Report - Hackathon

cs19btech11009
es19btech11003

We have tried using `GradientBoostingClassifier` , `RandomForestClassifier` ,`XGBClassifier` for the given data set and found the following result and among all 3, we got the best model from XGBClassifier.

Classifier	Accuracy
1.GradientBoosting	0.86211
2.RandomForest	0.84814
3.XGB	0.86884

Explanation:

Firstly The given dataset is read into a data frame i.e, 'dataset' for train data and 'dt1' for test data. As a next step we have replaced all the null values with most frequent values. And also we have removed those columns where null values are more than 70%.

We have imported Label Encoder from sklearn.preprocessing library and using this Label Encoder we have converted categorical data into numerical data to fit it to model.

After doing all these we were left with following attributes

Feature = [`'Agency Name','Vehicle Model','Traffic Control','Municipality','Circumstance','Related Non-Motorist','Driver Substance Abuse','ACRS Report`

```
Type','Report Number','Route Type','Road
Name','Cross-Street Type','Cross-Street
Name','Collision Type','Surface
Condition','Light','Injury Severity','Drivers License
State','Vehicle Damage Extent','Vehicle First Impact
Location','Vehicle Second Impact Location','Vehicle
Body Type','Vehicle Movement','Vehicle Continuing
Dir','Vehicle Going Dir','Speed Limit','Equipment
Problems','Parked Vehicle','Vehicle Year']
```

which we used finally to train our model . Then we have split train data into train and validation data in ratio 90% , 10% using train_test_split from `sklearn.model_selection import train_test_split.`

Among 3 models XGBClassifier gave the best accuray(0.86392) that is because xgboost uses a more regularized model formalization to control over-fitting, which gives it better performance over others. It pushes the limit of computations resources for boosted tree algorithms. Which is the reason we have used xgboost.
Hyper parameters that we've used for our final model are

```
XGBClassifier(use_label_encoder=False,n_estimators=240
,learning_rate=0.054,max_depth=8).
```

```
AUC Train :0.9724
AUC Valid:0.9290
```

When we tune n_estimators , learning_rate we observe better performance for better tuning

When we tried GradientBoosting & Random forest , GradientBoosting gave us a better performance this is because in gradient boosting it uses every previous to perform better over next trees this is reason why it gives the best performance. Even in these 2 cases we have to tune n_estimators , learning_rate properly to get a better accuries.

Hyper parameters that we’ve used for our these model are :

- RandomForestClassifier(n_estimators=230)
  AUC Train :1.0000
  AUC Valid:0.9203

- GradientBoostingClassifier(n_estimators=230,learning_rate=0.06,max_depth=8,subsample=0.5,validation_fraction=0.1,min_samples_leaf=1,min_samples_split=2,random_state=5,n_iter_no_change=20,max_features='log2',verbose=1)
  AUC Train :0.9659
  AUC Valid:0.9232