CS5590: Foundations of Machine Learning

FoML Hackathon: Driver Fault Classification

Sankalp Mittal(cs21mtech12010) and Tamal Mondal(cs21mtech12001)
Kaggle ID: 8854400
Display Name: CS21MTECH12010_CS21MTECH12001

November 15, 2021

1 Problem Statement

A driver accident dataset is given and we need to predict whether the driver involved in the accident is at fault or not. This is a **binary classification problem** and we have **42** features in the given dataset. The training data has **51490** samples and test data has **77235** samples.

2 Data Pre-Processing

2.1 Data Cleaning

The columns "Related Non-Motorist", "Non-Motorist Substance Abuse", "Off-Road Description", "Municipality" and "Circumstance" have more than **80% missing values** , so we have **removed these columns** from training and test data.

For the other object columns like "Vehicle Make", "Vehicle Model", "Vehicle Movement", etc. where there are less missing values, we have **added "UNKNOWN" in place of NULL** (we haven't imputed any value as all these columns are categorical and secondly, they have comparatively less than 20% NULL values).

Additionally we have removed the columns "Report Number", "Local Case Number", "Person ID", "Vehicle ID" and "Driverless Vehicle" through eye-balling as these are **not relevant for prediction**.

2.2 Feature Encoding

We first considered categorical attributes that have less than or equal to 30 unique values and these values are same across training and test set. Such attributes include "Agency Name", "Route Type", "Cross-Street Type" etc. For these attributes, we used **one-hot encoding**(from sklearn).

Next we considered those features that either have large number of unique values, or different values across training and test set, or have an ordinal nature for example the columns like "Light", "Injury Severity", "Vehicle Damage Extent", etc. We have used **ordinal encoding** (from sklearn) for features having large number of unique values as long as they are same across training set and test set. For the features that are not same across training and test set, we considered **manually encoding** them.

2.3 Exploratory Data Analysis

We removed some 4-5 features that were highly correlated (>0.9 correlation). These features were namely those that came after one-hot encoding some of the features.

2.4 Feature Engineering

We have extracted the columns **"Week", "Month", "Date", "Day of Week" and "Time"** (in hours) from the column "Crash Date/Time".

From "Location" column, we have made **"Location1" and "Location2"** columns by just separating the comma separated values present in Location column.

Additionally we have made **"pca0", "pca1" and "Cluster"** column from the columns "Latitude" and "Longitude" using Principal Component Analysis(PCA). After all the preprocessing step we have 230 features in final training and test data.

3 Models and Results

The best model that was selected by us was **CatBoost Classifier**. The second best model was **Light Gradient Boosting Machine (LGBM)**.

We chose these **ensemble** and particularly **gradient boosting** algorithms because they are a go-to methods when dealing with **tabular data** and by the nature of construction, **avoids overfitting**. We also tested the dataset on Random forest but after hyperparameter tuning, the gradient boosting methods namely Catboost and LGBM outperformed them all. LGBM and CatBoost were also **fast compared to XG Boost and Random Forest**.

We first tried Random Forest and it was giving us a consistent **84.3%** accuracy on publicly held test set even after all possible hyperparameter tuning. Switching to XG Boost didn't do much. However, the moment we switched to LGBM, we noticed a boost in accuracy to **86.2%** which we feel wouldn't have been possible with either Random Forest or XG Boost. After we exhausted the hyperparameter tuning of LGBM, we switched to CatBoost which started giving us a consistent accuracy of around **87%** and the best accuracy the we got was **87.5%**. This is the best we were able to achieve using any model.

The reason we tried Random Forest, XGBoost, LGBM and CatBoost models was because bagging and boosting methods perform well in Kaggle challenges(due to tabular data) and most of the times fetch top positions. Within these 4 methods, we had to experiment out the hyperparameters settings and we arrived at the conclusion that **CatBoost Classifier performed best**. The settings that we used in CatBoost classifier was keeping 3500 iterations, learning rate = 0.01, depth of each tree = 8, l2_leaf_reg = 4, "logloss" loss function , border_count = 1, subsample = 0.99, and "Gradient" leaf estimation method.

We have also tried to **combine these different models** to get better generalized predictions. Using the **weighted average** of CatBoost(weight - 0.5), LightGBM(0.3) and Random Forest(0.2), we have got an accuracy of **86.2%**. Also, we tried **majority voting** with a large number of good results from Random Forest, XGBoost, LGBM and CatBoost models which achived an accuracy of **87.3%**.

As for Kaggle submission, we had to finalize two of our best predictions, we have chosen to go with our best **CatBoost classifier predictions** and **the majority vote** of many of our best predictions(which can have less overfitting and generalization error).