Machine Learning Engineer Nanodegree

Capstone Project

Mohit Sharma
May 6th, 2019

I. Definition

(approx. 1-2 pages)

Project Overview

Road safety rules and regulations are designed to prevent the citizens from fatal incidents. Although policies are in place, we observe negligent behaviour of the drivers which lead to serious injuries or death crashes. It is of utmost interest of the authorities to understand and analyse human behaviour to take necessary corrective and preventive actions. Also, it is believed that understanding the driving behaviours could help in terms of fuel economy improvement and emission reduction which are strongly influenced by driving conditions and drivers' driving styles.

The major stakeholders are the citizens, road transport authorities, Insurers and Researchers/Data service providers.

Problem Statement

In order to design a driving assistance system there is a need to get an understanding of the data on the driving patterns and broadly distinguish bad drivers from good ones.

Develop a supervised learning model using classification algorithm to predict which drivers are aggressive in nature using the captured features. Currently, the drivers are labelled into 3 categories which include Normal, Vague and Aggressive. As we are mostly interested in correctly identifying the aggressive drivers, we would convert this problem from multilabel to binary classification by clubbing the Vague and Normal driving styles.

The data is provided in three different files.

Training Data – The dataset contains 12994 observations and 5 variables which include Length of vehicle in cm, weight of vehicle in kg, Number of axles etc. Out of 12994 approximately 21.3% of the drivers are labelled as aggressive, 49.4% as Normal and rest 30%as vague.

Train_WeatherData – The dataset contains 162,566 observations and 9 variables.

Train_Vehicletravellingdata – The dataset contains 162,566 observations and 10 variables. The dataset provides information about the preceding vehicle and the road with respect to the weather.

So, before we start to explore the data, we need to combine these three datasets to ensure we have every feature in one dataset for convenience. While we do this, we also create new variables which we believe can be of importance.

Metrics

For this problem, we believe the following metrics can be explored to identify the best Model.

Accuracy – It talks about, overall, how often the classifier is correct. The mathematical formula which is used to calculate Accuracy is **(TP+TN)/total**

Recall – It is the number of correct results divided by the number of results that should have been returned. In binary classification, recall is called sensitivity. It can be viewed as the probability that a relevant document is retrieved by the query.

The mathematical formula which is used to calculate Recall is **actual yes/total yes**

As our dataset is unbalanced with just ~21% examples of aggressive drivers we want to ensure that our model can predict as much aggressive drivers as possible from the actual aggressive drivers and that is why Recall as a metric is important.

II. Analysis

(approx. 2-4 pages)

Data Exploration

1. The first step in the complete dataset which we generated by combining the three datasets we had initially is to check for each variable type and ensure that correct types are assigned to each of the variables. We have 25 numerical variables and categorical variables are 6.

2. We noticed that total number of missing values or blanks in the dataset constitutes ~2% of the over all dataset and thus we choose to delete all such observations.

3. The summary statistics of numerical variables also reviled that some of the numerical variables have outliers. We also looked into histograms and boxplots for doing outlier analysis. Histograms and boxplot related information and charts are provided in next section.

Snapshot of decriptive statistics for first few numerical variables:

	TotalSwitches	TotalJourneyTime	AvgSpeed	MaxSpeed	AvgSpeedPrecedingVehicle	MaxSpeedPrecedingVehicle	AvgTimeGapPreceding
count	12657.000000	12657.000000	12657.000000	12657.000000	12657.000000	12657.000000	12657.000000
mean	4.451134	8.671565	83.718970	97.416923	83.739038	97.429644	207.472387
std	3.956084	30.841301	5.662169	10.397439	5.615351	10.384553	215.625538
min	0.000000	0.000000	44.000000	45.000000	42.000000	42.000000	3.000000
25%	1.000000	7.000000	81.000000	90.000000	81.000000	90.000000	78.000000
50%	4.000000	9.000000	84.000000	97.000000	84.000000	97.000000	127.000000
75%	6.000000	10.000000	87.000000	103.000000	87.000000	103.000000	248.000000
max	30.000000	3361.000000	130.000000	161.000000	120.000000	161.000000	1642.000000

