Universiteit Leiden

Master Computer Science

Detecting Critical Driving Events using Messy Smartphone Sensor Data

Name:	Heba Mazhar
Student ID:	s2093340
Date:	01/10/2019

Specialization : Advanced Data Analytics

1st supervisor : Thomas Bäck
2nd supervisor : Hao Wang

Master’s Thesis in Computer Science

Leiden Institute of Advanced Computer Science
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

Road accidents and injuries are the 9th leading cause of death in the world. [1] Road safety is an important area where technology can profoundly help. Using smartphones while driving has become a daily norm from the past few years. Smartphone applications provide us with abundant information on the road to help us reach from point A to point B, in the shortest time, avoiding traffic and speed cameras.

What if the smartphones could help us detect rash driving maneuvers on roads and make our driving experience safer altogether?

Smartphones are equipped with sensors that can be used to derive insights about the driver behind the wheel. This thesis focuses on detecting critical driving events that happen on the road using sensors available in our smartphones using machine learning algorithms. The critical driving event this thesis looks at in depth are sharp corners and harsh brakes leveraging the gyroscope and accelerometer smartphone sensor data technology.

This research describes methods to handle noisy sensor data and extract insights from it to best serve our purpose. Due to time constraints of the thesis, the algorithm proposed in this research could not be tested real time on a smartphone, however, this research opens an effective platform in the field of intelligent transportation system as well as telematics.

This research has been done in collaboration with TomTom B.V. in Amsterdam, with test driver participants from two countries, namely, Netherlands and India.

The outcome of this thesis is a machine learning model that can detect a harsh brake and sharp corner from a normal driving event using smartphone sensor data.

Keywords: *Signal Processing, Sensor Data, Machine Learning, Random Forests, Time Series*

Acknowledgement

First and foremost, I would extend my deepest gratitude to my supervisor, Thomas Bäck for his support and guidance over the duration of this thesis. I am also grateful to Hao Wang, for his precious insights and unending supervision over various spheres of this research.

I am thankful to my managers at TomTom, David Silsbury and Zeinab Bakhtiari, for their support in helping me carry out my research and giving me the opportunity to intern at TomTom under their supervision. I am grateful to my colleagues at TomTom, especially Driver Apps Data Science Team, for their critical comments and insights to my thesis that helped me expand my work over the course of my research.

I would like to extend a special thank you to the test drivers who helped me in collecting data and were a vital part of my research. They were namely Hashim, Zivile, Bilal, David, Vincent and Ronald.

I would like to extend gratitude to my brother, Hashim Mazhar, for his invaluable advice that always kept me going. I would also like to mention my sisters, my nieces and nephew for being my stress busters throughout my master’s degree.

I deeply thank my mother, Salma Akhtar, for being my biggest supporter, always lifting me up when I am down and the endless cups of coffee she made for me to keep me going.

I thank my friend, Wael Hussein, for always believing in me and always encouraging me to dream big.

Lastly, and most importantly, I dedicate this thesis to my late father, Mazhar Sayed Alam, who not only taught me never to give up in dire situations and the importance of knowledge but also dedicated his all in providing it for me. Without you, this would never be possible.

Contents

1. Introduction

Road safety is a shared responsibility. According to WHO's statistics 2018 [1], on a traffic study, almost 1.25 million people die in road crashes alone because of aggressive driving. Moreover, road accidents are ranked as the 9th leading cause of death worldwide, causing almost 2.2% of all death globally. [1]
We can make driving experience safer for anyone behind the wheel, using current advancements in technology.

A lot of previous research focusses on using sensors and hardware that is pre-deployed in vehicles. For instance, Onboard Diagnosis (OBD) adapters plugged into the vehicle's controller area network (CAN), or VBOX with speed sensors. However, a drawback of deploying these systems is they incur heavy costs; hence people are reluctant to install these systems. To overcome this, recent research is broadly based on exploiting the sensor technology equipped in our smartphones.

The use of smartphones while driving has become prevalent to apps like TomTom Speed Camera, Google Maps, Waze, etc. Our smartphones are equipped with rich sensor technology which is compact and efficient. The sensors in our smartphones are capable of measuring acceleration, rate of rotation, light, touch on screen, air pressure as well as our position in space.

These sensors can provide us with information about driving behaviour of a driver and their reckless driving maneuvers. We can leverage these sensors to provide safety assistance to the user while driving. Although sensor data retrieved from smartphones is noisy, it contains vital information with context to driving.

Using the power of sensors combined with machine learning algorithms, intelligent transportation systems can be transformed to make better safety predictions.

Previous research has been done in this area, assisting the driver, leveraging the sensor technology in mobile phones. The sensors used to provide this assistance are accelerometer, gyroscope, GPS and magnetometer. Research within this niche focuses on various aspects, detecting road information, driver behaviour, driving maneuvers or vehicle detection using smartphone sensors. Section 2 of this paper provides a literature review.

1.1 Research Goal

This thesis is conducted in collaboration with TomTom B.V. This research aims at detecting critical driving events that occur while driving solely using noisy (unformatted) smartphone sensor data. Critical events while driving can be detected in many ways, for instance by studying the lateral and longitudinal acceleration of the vehicle. [2] However, we are interested in two critical events that take place often while driving on the road namely harsh braking and sharp cornering. The focus of this thesis is to investigate the harsh brake and sharp corner critical events from representative driving data and classify it using machine learning algorithms.

At the end of this thesis, we should have a machine learning model that is able to detect harsh brake and sharp corner from normal driving event.

1.2 Outline

The outline of this thesis is as follows:

Firstly, Section 2 of this document gives background to the research topic, literature review as well as introduction and evaluation of mobile sensors. The experimental setup, methodology, algorithms and evaluation metrics used for this project are mentioned in section 3. The following section 4, will give us details on the data aggregation and features created for our dataset. Section 5 of this project is a statistical study of the raw as well as aggregated data where we further explore our dataset. In section 6 we run our experiments and discuss the results we obtain. Finally, in section 7 we conclude with key results, put forth our suggestions and recommendations for future research.

2. Background

2.1 Related Work

Fazeen, M., Gozick, B et al. [3] have described an approach of using 3-axis accelerometer data, to classify abnormal driving events, namely hard brakes and acceleration, and road anomalies. They achieved an accuracy of 85% for road anomalies. These events are mainly detected by calculating the time duration, difference, and slope between successive accelerometer readings on certain axes and comparing them to dynamic thresholds. However, the data smoothing, and noise removal procedure are not mentioned, as typically sensor data from smartphones is very noise prone. They also don't use any machine learning algorithma for their detection.

Castignani et al. used 2-axis accelerometer data to measure the g-force caused by longitudinal and lateral acceleration and used fuzzy logic to classify driving behaviour as normal, aggressive, and very aggressive. They claim that their classification is real-time and costs nothing. The average of the Euclidean norm of longitudinal acceleration and lateral acceleration along with speed is used to a fuzzy inference system. [3] However, the evaluation and accuracy of their approach are missing from their paper.

Another research by Johnson and Trivedi [4], where they proposed the MIROAD platform – A mobile Sensor Platform for Intelligent Recognition of Aggressive Driving. This platform used Dynamic Time Wrapping (DTW) [5] and smartphone sensors like accelerometer, gyroscope, magnetometer, GPS and video to detect, recognize and record these actions without external processing. [4] They collected their data on a sampling rate of 25Hz and used a combination of gyroscope signals on the x-axis, acceleration on y-axis and pitch signals, which are fed into the DTW algorithm to classify turns, braking, and excessive speed driving events. Their analysis shows they achieved an accuracy of 97% for classifying these aggressive driving events.

Eren et al. [31] propose a system like the MIROAD platform [4], however, they use Bayesian classifier to classify a drivers behaviour as safe or risky using smartphone sensors like accelerometer, gyroscope, GPS and magnetometer. Their study involved 10 drivers, of which 2 were manually labelled as "risky" according to their behaviour on the road. Their system correctly classified 14 drivers from a test set consisting of 15 drivers. The effectiveness of the technique cannot be judged as binary classification is easier than driving event classification. However, their results look promising.

Ma et al. [6], proposed a sensor noise distribution determination algorithm for a smartphone on a vehicle, namely DrivingSense. Driving Sense could detect three dangerous driving events, speeding irregular driving direction change and abnormal speed control. They make use of all 4 sensor signals, accelerometer, gyroscope, GPS as well as a microphone.

Junior et al. [7] investigate different machine learning algorithms to assess which sensor/ method assembly allows classification with higher performance. They have tested four machine learning algorithms, Artificial Neural Networks (ANNs), Support Vector Machines (SVM), Random Forests (RFs), Bayesian Networks (BNs). They concluded that gyroscope and accelerometers, along with bigger sliding windows were the most suitable to detect driving events. Also, their best working model was Random Forest, with an AUC score of 0.98 for aggressive breaking event and 0.99 for aggressive right change event. [7]

2.2 Mobile Sensors

Modern smartphones are equipped with micro-electromechanical systems (MEMS) such as accelerometers and gyroscopes to determine device orientation. Gyroscope, Accelerometer and Magnetometer built in smartphones form the Inertial Measurement Unit (IMU) available in smartphones. Apart from there, smartphones come with Proximity Sensor, Global Positioning System (GPS), Microphone, Touch Screen Sensors, Fingerprint Sensor, Barometer etc.

For this research, we exploit the accelerometer, gyroscope, GPS sensors from smartphones.

(a) Accelerometer:

The three-axis accelerometer (Figure 1) of the phone is used to detect vibration, acceleration and tilt to determine movement and orientation along the three-axis. Accelerometer also detects how fast your phone is moving in any linear direction. The values reported by accelerometers are measured in increments of the gravitational acceleration, with the value 1.0 second per second in the given direction. [8] Acceleration values may be positive or negative depending on the direction of the acceleration.

