Project report on

DRIVER BEHAVIOUR PROFILING

Submitted by:

Akshath Kaushal (IMT2018501) *akshath.kaushal@iiitb.org*

Ishaan Sachdeva (IMT2018508) *ishaan.sachdeva@iiitb.org*

Course:

T2-20-NC 812 / Internet of Things

Submitted to:

Prof. Jyotsna Bapat

International Institute of Information Technology Bangalore

The drive folder containing the apks, notebooks etc. can be found here.

Objective

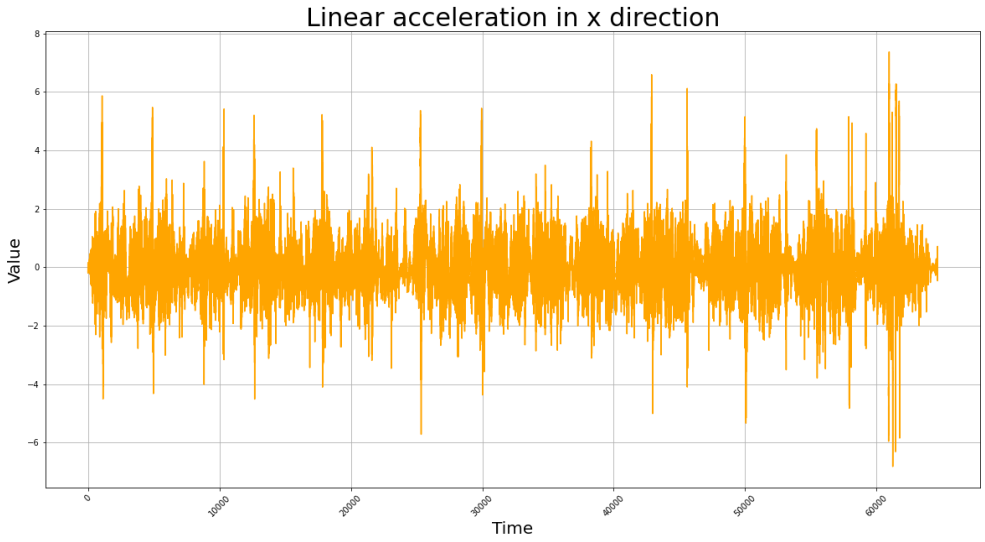
The objective was to create a robust prediction system that can predict the driving profile of the driver of a car. The initial goal was to develop a machine learning model that can take the values from various sensors of a connected mobile device and generate the state of driving based on the values of those sensors. Further step was to present the values in an effective and interactive way so to take necessary steps based on the same. The main objective behind this project was to find an innovative and affordable solution to curb the problem of road accidents globally. Since this was supposed to be implemented without any sophisticated hardware, we have to keep in mind constraints like device configuration, processing speed etc.

The dataset

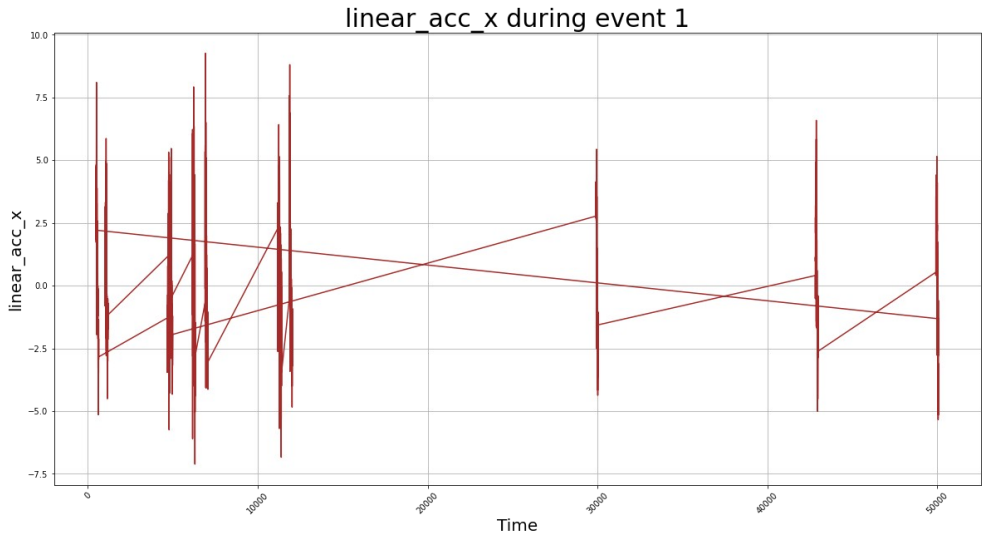
The dataset that was used to train various models can be found here. It uses a 2011 Honda Civic and a smartphone (Motorola XT1058) with Android version 5.1 to collect the data from the accelerometer, gyroscope and magnetometer sensors of the smartphone. However, due to the limitation of the available devices, we have used only the data from gyroscope and accelerometer to generate predictions. The dataset is available in four parts, each collected on different dates. Here are the initial values of the combined dataset:

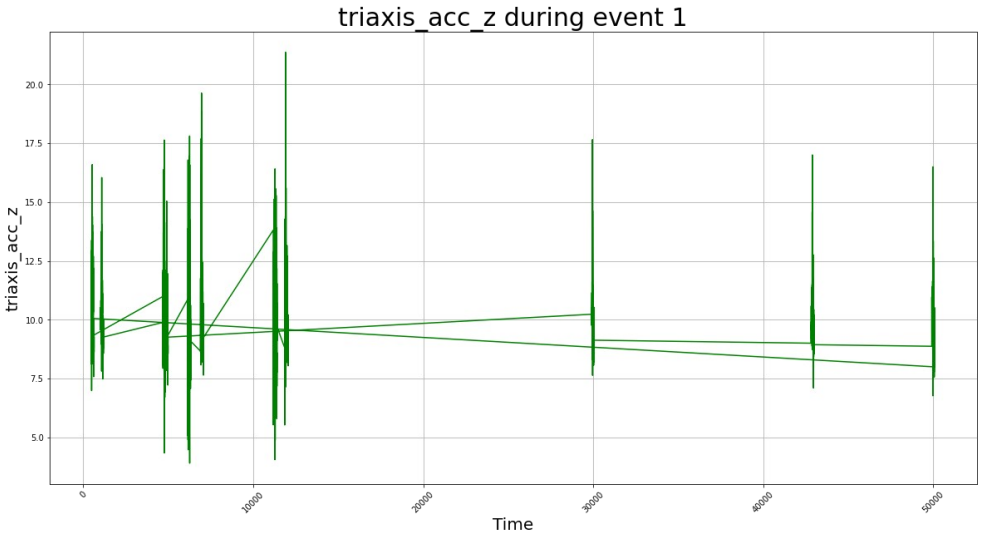
	timestamp	linear_acc_x	linear_acc_y	linear_acc_z	triaxis_acc_x	triaxis_acc_y	triaxis_acc_z	gyro_x	gyro_y	gyro_z	Time	Event
0	15/05/2016 16:03:47	0.336558	-0.169031	-0.148012	0.336558	-0.169031	9.658637	0.031439	0.360042	0.009180	0.000000	7
1	15/05/2016 16:03:47	2.022178	-0.125877	0.141352	2.022178	-0.125877	9.948004	0.235446	0.680577	-0.039432	0.010163	7
2	15/05/2016 16:03:47	0.777634	-0.088642	2.002689	0.777634	-0.088642	11.809340	0.118159	0.084886	-0.087033	0.029879	7
3	15/05/2016 16:03:47	-0.695542	-0.127862	-1.134230	-0.695542	-0.127862	8.672421	-0.176048	-0.288289	0.070482	0.049504	7
4	15/05/2016 16:03:47	-1.864220	0.843311	-1.783103	-1.864220	0.843311	8.023548	-0.137060	-0.123618	0.051219	0.069129	7