4. Here is the list of final variables which have been correctly classified into desired data type.

```
Out[13]: ID                              object
         TotalSwitches                    int64
         TotalJourneyTime               float64
         AvgSpeed                       float64
         MaxSpeed                         int64
         AvgSpeedPrecedingVehicle       float64
         MaxSpeedPrecedingVehicle         int64
         AvgTimeGapPreceding            float64
         MaxTimeGapPreceding            float64
         MinTimeGapPreceding            float64
         MedianTimeGapPreceding         float64
         AvgLengthvehiclePreceding      float64
         MaxLengthvehiclePreceding        int64
         MinLengthvehiclePreceding        int64
         MedianLengthvehiclePreceding   float64
         AvgWeightvehiclePreceding      float64
         MaxWeightvehiclePreceding        int64
         MinWeightvehiclePreceding        int64
         MedianWeightvehiclePreceding   float64
         RoadCondition                 category
         RateOfSwitch                   float64
         AirTemp                        float64
         Precipitation                 category
         PrecipitationIntensity        category
         RelHumidity                    float64
         WindDirection                  float64
         WindSpeed                      float64
         Weekday                       category
         VehicleLength(cm)                int64
         VehicleWeight(kg)                int64
         NumAxles                      category
         DrivingStyle                  category
         dtype: object
```

5. All the categorical variables were converted to dummy variables by using get_dummies() function from pandas.

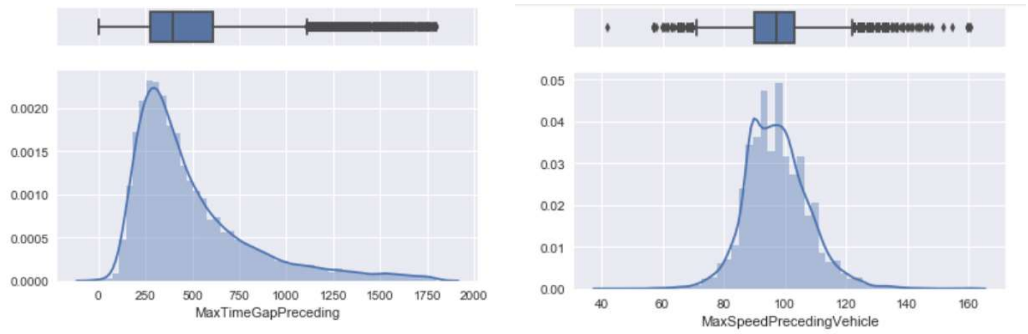
6. Based upon the outlier analysis we decided to take log transformation of the 11 variables. All the variables which we found to be skewed were considered for the log transformation.
   Below is the list of all the 11 variables:

["TotalSwitches", "AvgTimeGapPreceding", "MaxTimeGapPreceding", "MinTimeGapPreceding", "MedianTimeGapPreceding", "AvgLengthvehiclePreceding", "AvgWeightvehiclePreceding", "MedianWeightvehiclePreceding", "VehicleWeight(kg)", "VehicleLength(cm)"]

Exploratory Visualization

As mentioned earlier, histograms and boxplots were draw for each individual variable to understand the distribution. As the number of charts to analyse were very large I combined histograms and boxplots to be shown in same figure. These graphs were drawn using seaborn module in python. This analysis was really helpful in taking a call on which variables needs transformations.

Providing some of the charts for your reference. Among the charts shown below the chart on **left-hand side variable was chosen for log transformation**.



The above mentioned criteria was used to decide on variables which needed transformation.

Algorithms and Techniques

Before building the model following Techniques were implemented:

1. All numerical variables where scaled using MinMaxScaler() function. It was done because the scale of the variables was very different.

2. The data was divided into test and train using train_test_split() function. About 70% of the data was taken as train and 30% of it was taken as test.

3. Some 10 different classification algorithms (all capable of binary classification) where on the training dataset and accuracy, recall, precision and f1Score like metrics were calculated.

4. Among these models the GaussianNB model was finally selected.

    a. GaussianNB Algorithm
        i. Overall Accuracy: Train – 70.0% , Test – 70.2%
        ii. Overall Recall: Train – 67.6% , Test – 66.2%

5. For now, Gaussian looks like the best model. We don't have any hyperparameters to tune the model further for this algorithm.

Benchmark

According to one the papers which we read online the model accuracy was ~70%. We can consider this as a benchmark. However, the paper does not provide any information on the recall rate.

Link to the https://ieeexplore.ieee.org/document/5309718

Unfortunately, I am not sure if the data they used, and I am using is similar or not. I got this data from one of my friends in the industry to be solved as my capstone project. According to their claims they had an accuracy of ~65% with 62% recall rate. They had mostly used original variables and no scaling of the variables were done.