(b) Gyroscope*:*

Gyroscope measures the rate at which a device rotates around a spatial axis. It also provides orientation information as well as directions like up/down and left/right but with a greater precision, for example, how much the device is tilted. [8] Rotational values are measured in radians/second (rad/s) around the given axis. Rotation values may be positive or negative depending upon the direction of rotation. (Figure 2)

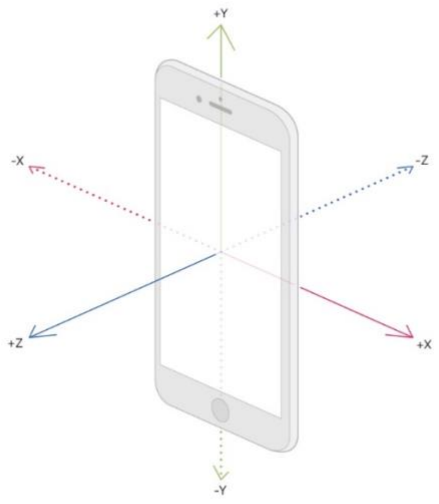


Figure 1: 3-axes of an accelerometer in a smartphone [9]

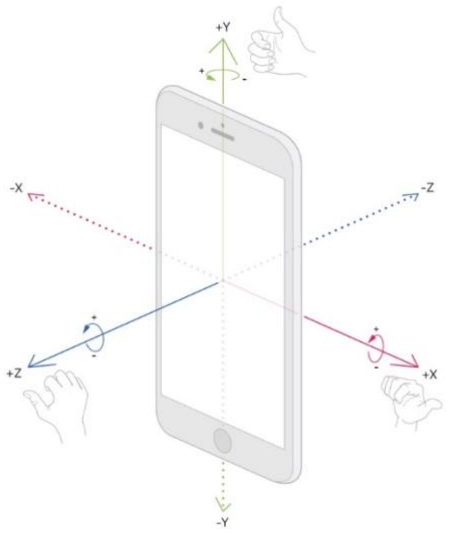


Figure 2: 3-axes of a gyroscope in a smartphone [9]

(c) GPS

Global Positioning System (GPS) unit in our smartphone is used to communicate with 31 satellites in orbit to determine our precise location on the Earth. [9] As GPS is a satellite-based system it gives your location irrespective of the weather conditions, 24 hours a day. Modern day GPS receivers have an accuracy of up to 30-40 centimeters (cm).

2.3 Sensor Evaluation on iOS vs Android phones

iPhone 6s uses only one InvenSense inertial sensors for accelerometer and gyroscope (6-axis accelerometer and gyroscope) [10]. The sampling rate for accelerometer and gyroscope can be set from 10 Hz up to 100Hz via the motion data manager. If there is a delay in receiving the data, interpolation may be necessary.

For Samsung Galaxy S8 inertial sensor slot belongs to STMicroelectronics with LSM6DL 6-axis inertial sensor. The sensitivity for Accelerometer is 0.061 mg/LSB and that of the gyroscope is 125 mdps/LSB. [11] [12]

3. Methodology

In this section, we describe our experimental setup as well as the methods and techniques used for this study in details.

3.1 Experimental Setup

For performing this research, ground truth data was required to be able to validate results as no data was available to take this research further. Thus, experiment setup was designed to collect data and manually label it. For collecting data for harsh brakes and sharp corner events 6 drivers were involved. The cars used for the experiment along with the percentage (%) of data collected with each were:

- Toyota Camry, automatic, 2019 model (15%)
- Volkswagen polo 1.2 united edition 2008, and (55%)
- Chevrolet Spark 1.0 (30%)

The mobiles phones used for collecting data along with the percentage (%) of data collected with each were:

- iPhone 6s (45%)
- Samsung Galaxy s8 (15%)
- ViVo Q10 (40%)

Two different applications were used on iOS and Android platforms to collect sensor data, namely $SensorPlay\,*$ and $Physics\ Tool\ Box**$ respectively. All phones were mounted on a CarKit holder during the data collection drive. The position of the CarKit holder was either on the dashboard or the windshield. Figure 3 shows the position of the phone on the dashboard for collecting the data.

* *https://apps.apple.com/us/app/sensor-play-data-recorder/id921385514*
** *https://play.google.com/store/apps/details?id=com.chrystianvieyra.physicstoolboxsuite&hl=en*

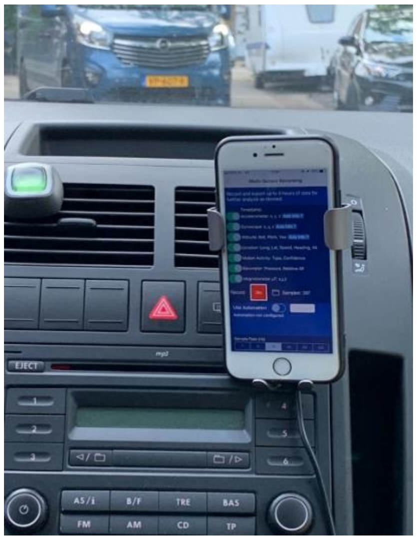


Figure 3: Position of Smartphone in the Car

The drives for data collection took place in two countries, India and the Netherlands. Data collected in India was much more prone to noise than in the Netherlands, the reason being that roads in India are more uneven and bumpy than those in the Netherlands. However, we preprocessed data from both countries together, to have a more realistic preprocessing of data.

Two individuals were in the car at a time of data collection, the passenger noted lap times whenever the driver performed a harsh brake or sharp corner event. When two individuals were not available, video recording of the drive was used to match the harsh brake or sharp corner event to the time it occurred according to the video recording of the drive, to build ground truth data.

Some parameters were kept fixed for the experiment whereas some diverse on purpose for having variety in data and mimic realistic driving situations.

The fixed parameters of the experiment are

- The orientation of the phone during the drive, in-car kit holder and portrait orientation.

The variable parameters of the experiment are

- Test cars used for driving
- Different phones used hence different sensor sensitivity for each
- Different types of roads (bumpy/rocky to even roads/race tracks)
- Different drivers with different driving styles
- Familiar vs unfamiliar roads
- The sampling rate for collecting data

3.1.1 Data Collection

Smartphone sensor data was collected as per the experiment described in section 3.1. For this project, data collected was for studying harsh brakes and sharp corners. These events were collected over a period of many drives (approximately 36 drives)

Accelerometer, gyroscope both on 3 axis (x,y,z) and GPS data were collected to build ground truth for harsh brake and sharp corners. The former two were collected at three samplings rates, 10Hz, 25 Hz, and 80 Hz, whereas the latter, GPS data, was collected at (+/-) 1-second accuracy, using the apps mentioned in 3.1 for Android and iOS platforms. For the sake of clarity, we will represent data collected for harsh brakes as $dataset\_harsh\_brakes$ and data collected for sharp corners as $dataset\_sharp\_corners$ for the rest of this project. Additionally, throughout this thesis "raw" signal would refer to signal before smoothing, and "aggregated" signal would refer to signal after smoothing.

3.2 Signal Preprocessing

The raw data obtained from mobile phones is very noisy, and a single data point is not very informative. Hence, to make sense of it but also not lose any important information, smart techniques need to be applied to aggregate the data without losing vital information.

3.2.1 Windowing with Overlapping

Windowing time series data is a common technique done in signal processing. It refers to splitting the input signal into grouped segments to make much better sense of a continuous signal. In this method, a time window for the signal is selected, and data points are grouped within that time window. This helps to reduce the noise from the signal without suppressing the outliers too much (since outliers are important for us to detect harsh brake and sharp corner events). The overlapping of windows helps to avoid causing discontinuities in the signal at the border. Windowing does thus change the signal, but the change is designed such that its effect on signal statistics is minimized. [13]

3.2.2 Fast Fourier Transformation

The Fast Fourier transform [14] is a faster version of the Discrete Fourier Transform. It commonly used to convert a signal in the time domain to the frequency domain. In the frequency domain, you look at the same signal in a different way, i.e. voltages present at various frequencies. [14] FFT has the mathematical property that allows us to represent any signal as a sum of a group of sine waves.
The term "strength of a signal" is used to represent energy of a signal calculated using FFT. In the discrete-time domain, energy of a signal is calculated as:

$$E_{(a)} = \sum_{t=0}^{\mathrm{n}} |x(t)|^2 \qquad (1)$$

We calculated the "strength of the signal", also known as energy, using *Equation 1* above, where $E(a)$ is the energy calculated on acceleration x, y, z axis separately, $x(t)$ are the values of acceleration on the respective axis, in the time window (t).

3.3 Machine Learning Algorithms

In this project, we use machine learning algorithms for classification. This section introduces the machine learning concepts and algorithms used in this project.

3.3.1 Decision Trees

The basic idea of decision tree classifiers is to divide a complex decision into a union of simple decisions using a tree structure. [15] They are very powerful compared to other machine learning classifiers as they can be visualized and interpreted. The way a decision tree works for classification tasks is that it generates a set of rules to classify the data. These rules can hence create a model that can classify even the complex of situations.

Each node in the decision tree represents an input variable from the dataset which is usually represented as ‘*X*’ in machine learning problems, whereas, each branch coming represents the partition of the dataset, and each leaf node represents a class label which is a decision taken after computing all attributes, which is denoted in machine learning problems as variable ‘*y*’.

A greedy approach is taken when dividing the input space recursively, where different splits are tested using a split criterion, for example, Gini index. The split with the minimum Gini index is selected, and all input variables and possible splits are evaluated and chosen in a greedy manner with respect to the Gini index. Figure 4 illustrates an example of a simple decision tree.

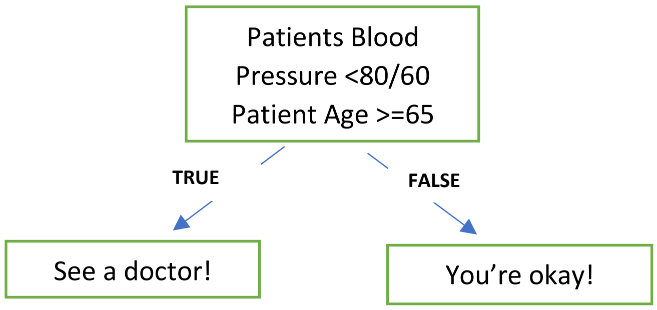


Figure 4: Simple illustration of a decision tree

3.3.2 Random Forests

Random Forests are ensemble learning methods built on decision trees. Kwok and Carter (1990), are the earliest to mention an ensemble of decision trees when they empirically noticed averaging multiple decision trees with different structure consistently produces better results. [16] Random Forest use a very powerful technique called Bootstrap Aggregation or Bagging. In this technique, multiple random samples with replacement are taken from the dataset, termed as bootstrap samples, and used as training data for each tree in the forest.

The predictions are done by combining the ensemble and taking the mean of models in case of regression, or majority vote in case of classification. Figure 5 depicts the concept of ensemble approach using majority voting and bootstrap samples.

