Driver Behaviour Analysis

Introduction

Driver behavior analysis is an emerging trend that suits the needs of multiple markets. The most traditional is automation, where detecting inattentive or aggressive driving behaviors is essential to improve safety in vehicles or to switch control in semi-autonomous vehicles. Another potential market is car insurance, which has been interested in monitoring driving activities in order to provide fair insurance premiums to its customers. A third one is fleet management market, where logistics fleet administrators need to know how their vehicles are used and how their drivers behave in order to mitigate potential risks and reduce operational costs.

Overview

In the first week, I teamed up with Sneha Kulkarni for that week's tasks, that were dataset comparison and EDA.

We went through the datasets given to us and Sneha chose 2 datasets which were the most suitable for DBA. I looked for good datasets available on the internet and compared 3 such datasets mutually. We both decided on the parameters for comparison based on the datasets that we chose for ourselves. After this comparison, we went through the datasets to determine which dataset would be best for EDA based on the parameters that the dataset contained and the amount of data available. The dataset which we chose for EDA is carla_dataset. Then we performed EDA and tried to gain insight into the data.

After that I did the tasks individually.

The next week, I implemented various machine learning models to classify the driver behaviour into different classes. After that, I made a comparison of the accuracies of all the models.

EDA

I have performed various manipulations on the dataset to extract some insightful information.

The Dataset

Source: carla_dataset

This data is collected to classify different driver behaviours & extract driver patterns. Carla Simulator platform used to collect data. 6-axis virtual IMU (Inertial Measurment Unit - 3 axis Accelerometer, 3 axis Gyroscope) sensor used to collect data.

The chosen car model is "Seat Leon". 7 different subject drives the car in the same path with 5 turns.

Columns:'Unnamed: 0', 'accelX', 'accelY', 'accelZ', 'class', 'gyroX', 'gyroY', 'gyroZ'

The "class" column contains driver names.

Driver names: [‘mehdi’, ‘apo’, ‘gonca’, ‘onder’, ‘berk’, ‘selin’, ‘hurcan’]

Code Description

For performing EDA I imported the libraries numpy, pandas, seaborn, matplotlib.pyplot.

I uploaded the data from my local device and saved it to full_data_carla.csv.

Then, I read the data into a pandas DataFrame named df.

‘Unnamed: 0’ column only contained index values but reading data into DataFrame created another index column so I dropped the column ‘Unnamed: 0’.

Then I showed information about the columns and some details about the columns with numerical data and the number of null values in each column. Fortunately, no column had any null values.

Then I showed the list of all the drivers.

I created a heatmap to show the correlation between the columns. The columns ‘accelY’ and ‘gyroZ’ had a correlation of 0.68. However, no columns had a correlation near 1. So, I did not remove any column.

I made separate DataFrames for each driver and made line plots for each column.

Observations from the plots

Apo:

- Observations from the Acceleration Graphs: The range is -100 to 100. However, the graph is mostly staying within -25 to 25. This suggests the presence of outliers. It appears that acceleration does not change very frequently but it may be so due to the large range. Hence, no satisfactory conclusion about the frequency of change in acceleration can be made from these graphs.
- Observations from the Gyroscope Graphs: The range is -100 to 100. We can see that the values change very frequently and often become very large, particularly in the case of gyroZ. This suggests frequent harsh turns.
- Hence, we may think that Apo is not a very good driver.

Onder:

- Observations from the Acceleration Graphs: The range for accelX is -15 to 5, for accelY it is -20 to 20 and for accelZ it is 2 to 16. It appears that the value changes very frequently, especially for accelX and accelY but we should note that the range is relatively small. Also, from the graph it seems that the number of outliers is not very large and the value of outliers is not very large as well.
- Observations from the Gyroscope Graphs: The range for gyroX is -20 to 30, for gyroY it is -40 to 30 and for gyroZ it is -80 to 80. It can be seen that the value changes frequently, which suggests harsh and bad turns. Also, the range of gyroZ is very large.
- Hence, we may conclude that Onder is a moderately good driver.

Hurcan:

- Observations from the Acceleration Graphs: The range for accelX is -100 to 20, for accelY it is -100 to 20 and for accelZ it is -30 to 50. We observe that the ranges are large but the graphs are mostly concentrated between -20 and 20. This suggests the presence of outliers with large values. We also see that accelX and accelY change very frequently. This suggests harsh acceleration.
- Observations from the Gyroscope Graphs: The range for gyroX is -60 to 80, for gyroY it is -80 to 60 and for gyroZ it is -100 to 75. We see that the ranges are large but the graphs for gyroX and gyroY mostly stay between -20 and 20 while the graph for gyroZ mostly stays between -50 and 25. This suggests the presence of outliers with large values. Also, the values change quite frequently. All these suggest bad and harsh turns.
- We may conclude from these graphs that Hurcan is not a very good driver.