Considering my friend has told me I guess the new features and data transformations has resulted into model which shows an improvement of ~5% in terms of both accuracy and recall.

III. Methodology

(approx. 3-5 pages)

Data Preprocessing

Data pre-processing was the most critical part. We expanded the original 16 features to close to 50 variables. This was done by deriving some new variables from the datasets. The information of all these variables is as given below:

Here is the list of variables which we created from travelData.csv along with the code snippet.

1. SwitchCount – How many switches did the driver made during the travel distance.
2. TotalJourneyTime – Total time taken for the journey.

3. RateOfSwitch – What is the rate of overtalking the other vehicle.

4. AvgSpeed – Average speed of the vehicle

5. MaxSpeed – Maximum speed of the vehicle

6. AvgSpeedPrecedingVehicle – Average speed of the preceding vehicle

7. MaxSpeedPrecedingVehicle – Max speed of the preceding vehicle

8. AvgTimeGapPreceding – Average gap with the preceding vehicle in seconds

9. MaxTimeGapPreceding – Max gap with the preceding vehicle in seconds

10. MinTimeGapPreceding – Min gap with the preceding vehicle in seconds

11. MedianTimeGapPreceding – Median gap with the preceding vehicle in seconds
12. AvgLengthvehiclePreceding – Average length of the preceding vehicle in cm

13. MaxLengthvehiclePreceding – Max length of the preceding vehicle in cm

14. MinLengthvehiclePreceding – Min length of the preceding vehicle in cm

15. MedianLengthvehiclePreceding – Median length of the preceding vehicle in cm
16. AvgWeightvehiclePreceding – Average weight of the preceding vehicle in kg

17. MaxWeightvehiclePreceding – Max weight of the preceding vehicle in kg

18. MinWeightvehiclePreceding – Min weight of the preceding vehicle in kg

19. MedianWeightvehiclePreceding – Median weight of the preceding vehicle in kg

20. RoadCondition – Existing variable just took the first value of each row for each driver data.

```
for i in unique_id:
    ID.append(i)
    dfTemp = traveldata[traveldata.ID == i]

    JourneyTime = pd.to_datetime(dfTemp.V1.iloc[dfTemp.shape[0]-1]) - pd.to_datetime(dfTemp.V1.iloc[0])
    TotalJourneyTime.append(JourneyTime)

    AvgSpeed.append(round(dfTemp.V4.mean(),0))
    MaxSpeed.append(round(dfTemp.V4.max(),0))

    AvgSpeedPrecedingVehicle.append(round(dfTemp.V8.mean(),0))
    MaxSpeedPrecedingVehicle.append(round(dfTemp.V8.max(),0))

    AvgTimeGapPreceding.append(round(dfTemp.V11.mean(),0))
    MaxTimeGapPreceding.append(round(dfTemp.V11.max(),0))
    MinTimeGapPreceding.append(round(dfTemp.V11.min(),0))
    MedianTimeGapPreceding.append(round(dfTemp.V11.median(),0))

    AvgLengthvehiclePreceding.append(round(dfTemp.V10.mean(),0))
    MaxLengthvehiclePreceding.append(round(dfTemp.V10.max(),0))
    MinLengthvehiclePreceding.append(round(dfTemp.V10.min(),0))
    MedianLengthvehiclePreceding.append(round(dfTemp.V10.median(),0))

    AvgWeightvehiclePreceding.append(round(dfTemp.V9.mean(),0))
    MaxWeightvehiclePreceding.append(round(dfTemp.V9.max(),0))
    MinWeightvehiclePreceding.append(round(dfTemp.V9.min(),0))
    MedianWeightvehiclePreceding.append(round(dfTemp.V9.median(),0))
```

```
    switch_count = 0
    lane = dfTemp["V3"]
    currentValue = []
    nextValue = []
    for j in range(0, len(lane)-1):
        if j < len(lane):
            #print(j)
            currentValue =  dfTemp.V3.iloc[j]
            nextValue = dfTemp.V3.iloc[j+1]
            if nextValue != currentValue:
                switch_count = switch_count + 1
    SwitchCount.append(switch_count)

SwitchCountDF = pd.DataFrame({'ID':ID, 'TotalSwitches': SwitchCount, 'TotalJourneyTime': TotalJourneyTime,
                              'AvgSpeed':AvgSpeed, 'MaxSpeed': MaxSpeed, 'AvgSpeedPrecedingVehicle':AvgSpeedPrecedingVehicle, 'Ma
                              'AvgTimeGapPreceding': AvgTimeGapPreceding, 'MaxTimeGapPreceding': MaxTimeGapPreceding,
                              'MinTimeGapPreceding': MinTimeGapPreceding, 'MedianTimeGapPreceding':MedianTimeGapPreceding,
                              'AvgLengthvehiclePreceding': AvgLengthvehiclePreceding, 'MaxLengthvehiclePreceding': MaxLengthvehic
                              'MinLengthvehiclePreceding': MinLengthvehiclePreceding, 'MedianLengthvehiclePreceding': MedianLengt
                              'AvgWeightvehiclePreceding': AvgWeightvehiclePreceding, 'MaxWeightvehiclePreceding': MaxWeightvehic
                              'MinWeightvehiclePreceding': MinWeightvehiclePreceding, 'MedianWeightvehiclePreceding': MedianWeigh
                              'RoadCondition': RoadCondition}, columns = ['ID', 'TotalSwitches', 'TotalJourneyTime', 'AvgSpeed',
                                        'AvgSpeedPrecedingVehicle', 'MaxSpeedPrecedingVehicle', 'AvgTimeGapPreceding',
                                        'MaxTimeGapPreceding', 'MinTimeGapPreceding', 'MedianTimeGapPreceding',
                                        'AvgLengthvehiclePreceding', 'MaxLengthvehiclePreceding', 'MinLengthvehiclePreceding',
                                        'MedianLengthvehiclePreceding', 'AvgWeightvehiclePreceding', 'MaxWeightvehiclePreceding'
                                        'MinWeightvehiclePreceding', 'MedianWeightvehiclePreceding', 'RoadCondition']
                             )
```