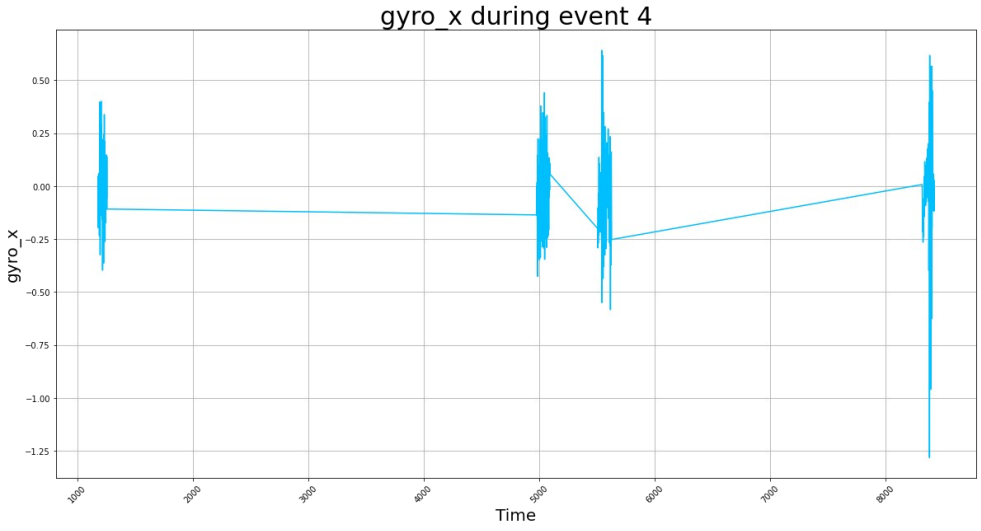
Also, the ground truth was not labelled in the dataset so we have to do that manually. There were a total of seven states of the vehicle, namely, *aggressive_left_curve*, *aggressive_right_curve*, *aggressive_left_lane*, *aggressive_right_lane*, *aggressive_acceleration*, *aggressive_breaking and non_aggressive_event.* The dataset did not have any null values and did not need any initial preprocessing except the labelling. For further clarity on the data, we plotted various kinds of plots to get insights about the values. First, we plotted the combined values of sensor measurements with time, and got plots like this:



Next, we plotted individual parameters with time, for a particular event. This gave us a lot of information about the change in values and helped us model accordingly:







Note: Rest of the plots are a part of preprocessing and can be found in the model training notebook.

Modelling and Procedure

For the modelling step, we decided to approach the problem with traditional methods like xgboost etc. Initially, we excluded the non_aggressive data and got an accuracy of about 74%.(link to the model) To improve upon this, we included the non_aggressive data and got an acuracy of about 94.7% (link to the model). Still, we though of improving further. As seen from the plots, it is clearly visible that the data is biased and the number values of non_agressive_event are quite high as compared to the other events. So we decided to go with class balancing by doing over sampling on the data in order to remove the bias. (link to the model)

Now, once the data was ready, we implemented XGBoost. For the parameters, we ran a simple RandomizedGridSearchCv. Here are the final parameters:

Parameter	Final value
tree_method	gpu_hist
Subsample	0.9
sampling_method	Uniform
Predictor	gpu_predictor
max_depth	15
max_bin	2048
Lambda	3
grow_policy	Depthwise
Eta	0.0005