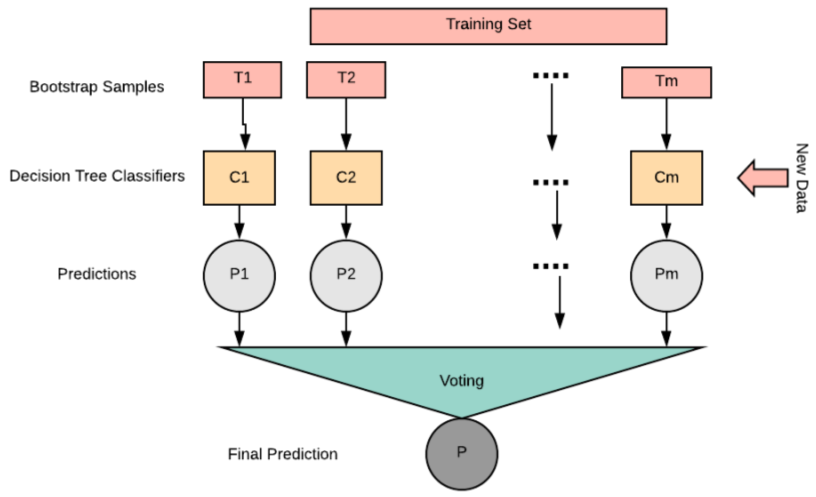


Figure 5: Random Forest ensemble learning with bootstrap samples (adapted from [17])

A disadvantage of bagging, however, is that it can result in decision trees that are highly correlated. Random forests overcome this disadvantage by randomly selecting features from each bootstrapped sample such that the predictions of the learned sub-trees have less correlation. The random forest algorithm is as follows:

1. Create random bootstrap samples from the training data set
2. Create a decision tree on bootstrap samples
    - Select random subsets of features
    - Split nodes on best features

3.4 Model Tuning

Once the model is trained, we need to tune the parameters of the model to give us the best results. This is achieved by hyperparameter optimization. Hyperparameters are important as they have a significant impact on the performance of the training model.
Two hyperparameter optimization techniques in scikit-learn* are available namely, Grid Search and Random Search.

Grid Search is a brute force technique where it trains the algorithm for all combination of parameters and measures the performance using cross-validation technique. The best performing combination is chosen as the best set of parameters. Although grid search is a simple algorithm, it can be computationally expensive with increasing number of parameters.

Random Search samples the search space randomly and evaluates the performance using cross-validation. It has an advantage of less computational time, as it does not search the search space exhaustively, but we are not always guaranteed best combination of hyperparameters.

*https://scikit-learn.org/stable/

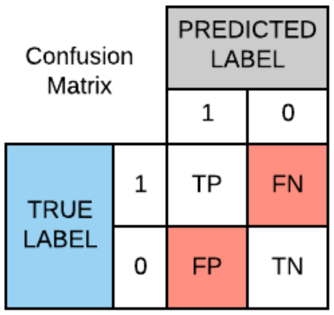
For this thesis, after discussion with the supervisors, we opted out to use hyperparameter optimization, as the model was performing well without any tuning of parameters.

3.5 Model Evaluation

We need some metrics to evaluate every machine learning model, to further understand how our model performs against unseen data, and to determine if our model is overfitting or underfitting data. Because we are dealing with an imbalanced dataset problem, we will use the metrics mentioned in this section for evaluating our model.

3.5.1 ROC-AUC Curve

Receiver Operating Characteristic and its Area Under Curve (ROC-AUC) is a state-of-the-art evaluation metric used to analyze the performance of your classifier. ROC was developed as a part of “Signal Detection Theory” during World War II, where they used it for the analysis of radar images. It was first found to be a useful evaluation measure in medical fields in the 1970s. [18]



Confusion Matrix		PREDICTED LABEL	
		1	0
TRUE LABEL	1	TP	FN
	0	FP	TN

Figure 6: Confusion Matrix

Abbreviation	Meaning
TP	# of labels where we predict True and they are actually True.
FP	# of labels where we predict True but they are actually False.
TN	# of labels where we predict False and they are actually False.
FN	# of labels where we predict False and they are actually True.

Table 1: Explanation of Abbreviations used in Confusion Matrix

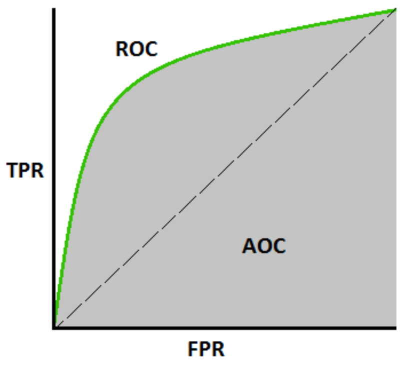


Figure 7: Example of a ROC Curve. Adapted from [32]

The ROC Curve is plotted using the False Positive Rate (FPR)on the x-axis against the True Positive Rate (TPR) on y-axis. These are calculated for different values of thresholds, as it allows you to regulate both rates and pick a threshold that best fits the problem at hand.

Consider the confusion matrix (Figure 6) which shows the True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN), resulting from comparing your classifiers predicted labels to true labels.
We can calculate the TPR and FPR from the confusion matrix in Equations (2) and (3) respectively.

$$TPR = \frac{TP}{TP+FN} \tag{2}$$

$$FPR = \frac{FP}{TN+FP} \tag{3}$$

The Area Under the Curve (AUC) of a ROC summarizes the skill of the model to differentiate between positive and negative class, on a scale between 0 to 1. ROC can be calculated only for classifiers that yield a probability of the classification. An AUC of 1, is that of an ideal classifier (Figure 7), which can perfectly differentiate between positive and negative class.

3.5.2 PR Curve

Using a ROC alone presents an overly optimistic view of the models' performance if there is a large imbalance in your dataset. [19] Hence, comes the Precision-Recall Curve (PR Curve), when there is a large class imbalance.
A PR Curve shows the relationship between precision (Positive Predicted Value) and recall (Sensitivity), where recall is on the x-axis against precision on the y-axis, for different thresholds.

The reason why the PR curve works well in case of imbalanced data is because of the avoidance of True Negatives (TNs) as the positive class is severely in minority. Because PR curve incorporates precision, which uses true positives, in contrast to true negatives, it is more sensitive to imbalance. [20] The area under the PR Curve is the Average Precision (AP), which just like the AUC, summarizes the skill of the model to differentiate between positive and negative class, between a scale of 0 to 1. An ideal classifier, towards the right-hand corner, is one which perfectly differentiates between positive and negative class and has an AP of 1.
We use Equation (4), (5) and (6), referring to our confusion matrix (Figure 7), to plot our PR curve and calculate our AP.

$$Precision = \frac{TP}{TP+FP} \tag{4}$$

$$Recall = \frac{TP}{TP+FN} \tag{5}$$

$$AP = \sum_n (R_n - R_{n-1})P_n,$$

where P_n and R_n are the Precision and Recall at the n^{th} threshold. (6)

3.5.3 Learning Curve

Learning curves are a good way to diagnose the performance of machine learning models. It gives you an insight to whether you model has a good bias and variance tradeoff, which helps us understand whether our model is overfitting or underfitting the data, as well as if our training and cross-validation sets are representative. Learning curves can also be used to optimize the parameters of your model.

To evaluate learning curves, dual curves are created, i.e., the train learning curve and the validation learning curve. The train learning curve gives us an idea of how well our model is learning from the given training dataset, whereas, the validation learning curve, gives us an idea of how well our model can generalize using the cross-validation method. [21]

3.5.4 Matthews Correlation Coefficient (MCC)

Matthews correlation coefficient (MCC) is extensively used in bioinformatics as an evaluation metric. It is a measure of quality for binary classification problems. [22]

MCC is a good metric to evaluate imbalanced datasets as it considers the proportion of the classes, so even if the dataset is highly imbalanced, it is considered as a balanced measure.
The MCC score lies between -1 to +1, where a coefficient of -1 indicates a definitive disagreement between predicted and true labels, 0 means a random prediction and +1 indicates a perfect agreement between predicted and true labels.
Referring to our confusion matrix (Figure 7), the MCC is calculated as below (Equation 7)

$$MCC = \frac{TP*TN-FP*FN}{\sqrt{(TP+FP)*(TP+FN)*(TN+FP)*(TN+FN)}} \tag{7}$$

4. Data Aggregation and Feature Engineering

Creating features is an essential step for every machine learning problem, as feature generation makes our dataset more representative of the problem at hand. In this section, we discuss in detail how we smoothen of our noisy sensor data from smartphones as well as our manual feature generation techniques.

4.1 Signal Smoothing

Smoothing the noisy signal is vital when it comes to classifying harsh brakes and sharp corners. This is because dealing with raw sensor data, collected at different sampling rates, is not very interpretable. Sliding window with overlapping (section 3.2.1) is a common technique used in signal processing.

For our use case, accelerometer and gyroscope data are aggregated on a window, where the size of each window is twice times the sampling rate (equivalent to two-second windows), with a 25% overlap. This is illustrated in Figure 8. The overlap window is so that if any abnormal event (harsh brakes and sharp corners) falls under two windows, is still considered as one event instead of two. This will help us reduce labelling and creating artificial abnormal events, as well as helps us keep the continuity of the signal. Since our intervals are not independent of each other, overlapping is a way to ensure that we don't lose any important information at the boundaries. From these created windows, four features are extracted, namely, mean, minimum, maximum and simple average. (Refer section 4.2.3)

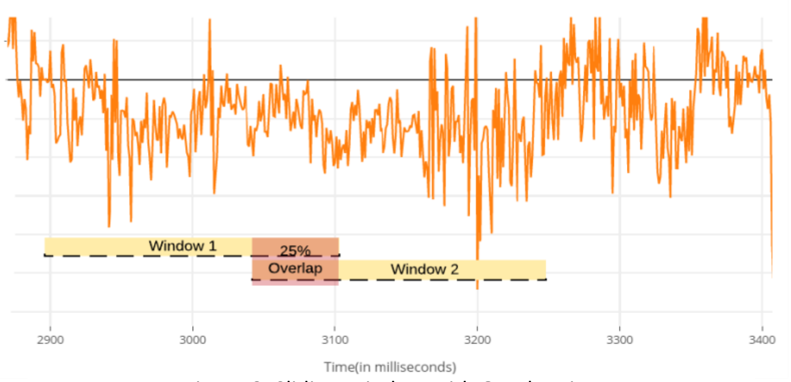


Figure 8: Sliding Window with Overlapping

Given a time series, $Y = y_1, y_2, y_3, \ldots\ldots, y_L$, where L is the length of the given time series

Let $k = 2 * sampling\ rate$. The window (w) of the time series is then obtained as:

$$w_0 = y_1, y_2, y_3, \ldots\ldots, y_k \tag{8}$$

.

.

.

$$w_i = y_{i-k/4}, y_{i+1-k/4}, \ldots\ldots, y_{i+3k/4} \tag{9}$$

where $i \in [1, L]$. We assume here that k is a multiple of 4.