Gonca:

- Observations from the Acceleration Graphs: The range for accelX is -100 to 20, for accelY it is -100 to 40 and for accelZ it is -20 to 60. The ranges are quite large but the graphs mostly stay between -20 and 20. This observation hints at the presence of large valued outliers. The values for accelY change frequently while not very frequently for accelX and accelZ. All these hint at harsh acceleration.
- Observations from the Gyroscope Graphs: The ranges are -100 to 100. However, the graphs for gyroX and gyroY mostly stay between -25 and 25 while the graph for gyroZ mostly stays between -75 and 25. This observation hints at the presence of large valued outliers. The values vary quite frequently, it is so for gyroZ especially. This hints at bad turns.
- Hence, we may conclude that Gonca is not a very good driver.

Mehdi:

- Observations from the Acceleration Graphs: The range for accelX is -100 to 20, for accelY it is -100 to 75 and for accelZ it is -20 to 80. However, the graphs mostly stay between -20 and 20. This means that there are outliers of very large values. The values change very frequently for accelY and quite frequently for accelX.
- Observations from the Gyroscope Graphs: The ranges are -100 to 100. However, the graphs for gyroX and gyroY usually stay between -25 and 25

while the graph for gyroZ mostly stays between -75 and 50. This suggests large valued outliers for gyroX and gyroY. The values of gyroZ change very frequently, becoming very large at many points. The values of gyro X and gyroY also change quite frequently.

- Hence, we can conclude that Mehdi is not a very good driver.

Berk:

- Observations from the Acceleration Graphs: The range for accelX is -40 to 10, for accelY it is -30 to 20 and for accelZ it is 0 to 30. The graphs for accelX and accelY mostly stay between -20 and 10, that for accelZ mostly stays between 5 and 10. It suggests the presence of outliers. It seems that the graph for accelY changes very frequently while that for accelX changes quite frequently but we must note that the ranges are relatively small.
- Observations from the Gyroscope Graphs: The ranges for gyroX and gyroY are -40 to 40 and that for gyroZ is -75 to 75. The graphs for gyroX and gyroY usually stay between -20 and 20 while that for gyroZ mostly stays between -50 and 25. The graphs vary very frequently but it is to be noted that the ranges are relatively small.
- Hence, we can draw the conclusion from these graphs thar Berk is a moderately good driver.

Selin:

- Observations from the Acceleration Graphs: The range for accelX is -6 to 8, for accelY it is -30 to 20 and that for accelZ is 0 to 17.5. The graph for accelX mostly stays between -4 and 6, that for accelY stays between -20 and 10 and that for accelZ stays between 5 and 12.5. It appears that the values change very frequently, especially for accelX but it must be noted that the ranges are very small.
- Observations from the Gyroscope Graphs: The range for gyroX is -40 to 20, for gyroY it is -30 to 20 and for gyroZ it is -80 to 80. The graphs for gyroX and gyroY mostly stay between -10 and 10 while that for gyroZ mostly stays between -40 and 20. It appears that the values change very frequently but it must be noted that the ranges are relatively small.
- Hence, we may conclude that Selin is a good driver.

The rest of the EDA in the notebook is done by Sneha Kulkarni.

She has plotted histograms for all parameter. She has divided the dataset for each driver and for each smaller dataset, i.e., for each driver she has made boxplots for each column. She has made barplots for mean absolute values of acceleration and angular velocities of each driver. She has made 3D scatter plots for acceleration and gyroscopic values for each driver.

Machine Learning Models

After the required data pre-processing, I have implemented various machine learning models to classify the driver behaviour into three classes. Then I have compared the accuracies of the various models.

Code Description

For classifying driver behaviour, I have imported the libraries pandas, from sklearn preprocessing, from sklearn.model_selection train_test_split, from sklearn.ensemble RandomForestClassifier, from sklearn.metrics accuracy_score, from sklearn.naive_bayes GaussianNB, from sklearn.linear_model LogisticRegression, from sklearn.ensemble ExtraTreesClassifier, from sklearn.svm LinearSVC, from sklearn.preprocessing StandardScaler, from sklearn.ensemble AdaBoostClassifier, from sklearn.ensemble GradientBoostingClassifier, and I have ignored warnings.

I uploaded the data from my local device and saved it to full_data_carla.csv.

Then, I read the data into a pandas DataFrame named df.

‘Unnamed: 0’ column only contained index values but reading data into DataFrame created another index column so I dropped the column ‘Unnamed: 0’.

Then, I created separate DataFrames for each driver.

I calculated the first and third quartile for every column of each new DataFrame.

I created a new column ‘Behaviour’ for each dataset. This column is the target variable. Then I assigned the value ‘Normal’ to all the entries in all the DataFrames.

I assigned the behaviour ‘Drowsy’ to all the entries in a DataFrame where the value of each column is less than the respective first quartile (for the respective DataFrame). For example –

If for row 50 of Apo accelX<apo_ax_1stquar, accelY<apo_ay_1stquar, accelZ<apo_az_1stquar, gyroX<apo_gx_1stquar, gyroY<apo_gy_1stquar, gyroZ<apo_gz_1stquar. Then row 50 is assigned ‘Drowsy’.

I assigned the behaviour ‘Aggressive’ to all the entries in a DataFrame where the value of any column is more than the respective third quartile (for the respective DataFrame). For example –

If for row 50 of Apo accelX>apo_ax_3rdquar, accelY<apo_ay_3rdquar, accelZ<apo_az_3rdquar, gyroX<apo_gx_3rdquar, gyroY>apo_gy_3rdquar, gyroZ<apo_gz_3rdquar. Then row 50 is assigned ‘Aggressive’.

