









Hackathon Report

4	CS19BTECH11001		0.88502	9	1d
5			0.88310	12	1h
Your Best Entry 					
Your submission scored 0.88129, which is not an improvement of your best score. Keep trying!					
6	Test007				
7	AI20BTECH11006_AI20BTECH...				
8	Dedsec				
9	AI20BTECH11004_AI20BTECH...				

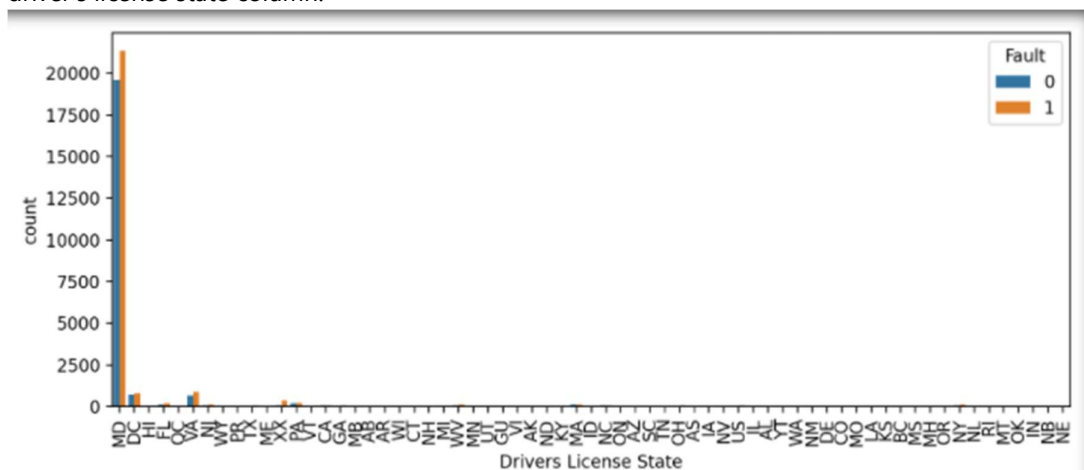
loading1234 

Kaggle Novice

Follow

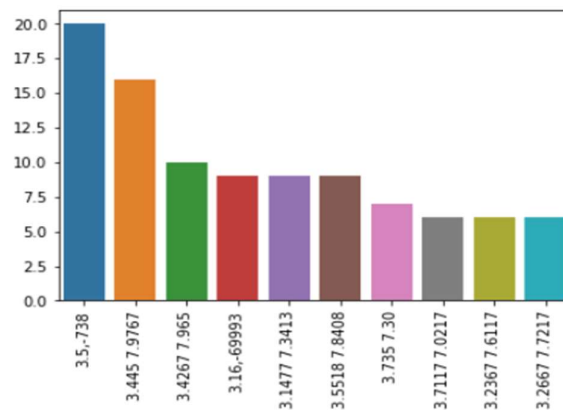
STEPS FOLLOWED in the process of building the best model:

1. Checked the number of NAN values for every features. If it exceeded 85-90 percent, we dropped those features.
2. Next, to understand the importance and analysis of each column we plotted the number of fault and 1 fault for each of the column values. The following shows the graph for testing the driver's license state column:



- Based on this information and eyeballing we removed certain columns which were not useful or had mostly unique values like Person ID column.
- We did further analysis of each and every column by plotting graphs. It helped us understand the features manually e.g.:

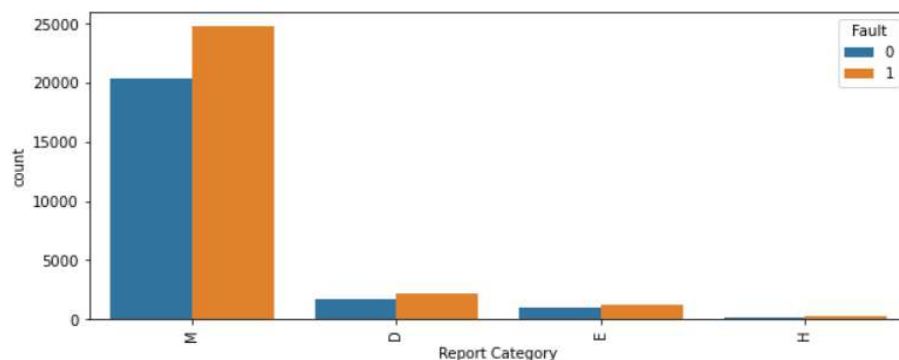
```
sns.barplot(data = df, x = df['Location'].value_counts()
plt.xticks(rotation = 90)
plt.show()
```



We were not able to understand the 'Location' column. We decided to keep the column only because it seemed to give almost similar/ better accuracy on test data somehow. We divided the location column into 2 sub columns.

- We found invalid data on Vehicle year but decided to keep them as they were present in test data also.
- For report number, we divided the entire column into categories based on the starting character. Eg if report number is M453P, then it becomes M category.

The following shows the count plot for the Report Category with hue set to Fault which we engineered above using our own convention.