4.2 Feature Extraction

Feature extraction is a vital step for machine learning algorithms, as they learn about the dataset from the features generated. The features should be generated as such they are representative of the problem we are solving.

4.2.1 Jerk

Jerk is the derivative of acceleration, which is the rate of change of acceleration with time. The intuition behind creating this feature is that when an abnormal event takes place at a steady motion, the car will slow down smoothly in contrast to if an abnormal event takes place in a short span of time, the car will stop in a jerky motion.

The formula used to calculate jerk is illustrated in Equation (10):

$$Jerk(t) = \frac{d\overrightarrow{(a)}}{dt} \tag{10}$$

Where $\overrightarrow{(a)}$ denotes acceleration at time (t).

4.2.2 Magnitude of Acceleration

The magnitude of acceleration was also calculated, which is the vector sum of the acceleration vector on all three axes. The magnitude is calculated by squaring acceleration on each axis, adding them up and taking the square root of the sum, as shown in equation 4 below.

$$|\vec{a}| = \sqrt{(a_x)^2 + \left(a_y\right)^2 + (a_z)^2} \tag{11}$$

Where a_x, a_y, a_z is acceleration on *x, y, z*-axis respectively.

4.2.3 Time Domain Features

We split our original data into windows (section 4.1)and calculate features in 4.2.1 and 4.2.2 on each window, and extract time-domain features from these windows. The time-domain features we extracted for our signal are mean, minimum, maximum and standard deviation for each window.
We used the $numpy$ library in Python to calculate the above four mentioned features. These four features were calculated for accelerometer, gyroscope and jerk on all three axes(x,y,z), as well as magnitude for each window.

4.2.4 Frequency Domain Features

Frequency domain feature was calculated using the Fast Fourier Transform (FFT) explained in section 3.2.2. The energy of the signal, also referred to as strength of the signal is calculated for acceleration on all three axes (a_x, a_y, a_z) with Equation 1 mentioned earlier.

In summary, our final set of manually created features are as follows:

Acceleration (X,Y,Z)	*	(minimum, maximum, mean, standard deviation)	=	3*4 = **12**
Gyroscope (X,Y,Z)	*	(minimum, maximum, mean, standard deviation)	=	3*4 = **12**
Jerk (X,Y,Z)	*	(minimum, maximum, mean, standard deviation)	=	3*4 = **12**
Speed	*	(minimum, maximum, mean, standard deviation)	=	1 *4 = **4**
Magnitude	*	(minimum, maximum, mean, standard deviation)	=	1*4 = **4**
Energy (X,Y,Z)			=	3*1 = **3**

Total number of manually generated features is **47.**

The dataset for the above aggregated features will be termed as $aggregated\_harshbrakes$ and -$aggregated\_sharpcorners$ respectively for harsh brake events and sharp corner events.

4.2.5 Automatic Feature Extraction with TSFRESH

Extracting features for time series data that are descriptive of our problem at hand is quite often very challenging. Hence, come libraries dedicated to time-series data that generate features from a wide pool of mathematical concepts related to time series. TSFRESH [23] is a python package which is dedicated to time series data calculating large number of time series features automatically. TSFRESH allows you to select from 64 number of implemented features, which makes it more universal. It is compatible with $Scikit\ Learn$ and multivariate time series data.

abs_energy, $aggregate\_correlation$, $friedrich\_coefficeints$ etc. are examples of few of 64 features it calculates. More details can be read here. [23]
For our study, we used the $ComprehensiveFCParametes()$ from $TSFresh$, which calculates all the features (794 features per time series) defined by default by TSFresh for time series data.

4.3 Feature Selection

Selecting a subset of features that are descriptive of the labels you are trying to classify is a crucial step for any machine learning problem. Feature selection algorithms are a family of greedy search algorithms that reduce your n-dimensional feature space to m-dimensional feature space, where $m < n$. We explain two feature selection algorithms used in our use case.

4.3.1 Boruta

Boruta is an algorithm that is a wrapper build around Random Forest. It works well for small datasets which are higher dimensions. Boruta is also a God of forest in Slavic mythology, hence the name. [24] In other words, Boruta shares the foundation of random forest classifier, namely adding randomness to our system and inferring results from the ensemble of randomized samples, can reduce the impact random fluctuation and correlations that can lead to misleading results. Hence, randomness in Boruta will be vital in selecting those attributes which are important. [24]

The algorithm works as follows:

1. The first step is to create shadow features which are nothing but shuffled copies of original features as this add randomness to the given data.
2. Then a random forest classifier is trained on the extended dataset and calculates the feature importance measure to evaluate the importance of each feature. The higher the value, the more important the feature.
3. It checks at every iteration whether the real features have a higher Z-score than its shadow features and constantly removes features which seem highly unimportant. This step is continued until maximum iterations is reached or all features are confirmed or rejected.

Boruta is different from other feature selection algorithms in the sense that it finds all features either strongly or weakly relevant to the decision variable. [24]

4.3.2 Recursive Feature Elimination Cross-Validation (RFECV)

This is a function available in $Scikit - Learn$ package in class $feature\_selection$. The goal of Recursive Feature Elimination (RFE) is to select a subset of features recursively from the given pool of features. The classifier is first trained on the initial set of features, and the importance of each feature is obtained. Then recursively, the least important features are removed from the current set of features, until the desired number of features is selected. [25]

An intersection of features generated by both these algorithms is chosen as our final set of features for training our classifiers.

5. Dataset

This section shows a statistical study on $dataset\_harsh\_brakes$ followed by $dataset\_sharp\_corners$ and explores the raw as well as smoothened signals and infer some insights from it.

5.1 Exploratory Data Analysis

(a) Harsh Brakes

Before proceeding with applying machine learning algorithms to our dataset, we do a statistical study on our dataset.
Figure 9 shows what a harsh brake looks like from a raw acceleration signal (raw i.e., signal before smoothing) on the z-axis collected at a sampling rate of 10Hz. (Refer to axes from Figure 1).

The blocks highlighted in black both Figure 9 and Figure 10**,** indicate how harsh brakes exert a higher g-force than a normal brake. The g-force reaches up to a value of approximately -1g (i.e., -9.8 m/s^2) for harsh brake events.
Figure 11 illustrates the minimum value of aggregated acceleration on the z-axis, after signal smoothing.

However, other features also indicate that a harsh brake event has taken place while driving. Figure 12(a), Figure 12(b) illustrates how the calculated energy of a signal differs when looking into a harsh brake event versus a normal barking event. For a harsh brake, the energy is much higher in comparison to normal braking event. These statistics show how a harsh brake differs from normal brake.