Then I concatenated all these 7 DataFrames into a single DataFrame result.

The columns ‘class’ and ‘Behaviour’ are of object datatype. Hence, I encoded them using label encoding. I also made dictionaries classes and Behaviour which contain the original value against encoded value of class and Behaviour columns respectively.

I stored values of the DataFrame in to a variable data using the <DataFrame name>.values() function to perform train test split on it and made X_train, y_train, X_test and y_test.

X_train and X_test contain all the features while y_train and y_test contain the target variable. X_train and y_train were used to train the models. X_test and y_test were used to test the models.

The sizes of the training sets and the testing sets was in the ratio 2:1.

I used StandardScaler to scale all the features.

Models

I used the following models to classify driver behaviour using my dataset.

Random Forest Classifier:

Random forest is a Supervised Machine Learning Algorithm that is used widely in Classification and Regression problems. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression.

One of the most important features of the Random Forest Algorithm is that it can handle the data set containing continuous variables as in the case of regression and categorical variables as in the case of classification. It performs better results for classification problems.

Result: Accuracy = 99.396 %

Gaussian Naïve Bayes Model:

Gaussian Naïve Bayes is a variant of Naïve Bayes that follows Gaussian normal distribution and supports continuous data.

Naive Bayes are a group of supervised machine learning classification algorithms based on the Bayes theorem. It is a simple classification technique, but has high functionality. They find use when the dimensionality of the inputs is high. Complex classification problems can also be implemented by using Naive Bayes Classifier.

Result: Accuracy = 36.506 %

Logistic Regression Model:

Logistic regression is a classification algorithm.

It is intended for datasets that have numerical input variables and a categorical target variable that has two values or classes. Problems of this type are referred to as binary classification problems.

Logistic regression is designed for two-class problems, modeling the target using a binomial probability distribution function.

Logistic regression, by default, is limited to two-class classification problems. Some extensions like one-vs-rest can allow logistic regression to be used for multi-class classification problems, although they require that the classification problem first be transformed into multiple binary classification problems.

Result: Accuracy = 77.697 %

Extra Trees Classifier:

Extremely Randomized Trees, or Extra Trees for short, is an ensemble machine learning algorithm.

Specifically, it is an ensemble of decision trees and is related to other ensembles of decision trees algorithms such as bootstrap aggregation (bagging) and random forest.

The Extra Trees algorithm works by creating a large number of unpruned decision trees from the training dataset. Predictions are made by averaging the prediction of the decision trees in the case of regression or using majority voting in the case of classification.

Unlike bagging and random forest that develop each decision tree from a bootstrap sample of the training dataset, the Extra Trees algorithm fits each decision tree on the whole training dataset.

Like random forest, the Extra Trees algorithm will randomly sample the features at each split point of a decision tree. Unlike random forest, which uses a greedy algorithm to select an optimal split point, the Extra Trees algorithm selects a split point at random.

Result: Accuracy = 97.310 %

Linear Support Vector Classifier:

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

Result: Accuracy = 78.017 %

AdaBoost Classifier:

Ada-boost or Adaptive Boosting is one of ensemble boosting classifier proposed by Yoav Freund and Robert Schapire in 1996. It combines multiple classifiers to increase the accuracy of classifiers. AdaBoost is an iterative ensemble method. AdaBoost classifier builds a strong classifier by combining multiple poorly performing classifiers so that you will get high accuracy strong classifier. The basic concept behind Adaboost is to set the weights of classifiers and training the data sample in each iteration such that it ensures the accurate predictions of unusual observations. Any machine learning algorithm can be used as base classifier if it accepts weights on the training set. AdaBoost uses Decision Tree Classifier as default Classifier.

Result: Accuracy = 75.986 %

Stochastic Gradient Boosting Classifier:

Gradient boosting classifiers are a group of machine learning algorithms that combine many weak learning models together to create a strong predictive model. Decision trees are usually used when doing gradient boosting.

In the case of Gradient Boosting Machines, every time a new weak learner is added to the model, the weights of the previous learners are frozen or cemented in place, left unchanged as the new layers are introduced. This is distinct from the approaches used in AdaBoosting where the values are adjusted when new learners are added.

The power of gradient boosting machines comes from the fact that they can be used on more than binary classification problems, they can be used on multi-class classification problems and even regression problems.

Taking random subsamples of the training data set, a technique referred to as stochastic gradient boosting, can also help prevent overfitting. This technique essentially reduces the strength of the correlation between trees.

Result: Accuracy = 97.426 %

Comparing Accuracies

S.No.	Machine Learning Models	Accuracy (%)
1	Random Forest Classifier	99.396
2	Gaussian Naïve Bayes Model	36.506
3	Logistic Regression Model	77.697
4	Extra Trees Classifier	97.310
5	Linear Support Vector Classifier	78.017
6	AdaBoost Classifier	75.986
7	Stochastic Gradient Boosting Classifier	97.426