We noticed that some of the drivers did not made any switches between the lanes and this when we calculated the Rate of Switch some NaN were generated these were taken care by the following code.

```
SwitchCountDF['TotalJourneyTime'] = SwitchCountDF['TotalJourneyTime'].astype('timedelta64[m]')
SwitchCountDF['RateOfSwitch'] = round(SwitchCountDF['TotalSwitches']/SwitchCountDF['TotalJourneyTime'],2)
SwitchCountDF['RateOfSwitch'][SwitchCountDF['RateOfSwitch'] == -inf] = 0
```

The weather data had some blanks and missing values we wanted to know what percent of this data is missing. For the weather data we created

1. Weekday – It is a number suggesting which day of the week it is.

We then used the following code to get the most common values for each of the features.

```
ID = []
AirTemp = []
Precipitation = []
PrecipitationIntensity = []
RelHumidity = []
WindDirection = []
WindSpeed = []
Weekday = []
for i in unique_id:
    dfTemp = weatherdata[weatherdata.ID == i]
    if dfTemp.shape[0] != 0:
        ID.append(i)
        AirTemp.append(Counter(dfTemp['V12']).most_common(1)[0][0])
        Precipitation.append(Counter(dfTemp['V13']).most_common(1)[0][0])
        PrecipitationIntensity.append(Counter(dfTemp['V14']).most_common(1)[0][0])
        RelHumidity.append(Counter(dfTemp['V15']).most_common(1)[0][0])
        WindDirection.append(Counter(dfTemp['V16']).most_common(1)[0][0])
        WindSpeed.append(Counter(dfTemp['V17']).most_common(1)[0][0])
        Weekday.append(dfTemp.Weekday.iloc[0])

weatherFinal = pd.DataFrame({'ID': ID, 'AirTemp': AirTemp, 'Precipitation': Precipitation,
                             'PrecipitationIntensity': PrecipitationIntensity, 'RelHumidity': RelHumidity,
                             'WindDirection': WindDirection, 'WindSpeed': WindSpeed, 'Weekday': Weekday },
                            columns = ['ID','AirTemp', 'Precipitation', 'PrecipitationIntensity', 'RelHumidity',
                             'WindDirection', 'WindSpeed', 'Weekday'])

weatherFinal['PrecipitationIntensity'] = weatherFinal['PrecipitationIntensity'].apply(lambda x: "Missing" if x == " " else x)
```

In [67]:
```
from sklearn.preprocessing import MinMaxScaler

# Initialize a scaler, then apply it to the features
scaler = MinMaxScaler() # default=(0, 1)

finalRawData[numerical] = scaler.fit_transform(finalRawData[numerical])

# Show an example of a record with scaling applied
finalRawData.head(n = 5)
```