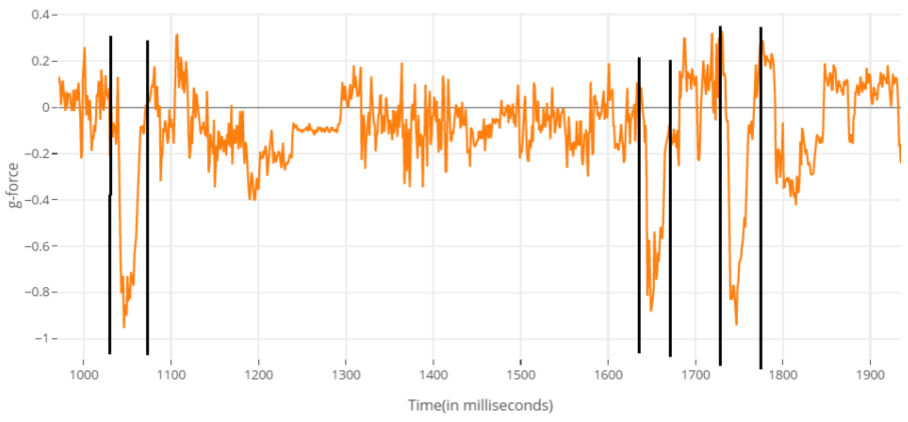


Figure 9: Harsh brakes on the raw acceleration z-axis

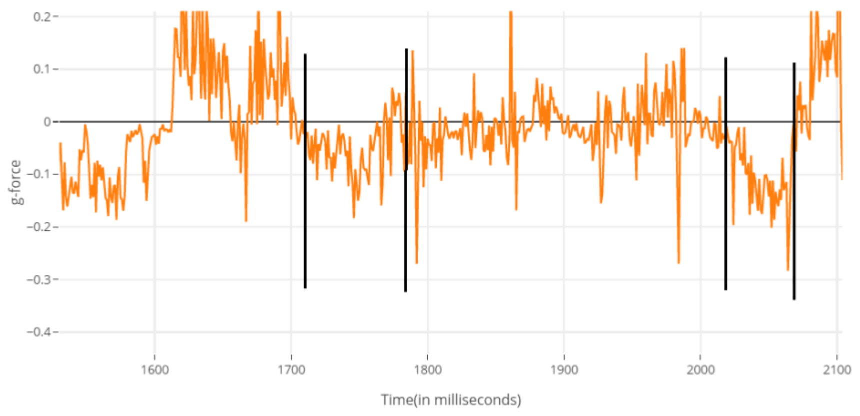


Figure 10: Normal brakes on the raw acceleration z-axis

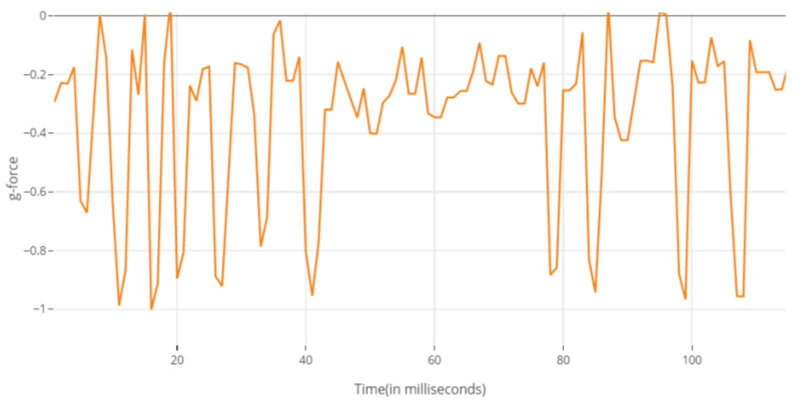


Figure 11: Harsh brakes on the aggregated minimum value of acceleration z-axis

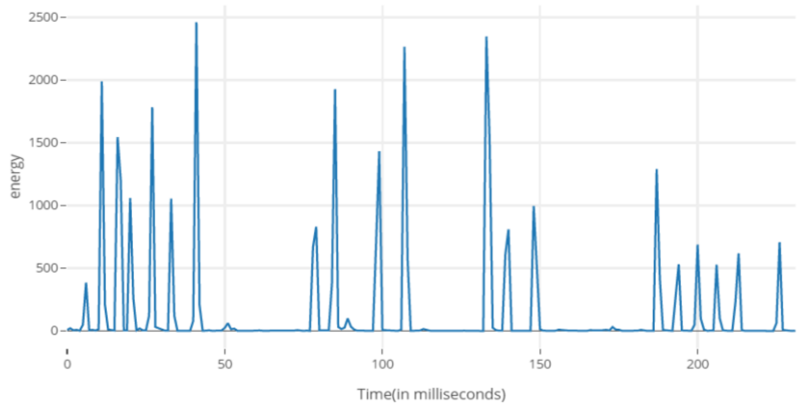


Figure 12(a): Energy on acceleration z axis during a harsh brake

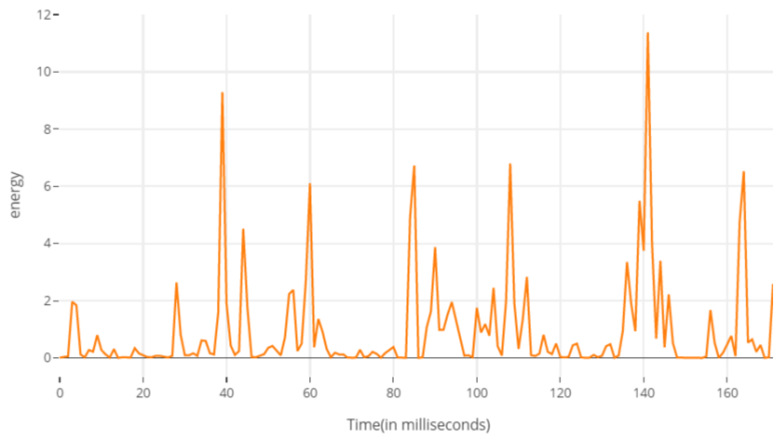


Figure 12(b): Energy on acceleration z axis during a normal brake

(b) Sharp Corners

For looking at sharp corner events and comparing them to normal turning events, we looked closely at the acceleration on the x-axis. As illustrated in Figure 13**,** which shows the raw acceleration on the x-axis at a sampling rate of 25 Hz, the normal turn has a lesser g-force and take more time to complete. Whereas, in Figure 14, exerts a higher g-force but also take lesser time to complete indicated by the width of the curve. A left turn is associated with a negative g-force, whereas a right turn is associated with positive g-force.

To further understand how a sharp corner looks like we look at the gyroscope data on the y-axis. Fig 15 (a) and (b) show the differences in the gyroscope data on the y-axis of normal turns versus the sharp corners. For normal turns, the width of the signal is wider with a lesser g-force exerted, whereas that of a sharp corner has a narrower width with a higher g-force exerted. A left turn is associated with a positive angular velocity whereas the right turn is associated with negative angular velocity.

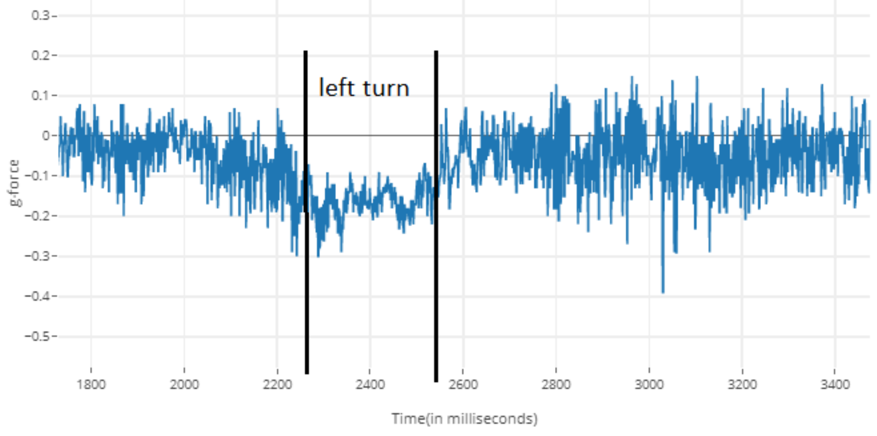


Figure 13: Normal turns on the raw acceleration x-axis

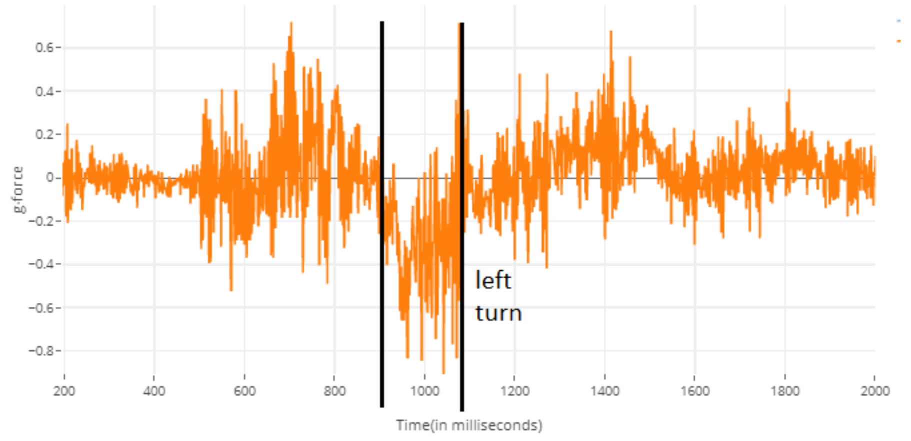


Figure 14: Sharp turn on the raw acceleration x-axis

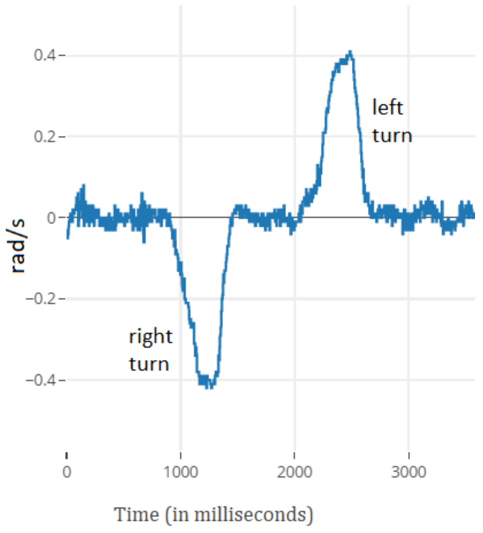


Figure 15 (a): Normal turns gyroscope y-axis

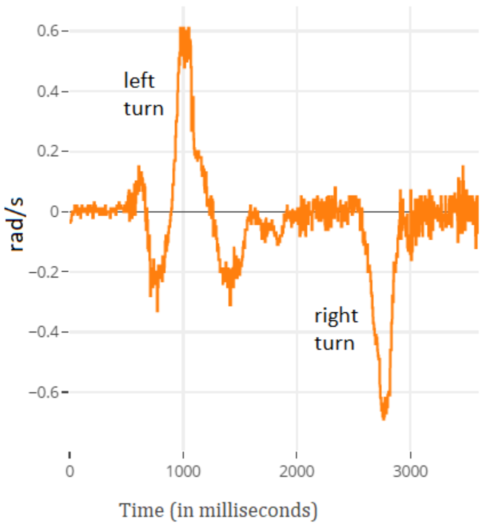


Figure 15 (b): Sharp turns gyroscope y-axis

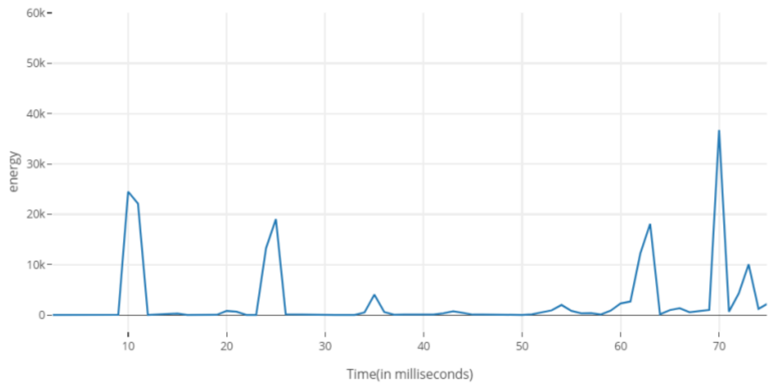


Figure 16: Energy on acceleration x-axis during a normal turn

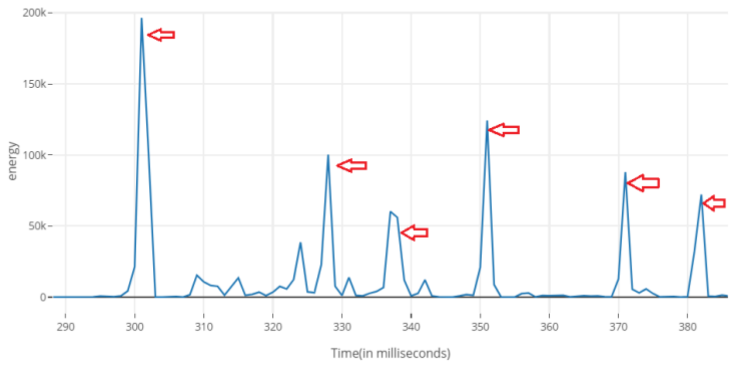


Figure 17: Energy on acceleration x-axis during a sharp turn

The energy calculated for acceleration on x-axis for both normal turns as well as sharp corner, using methods explained in section 3.2.2, we observe in Figure 16 and 17, the energy exerted by a sharp corner event is much more than a normal turn as expected. The red arrows in Figure 17 indicate the energies of sharp corner events.

This proves the normal turns have different statistical properties than sharp corner and hence they can be classified. The next section further reinstates this conclusion.

5.2 Density Plot of Features

(a) Harsh Brakes

We had around 12% of harsh brakes in our dataset, which was approximately 220 harsh brakes to learn from. This accounts for a highly imbalanced dataset.