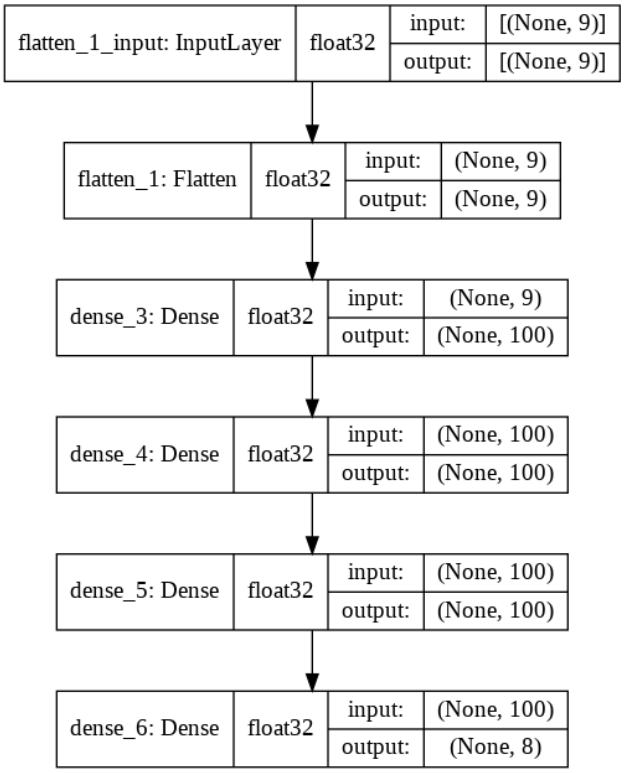
These parameters gave us the final training accuracy of about 97.7% and validation accuracy of 98.2%.

Model	Validation Accuracy
XGBoost without non_aggressive events	74.6%
XGBoost with non_aggressive events	94.2%
XGBoost with non_aggressive events and class balancing	98.2%

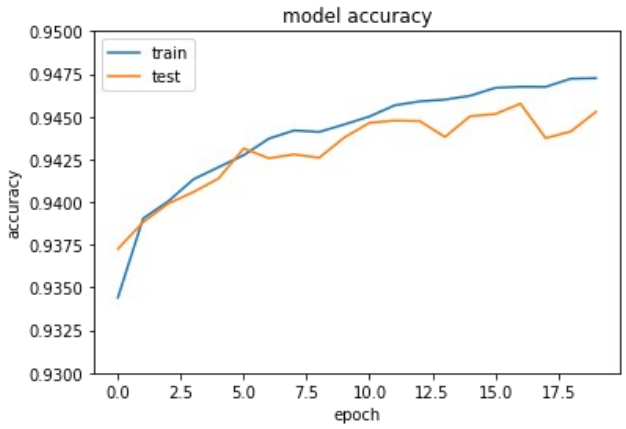
Next, we further tested different traditional ML models on the dataset (no class balancing) to get the following accuracy scores

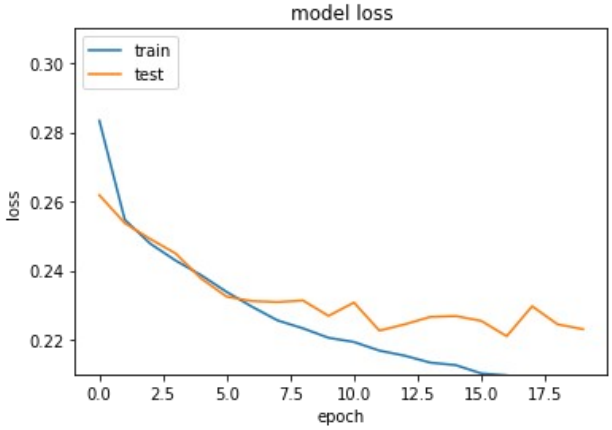
Model	Accuracy score
Decision Tree	94.03%
Random Forest	94.29%
LightGBM	94.26%

Next, we decided to implement modern ML methods like neural nets. Here is the layout of the Neural Net that we implemented

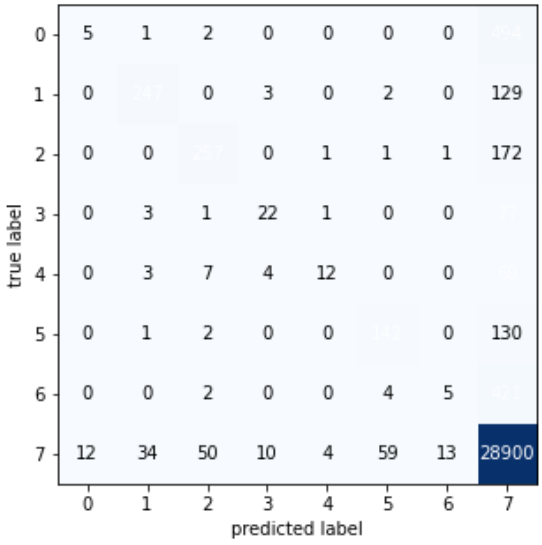


This model gave us an accuracy of 94.65% on the training dataset and 94.53% on the validation dataset.





