... Please wait..." to indicate that the process may take time.