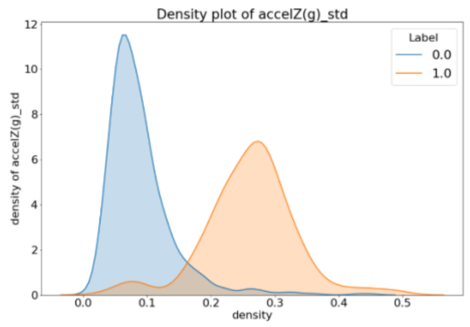
We further studied density plots of our aggregated data to have a clearer idea of the distribution of values of harsh brakes from normal driving events. Figure 18(a)(b)(c)(d) is density plots of normal driving event which is denoted by $label$ '0' and harsh brake event which is denoted by $label$ '1'.

We have plotted minimum (min), standard deviation (std) values from each window on acceleration z-axis, as well as the mean magnitude of each window and calculated energy on acceleration z-axis.

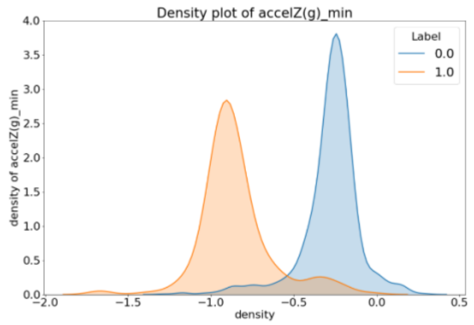
Figure 18 (a) and (c) depict a right-skewed distribution with very little overlap between values of harsh brakes and normal driving samples. Most values for minimum on acceleration z-axis, lie on the negative scale, which is expected. Harsh brake events take a value of minimum $-2.0g$, whereas this isn't observed with normal driving. The mean of magnitude of acceleration has higher values for harsh brake samples than for normal driving samples, as rate of change of velocity is expected to be faster when a harsh brake event is performed.

Also, the distribution of energy on acceleration z-axis in Figure 18 (d) indicates a very high variance for harsh brake samples than normal driving. Also, harsh brakes have much higher energy than normal driving which is expected.

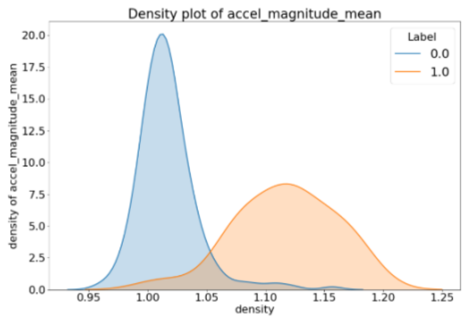
All four graphs show more variance for harsh brake samples than for normal driving samples.



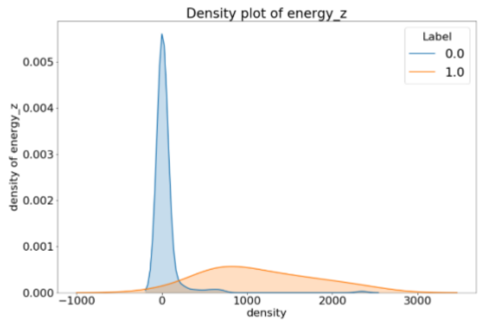
(a) Density plot of std on acceleration z-axis



(b) Density plot of min on acceleration z-axis



(c) Density plot of mean for magnitude of acceleration



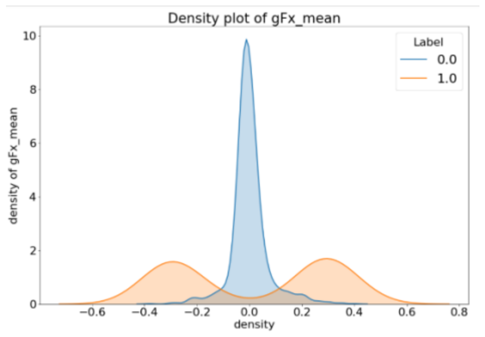
(d) Density plot of energy exerted on acceleration z-axis

Figure 18: Density plot of different features for harsh brakes (Label 1) versus normal driving (Label 0)

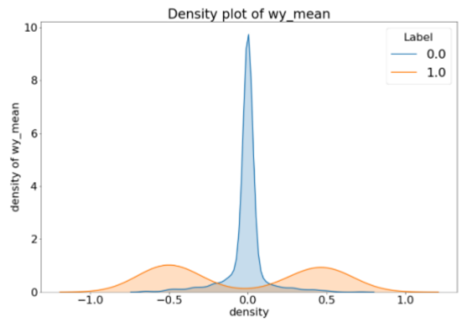
(b) Sharp Corners

Like harsh brakes, we had 10% of our dataset as $sharp\_corners$. This again indicates the presence of a highly imbalanced dataset. The 10% of $sharp\_corners$ account for around approximately 180 $sharp\_corner$ events.

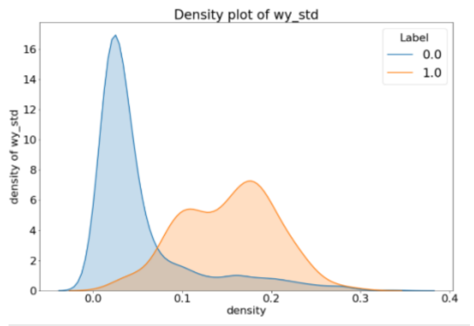
The density plots of sharp corners were studied to further understand the distribution of data for both sharp corners and normal driving. Figure 19 (a)(b)(c)(d) is density plots of normal driving events denoted by $label$ '0' and sharp corners denoted by $label$ '1'. The acceleration on x-axis (Figure 19(a)) has a g-force of $\succ 0.6g$ and $+0.6g$ for left and right turns respectively, whereas for normal driving the values lie between $-0.2g\ and\ +0.2g$. Figure 19 (b) illustrates the mean of the gyroscope on the y-axis, with values or sharp corners higher than the normal driving which is expected. Figure 19(c), standard deviation of gyroscope on y-axis, depicts a right-skewed distribution for normal driving, whereas the sharp corners show more variance in the standard deviation. Figure 19 (d) is a clear indication of the sharp corners being denser toward higher energy values whereas normal driving not so much. Also, here as well the sharp corners have more variance than normal driving.



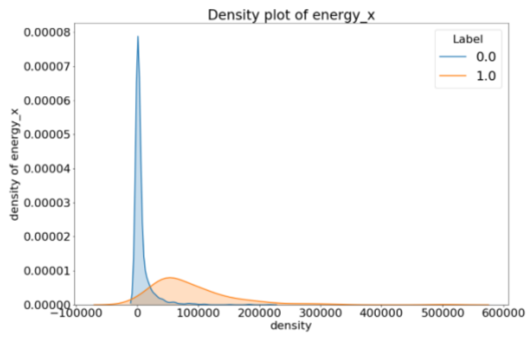
(a) Density plot of mean on acceleration x-axis



(b) Density plot of mean on gyroscope y-axis



(c) Density plot of std on gyroscope y-axis



(d) Density plot of energy exerted on acceleration x-axis

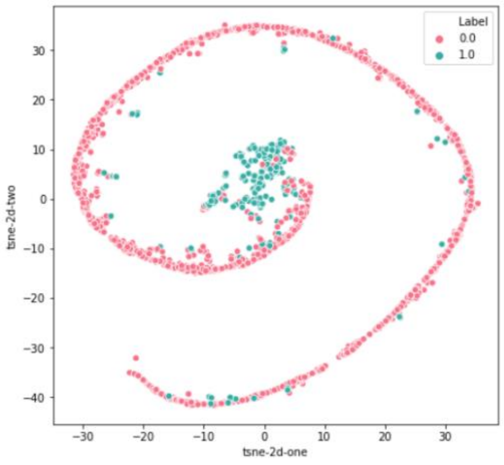
Figure 19: Density plot of different features for sharp corners (Label 1) versus normal driving (Label 0)

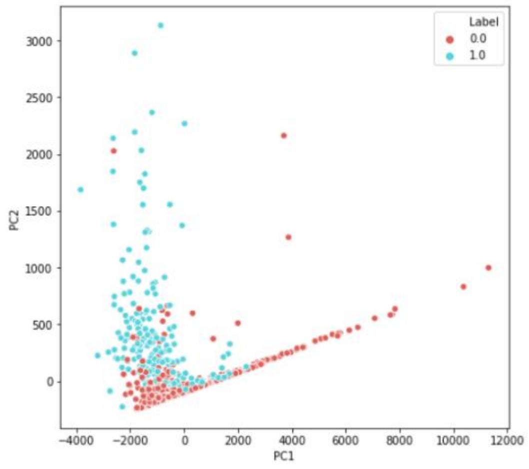
5.3 Low Dimensional Plot of Highly Dimensional Data

(a) Harsh Brakes

Below (Figure 20 (a) (b)) is a representation of our higher dimensional harsh brakes dataset in a low dimensional space. This is just a simple visualization to see how difficult it is to classify our harsh brakes from normal driving events. We used TSNE [26][27] from $scikit - learn$ to visualize it (Figure 20(a)). TSNE (T-distributed Stochastic Neighbor Embedding) captures the non-linear relationship between your data points, and clusters similar points together in low dimensional space. We have also plotted the 2 principal components obtained via PCA [28] to help us understand how harsh brakes and normal driving differ in low dimensional space. (Figure 22 (b)).

Figure 22, label 1 signifies a harsh brake and label 0 signifies a normal driving event. It is evident from the figure that harsh brakes form a cluster, thus making it further clear that our problem is highly classifiable.