Confusion matrix



From here we can see the miss detections (ie aggressive events) are high since the data set is biased.

Classification report

	precision	recall	f1-score	support
0	0.29	0.01	0.02	502
1	0.85	0.65	0.74	381
2	0.80	0.59	0.68	432
3	0.56	0.21	0.31	104
4	0.67	0.13	0.21	95
5	0.68	0.52	0.59	275
6	0.26	0.01	0.02	432
7	0.95	0.99	0.97	29082
accuracy			0.95	31303
macro avg	0.63	0.39	0.44	31303
weighted avg	0.92	0.95	0.93	31303

Precision = TP/TP+FP denominator tells the total no of events which the model predicted aggressive(positive) and the ratio tells us, out of the events detected to be aggressive how many were actually aggressive

Recall = TP/TP+FN ie. out of total aggressive events, how many were detected correctly.

Since for some events the data samples in the dataset are less therefore the recall is low and due to the same reason ie biased dataset, there are miss detections because of which the precision of some events are low.

Overall the accuracy of neural network is 95%

Testing

There are a plethora of apps on android and ios which with help of accelerometer and gyroscope values try to track are physical activity for example in google fit.

Here we have tried to mimic the same behaviour. We are collecting the accelerometer and gyroscope values in real time and our ML model based on the values is predicting whether aggressive events are detected.

For the testing of our models, we implemented two separate approaches:

1) For the traditional ML algorithm, we decided to do the processing of a more complex machine than a smartphone as in the long term, the smartphone processor might become overloaded due to te huge amount of data. Hence, we implemented a unique way to obtain the data and send it to a pc for processing and generating predictions. Firstly, we made a mobile app in Java that enables us to get the values from sensors and display and send those values to backend. For the backend, we used google sheets as a database and used google APIs to obtain the data from mobile sensors and populate the google sheet through google forms. Here is the gist of how this is done:

1) The mobile app captures the sensors values.

2) The values are sent to a google form whose URL is already embedded in the app's code.

3) The form captures the values and sends them to a linked spreadsheet.

4) This spreadsheet is captured by the model and is parsed using Pandas library.

The final predictions can be seen in the testing notebook whose link is attached.

2) For the Neural net, we developed a separate app in which the data collection and prediction is done. There is no need to send data to a separate device. It collects the data and based on the number of aggressive events, score of driver is calculated. For the demo the testing time is set to 15 seconds and threshold for a very poor score is 5.

One can also view the scores of previous trips in the report section in the app.

Future scope

There are lot of scope of such applications

1) Ola and uber can this to track the driver's driver behaviour.

2) Rider can use it to see the driver's driver behaviour and sound him whether that the driver is drunk.

3) With the new advancements in the driverless cars, this application can prove to be a very effective method to test the algorithms of the car and can provide vital data for the improvement of the same.

Future improvements

1) Can be integrated with maps for better understanding.

2) Can also take velocity into account to better analyse the aggressive behaviour.

3) More data samples can be collected.

4) The design of algorithm can be improved.

5) Sensor value collected are raw sensor values which could be noisy processing of sensor values ie low pass filtering and then feeding to the algorithm will yield better results.

Acknowledgements

We would like to thank Prof. Jyotsna Bapat and the TAs, Sharavari NP and Pradnya Gaonkar for their continuous support for providing excellent guidance that enabled us to explore new horizons and approaching the goal in a new and innovative way.