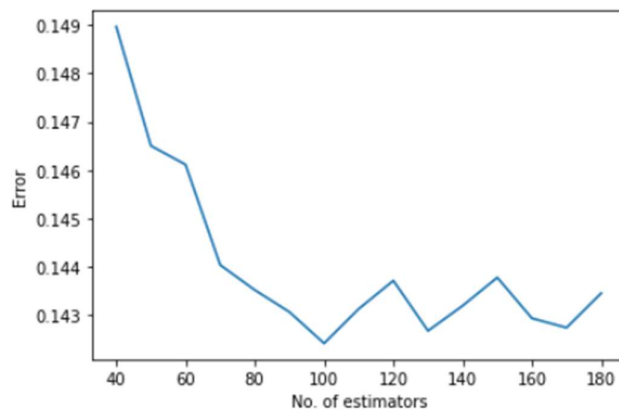


- In all the remaining column we have changed the NAN values into the Unknown values inorder to not waste any useful data.
- Fill all the rest of the NAN values as -1

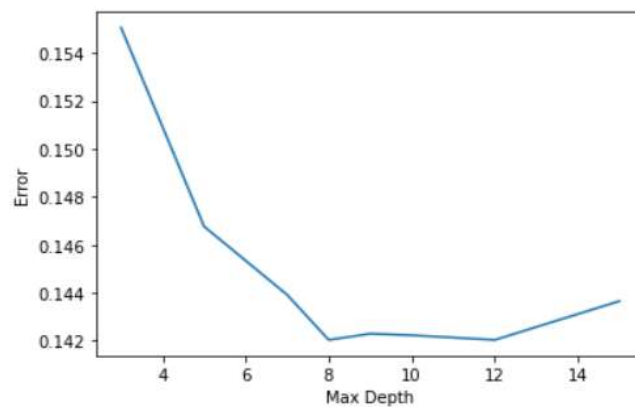
9. Included 4 extra columns in some of the models we made namely : 'Road Name', 'Cross-Street Name', 'Vehicle Make', 'Vehicle Model' and assigned them frequency encoded values since they were having many categories.
10. We used get_dummies function to convert the categorical features into numerical features in the dataset(also known as one-hot encoding).
11. Now, the same data cleaning is done on the given test set. Then we do intersection of test and train columns to only include the common columns.
12. We had tried using different models like random forests, Gradient Boost, XGBoost and LGBM. But we were getting the best accuracies with XGboost and so decided to go with it.
13. To get an idea of which hyperparameters might give better accuracy on the test case we tried running it on the validation set (70:30 split)

No. of estimators Vs Validation Error

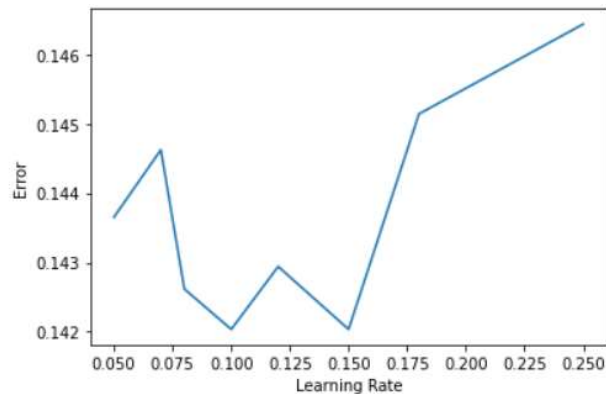
Time Taken: 290.294 secs



Max Depth Vs Validation Error



Learning Rate Vs Validation Error

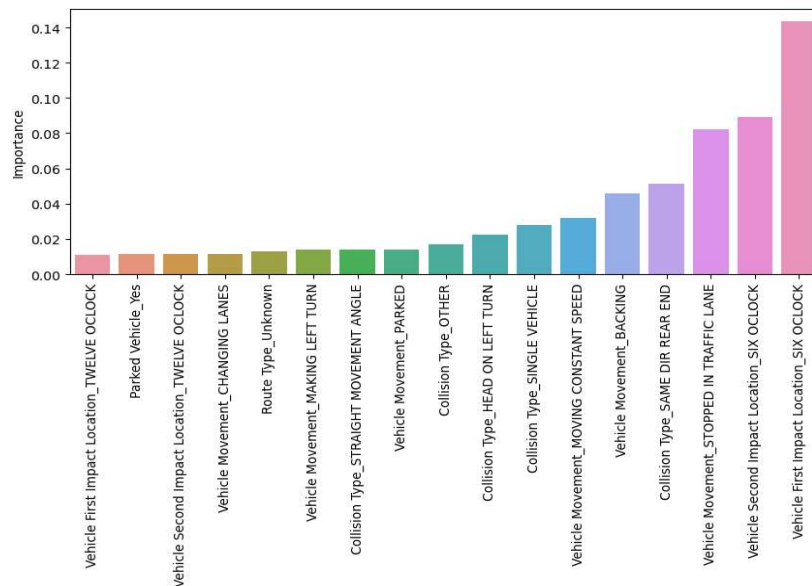


14. Based on the graphs, running exhaustive Grid-SearchCV and Random-SearchCV, and hit and trial method (for the test accuracy), we were getting an accuracy of 87.2 % on the public test dataset.

The following parameters were producing one of the most accurate models using a single XGBoost model.

booster = 'dart', n_estimators = 150, min_child_weight = 3, max_depth = 9, learning_rate = 0.1, gamma = 0.5, colsample_bytree = 0.4, eta = 0.1, reg_lambda = 0.2, reg_alpha = 0, scale_pos_weight = 1, eval_metric = 'logloss'

Plotted the feature Importance graph for the top 1% of the features the model thinks is important for the predictions,



15. Finally, we wanted to increase the accuracy even more and to get to the top of the leaderboards, we have taken 5 of our best accuracy model's predictions and did a majority

voting(similar idea as random forests) which fetched us an Accuracy of **88.3%** on the public leaderboards.