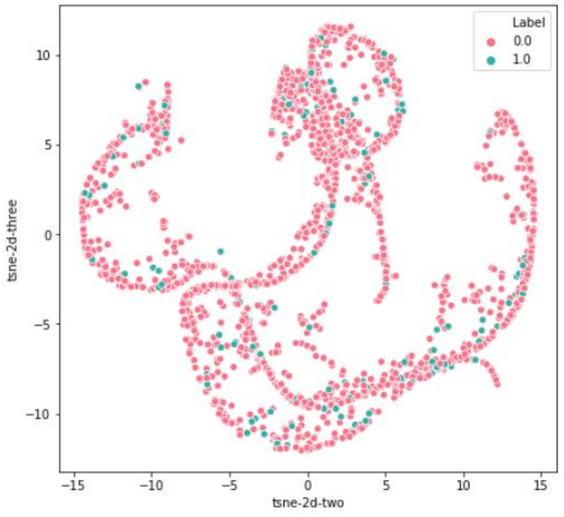
(a): 2-D plot of highly dimensional harsh brakes (Label 1) and normal driving (Label 0)

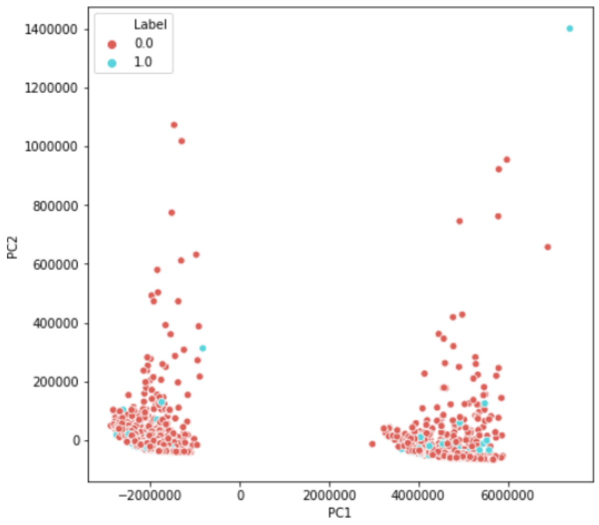
(b) Principal Components obtained with PCA for harsh brakes (Label 1) and normal driving (Label 0)

Figure 20: 2-D plot of high dimensional driving data (harsh brakes vs normal driving)

(b) Sharp Corners

Using TSNE, the low dimensional visualization of sharp corners can be seen in Figure 21(a)**.** The plot suggests that sharp corners don't form a certain cluster, hence they are harder to classify than the former, harsh brakes. Similarly, we have also added the plot for principal components obtained with PCA (Figure 21(b)) to understand how sharp corners differ from normal driving in low dimensional plane.
Here label 0 denotes normal driving and label 1 denotes sharp corner event.





(a): 2-D plot of highly dimensional sharp corners (Label 1) and normal driving (Label 0)

(b) Principal Components obtained with PCA for sharp corners (Label 1) and normal driving (Label 0)

Figure 21: 2-D plot of highly dimensional driving data (sharp corners vs normal driving))

6. Experiments

We experiment with different types of classifiers mentioned in section 3.3. For both our classifiers we have designed two experiments, Experiment #1 where we use under sampling techniques along with PCA and Experiment #2 where we use automatic feature generation techniques. (section 4.2.5)

We split our data into training and test set in the ratio 80:20 and use stratified 10-fold cross-validation on the training set to ensure our model is not overfitting. We use stratified fold to make sure that one class is not over-represented in a fold, as this can happen with imbalanced dataset.

To evaluate the performance of our model, we have used ROC AUC, AP as well as MCC. (section 3.5)

All experiments were done on an HP laptop with minimal specs, as the size of the data was not a lot. Later, we justify our choice of methods and techniques, as well as our model evaluation that works best with our dataset.

6.1 Experiment #1 – Under Sampling Dataset and PCA

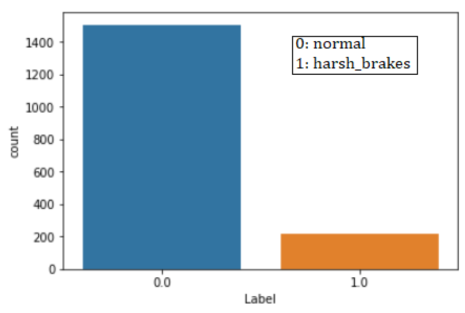
Since our dataset is very imbalanced and has high dimensionality, we decided to use under-sampling of the majority class from the training set as well as use Principle Component Analysis (Principal Component Analysis) to reduce the dimensionality of our data.

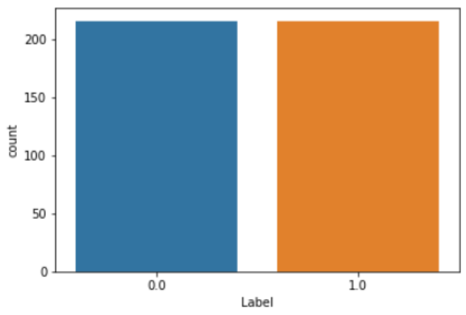
Under-sampling is basically taking the majority class and dropping data from the majority class until dataset is balanced. One disadvantage of under-sampling is that we lose important information from the data points that are dropped. We used the $scikit - learn\ RandomUnderSampler$ [28] function from the package $imblearn.random\_undersampling$ setting a random state to reproduce results.

After under-sampling, we still have high dimensional data, i.e., number of features of our dataset is very large in comparison to the number of observations in our dataset. In such a scenario, machine learning algorithms struggle to learn and become effective models. Thus, we opt for PCA. Simply put, PCA combines our highly dimensional input space in such a way that the most important parts of our data are retained, which each new variable being independent of one another. The number of components chosen for our dataset that captured the maximum variance is 3.

After under-sampling our dataset looks like depicted in Figure 22 ((a), (b)):

We use the under-sampled dataset with both classifiers mentioned in section 3.3. The results will be discussed in section 7**.**





(a) Before under sampling dataset (b) after under sampling dataset

Figure 22: Illustration of under-sampling dataset(before versus after)

6.2 Experiment #2 Using TS Fresh and Boruta with RFECV

One major disadvantage of under-sampling our dataset is that we often lose a lot of important information. Also, since we have a highly imbalanced dataset, under-sampling the majority class reduces the size of the data drastically, leaving very little room for training our classifier. To avoid this, we used our imbalanced dataset and instead creates features for our dataset that are more representative of the problem we are trying to classify. For this purpose, we used an automatic time series feature generation package called $TSFresh$ (section 4.2.5). Keep a note that prior to using $TSFresh$, our aggregated dataset consists of 47 features generated by manual feature engineering. (section 4.3.2)

Passing raw data is not very indicative of the problem we are trying to solve, in addition to different sampling rates we have used to add diversity in our dataset, we pass our aggregated datasets, namely, $aggregated\_harshbrakes$ and $aggregated\_sharpcorners$ (both separately) in the form of (X, y) to $TSFresh$. X represents the features of our data and y represents the labels of our data. The total aggregated features passed to $TSFresh$ for both our events are given in Table 2**.**

We use the $extract\_features()$ function in $TSFresh$, with input to the function our time series (Table 2), and $ComprehensiveFCParameters()$, $column\_kind$ (where you input the different sensor values your data contains), and $impute\_function$ (which handles missing values) The EfficientFCParameters() calculates all features per time series which are not computationally expensive. This reduces the computational costs for calculating features for our data. The total number of features calculated were 22,064. (788 features per time series)

Features given to $TSFresh$			
acceleration_x_std	acceleration_x_mean	acceleration_x_min	acceleration_x_max
acceleration_y_std	acceleration_y_mean	acceleration_y_min	acceleration_y_max
acceleration_z_std	acceleration_z_mean	acceleration_z_min	acceleration_z_max
gyroscope_x_std	gyroscope_x_mean	gyroscope_x_min	gyroscope_x_max
gyroscope_y_std	gyroscope_y_mean	gyroscope_y_min	gyroscope_y_max
gyroscope_z_std	gyroscope_z_mean	gyroscope_z_min	gyroscope_z_max
speed_std	speed_mean	speed_min	speed_max

Table 2: Input features for TSFresh

However, all these features are not relevant in our case. Hence, the next step is to use appropriate feature selection methods to obtain the most important features that are representative of our problem. For classifying harsh brakes and sharp corners, we use $Boruta$ and $RFECV$ from $scikit - learn$ and use the common features (intersection of features) generated by both as our final set of features for training. [30] (Section 4.3.1 and 4.3.2, respectively)

Parameters we used for Boruta are $n\_estimators = 10$, $max\_iter = 200$, $alpha = 0.5$, $random\_state = 88$, where $n\_estimators$ describes the number of trees in the forest, max_iter the maximum iterations to perform and $alpha$ is the level at which p-values are rejected. The execution time of Boruta was 59 seconds for $dataset\_sharp\_corners$ and 45.3 seconds for $dataset\_harsh\_brakes$.

For RFECV we used $cv = 10,\ step = 1,\ scoring = 'f1'$ as parameters again with $random\_state\ = 88$ for reproducible results, where cv specifies the number of folds for cross-validation, $scoring$ describes the score to maximize at each fold. We have chosen scoring as ‘$f1$’. The execution time for RFECV was 45 minutes 34 seconds for $dataset\_sharp\_corners$ and 34 minutes 12 seconds for $dataset\_harsh\_brakes$.

The final features selected for harsh brake events and sharp corner events with respect to their importance is given below. Figure 23 (a) and (b) illustrates the selected features sorted in descending order of the importance.

(a) Harsh brakes

For harsh brakes, the common features selected by $Boruta$ and $RFECV$ are 8, namely:

1- $accelZ(g)\_std$: Standard deviated of windowed signal on acceleration Z-axis.
2- $accelZ(g)\_min$: Minimum value of windowed signal on acceleration Z-axis.
3- $accelY(g)\_mean$: Mean of windowed signal on acceleration on Y-axis.
4- $energy\_z$: Energy of windowed signal calculated using FFT on acceleration Z-axis.
5- $gyroZ(rad/s)\_std$: Standard deviation of windowed signal on gyroscope Z-axis.
6- $gyroY(rad/s)\_std$: Standard deviation of windowed signal on gyroscope Y-axis.
7- $jerkZ\_min$: Minimum value of the windowed signal for calculated jerk on Z-axis.
8- $Speed(m/s)\_max$: Maximum speed of the windowed signal.

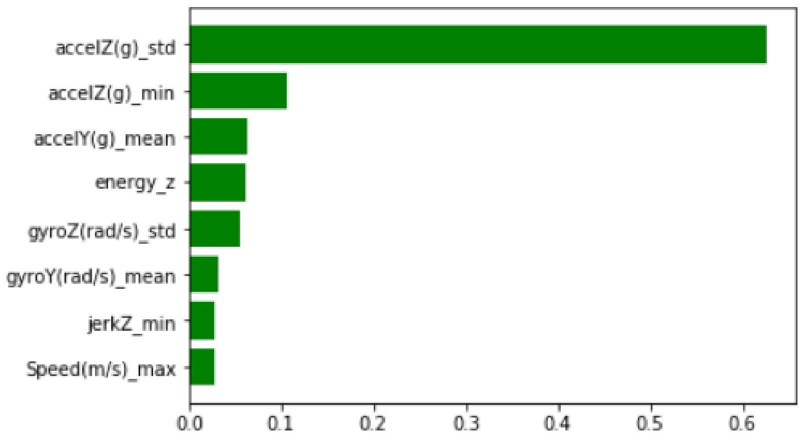


Figure 23 (a): Feature importance of selected features for harsh brakes

As we notice, the features selected by $Boruta$ and $RFECV$ were those created manually for classifying harsh brakes, none of the automatically generated features were selected.