Out[67]:

	ID	TotalSwitches	TotalJourneyTime	AvgSpeed	MaxSpeed	AvgSpeedPrecedingVehicle	MaxSpeedPrecedingVehicle	AvgTimeGapPreceding
0	DR_24526	0.133333	0.002678	0.500000	0.491379	0.576923	0.420168	0.039658
1	DR_30052	0.233333	0.002975	0.441860	0.396552	0.564103	0.504202	0.082367
2	DR_40928	0.133333	0.002678	0.476744	0.482759	0.512821	0.495798	0.095790
3	DR_66033	0.200000	0.002975	0.453488	0.379310	0.538462	0.445378	0.070165
4	DR_45266	0.166667	0.002678	0.453488	0.379310	0.525641	0.453782	0.065284

5 rows × 32 columns

We also created dummy variables using get_dummies() function from pandas package.

```
# TODO: One-hot encode the 'features_log_minmax_transform' data using pandas.get_dummies()
features_final = pd.get_dummies(features)
features_final.head()
```

The algorithm used mostly required the dummy variables to be converted into one-hot notation and that was done.

Implementation

While running the scaler algorithm for the first time we faced some challenges. The Algorithm was throwing an error stating that some of the values are too high, or blank or have very large value. If took some ~2 hours to learn that this error was generated because of the presence of Inf values present in the dataset.

These Inf values were generated in the new variables like RateOfSwitch as not all the drivers made the switch between the lanes during their journey. We finally identified all such instances and replaced them with the 0.

Some of the other challenges faced were related to plotting the graphs. However, a quick research on google helped me in overcoming the same in no time.

Refinement

To get better results at the initial state I have ensured that we bring all the variables on the same scale, we have also taken data transformations of the features based upon the data analysis and exploration.

I did try to build the model using principal components, however the model did not see any improvement and thus we scrapped that model and went ahead with the initial model.

IV. Results

(approx. 2-3 pages)

Model Evaluation and Validation

A. GaussianNB Algorithm
    i. Overall Accuracy: Train – 70.0% , Test – 70.2%
    ii. Overall Recall: Train – 67.6% , Test – 66.2%

The model results were tested on test dataset. We also checked the confusion matrix and the results seem to hold and sustain.

```
pd.crosstab(y_train, train_predictions, rownames=['True'], colnames=['Predicted'], margins=True)
```

Out[62]:

Predicted	0	1	All
True			
0	4929	2039	6968
1	613	1278	1891
All	5542	3317	8859

```
In [90]: pd.crosstab(y_test, test_predictions, rownames=['True'], colnames=['Predicted'], margins=True)
```

Out[90]:

Predicted	0	1	All
True			
0	2138	858	2996
1	271	531	802
All	2409	1389	3798

The model results are robust and have been tested on test dataset.

Justification

Unfortunately, we did not have a benchmark other than what I know from my friend. Considering those results, I find the final solution and its results have improved substantially.

V. Conclusion

(approx. 1-2 pages)

Free-Form Visualization

Yes I did built enough visualization to help me build a better model.

Reflection

The process for building the project was fairly simple.

1. Explore and understand all the features.
2. Create new variables and combine the data from three different sources into one. While doing so we also created the new variables, transformed few and ignored those which have all the values as unique.
3. In the final rawdata set which we got was then used to explore further by generating statistical summaries, visualizations and treating values which were generated as part of the calculations. One such instance was generation of NaN and Inf for some cases.
4. The final features where then scaled to ensure they all are on the same scale.
5. The dataset was then divided into train and test.
6. Model was built using GassianNB and tested on test dataset to ensure that model which we have created is robust.
7. Thinking of new variables and dealing with unexpected error generated due to NaN and Inf were some of the challenges which we faced during the model building.
8. Building visualization which is easy to understand and explore was another challenging part of this exercise.
9. Provided the data in the given format we could use this model to predict the DrivingStyle.

Improvement

To further improve the models, I did try to run the xgboost algorithm. However, I did not see an improvement in the recall as compared to the current model. I also tried

using PCA and again model did not se any further improvements. I think new variables could help improve model. Also, wished to create two more feature from the current features but not sure how to create them. These new variables will capture the speed of the vehicle and the preceding vehicle at the time of overtaking.