(b) Sharp Corners

For sharp corners, the common features selected by $Boruta$ and $RFECV$ are 5, namely:

1- 5_abs_energy: Calculates the absolute energy of the time series which is the sum over the squared values. This is calculated from acceleration on x-axis by TSFresh. [23]
2- $21_fft_coefficient_coeff_0_attr$_abs: Calculates the fourier coefficients of the one-dimensional discrete Fourier Transform for real input by FFT algorithm. [14] This is calculated from gyroscope on y-axis by TSFresh. [23]
3- $accelX(g)\_min$: Minimum values of the windowed signal on acceleration X-axis.
4- $accelX(g)\_std$: Standard deviation values of the windowed signal on acceleration X-axis.
5- $gyroY(rad/s)\_mean$: Mean of the windowed signal on gyroscope Y-axis.

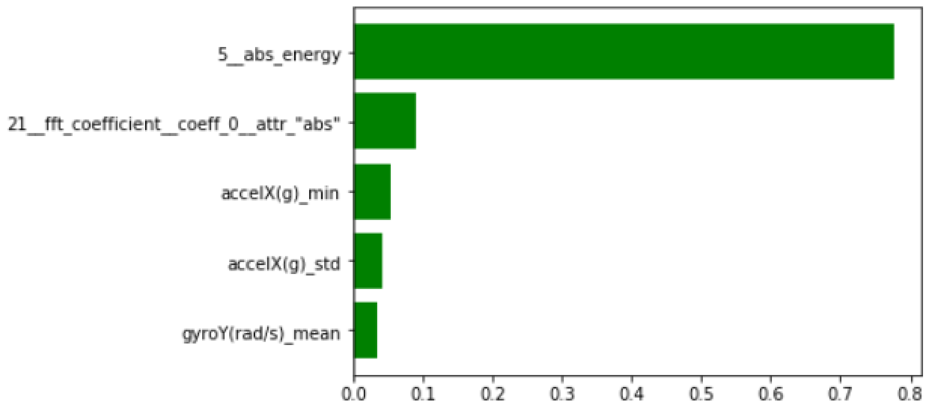


Figure 23 (b): Feature importance of selected features for sharp corners

Figure 23(b) suggests the features chosen by our feature selection algorithms are a mix of both, manually extracted features as well as automatically created features. A sharp corner event is a bit more of a challenging problem to learn than a harsh brake event. However, the feature that is most important to describe the event is simple to calculate, and does not require a lot of computational time (Equation 12)[23]

$$Energy = \sum_{i=1}^{n} x_i^2 \qquad (12)$$

7. Results and Discussion

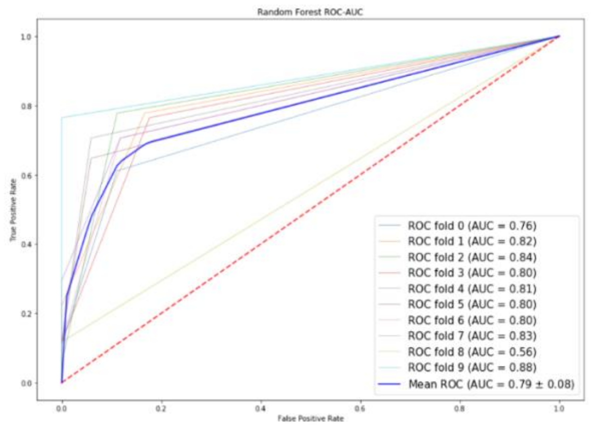
In this section, we will first explore the results obtained with the classifiers in section 3.3 with our chosen evaluation metrics mentioned in section 3.5**,** according to our experimental setup described in section 6.

7.1 Model Performance

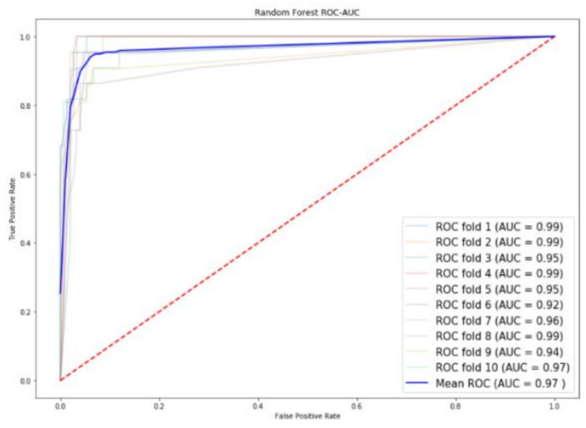
(a) Harsh Brakes

For our chosen classifiers, Decision Trees and Random Forests, the best performing classifier was Random Forests based on all three-evaluation metrics, ROC AUC, AP and MCC.

We also observed that the classifier performs much better with our Experiment 2 using automatic features with manually generated features than Experiment 1 using manually generated features with under-sampling majority class. Figure 24 shows the ROC AUC using Random Forests for both experiments over stratified 10-fold cross-validation. We can see the mean AUC under ROC for Experiment 1 is 0.79 whereas for Experiment 2 is 0.97. Although no automatically generated features were selected in Experiment 2 for our harsh brakes, the two feature selection methods, help increase the AUC by selecting only those features representative of the problem.
The learning curves (Figure 25) converge for both experiments; however, it converges faster for experiment 2, with a good bias and variance tradeoff (which indicates our model is not overfitting) when compared to the learning curve of experiment 1 (Figure 25(a)), where there is high variance indicated by the large gap between training and cross-validation score.

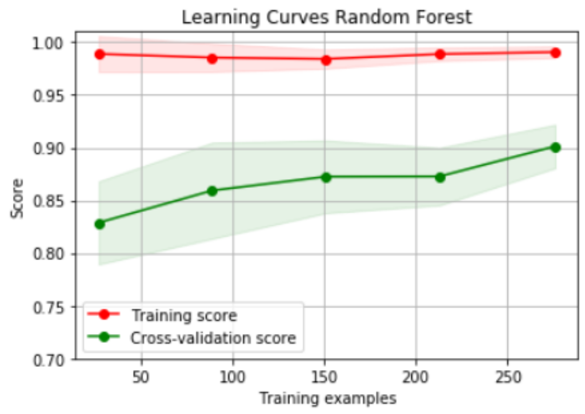


(a) : ROC-AUC with Experiment #1

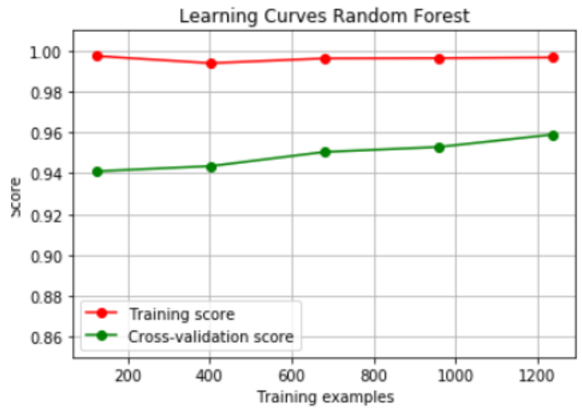


(b) : ROC-AUC with Experiment #2

Figure 24: ROC AUC on stratified 10 fold cross-validation for harsh brakes using random forests

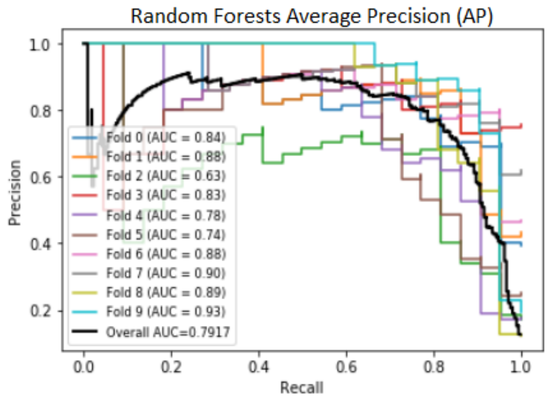


(a): Learning Curves Experiment #1

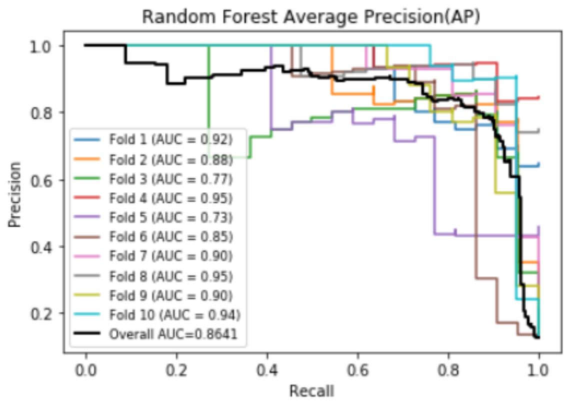


(b): Learning Curves Experiment #2

Figure 25: Learning Curves on stratified 10 fold cross-validation for harsh brakes using random forests



(a) Precision Recall Curve Experiment #1



(b) Precision Recall Curve Experiment #2

Figure 26: Precision-Recall Curves on stratified 10-Fold cross-validation for harsh brakes using random forests

When comparing the Average Precision (AP) for both experiments (Figure 26), it also suggests that our Experiment#2 is a better setup to classify harsh brakes with a value of 0.86, averaged over stratified 10-fold cross-validation.

The Matthews Correlation Coefficient (MCC) calculated for our model for both experiments can be seen in Table 3.

The results for harsh brakes with random forest classifier are also displayed below in Table 3.

Metric	*Experiment #1*	*Experiment #2*
Average Precision (AP)	0.79	**0.86**
ROC-AUC	0.79	**0.97**
MCC	0.76	**0.83**

Table 3: Averaged results over stratified 10-fold cross-validation using Random Forests for both experiments

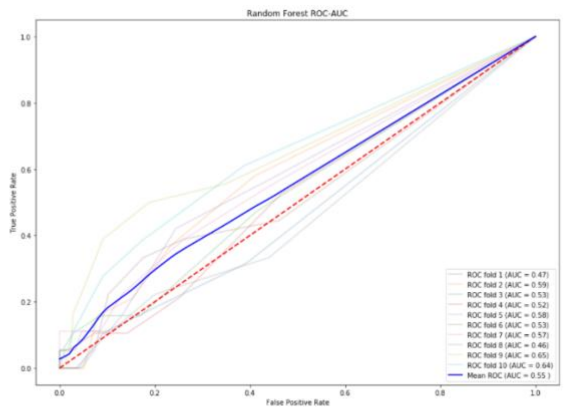
(b) Sharp Corners

For our chosen classifiers, we chose Random Forest as our classifier of choice as it performed better than Decision Trees based on our evaluation metrics, ROC AUC, AP and MCC.

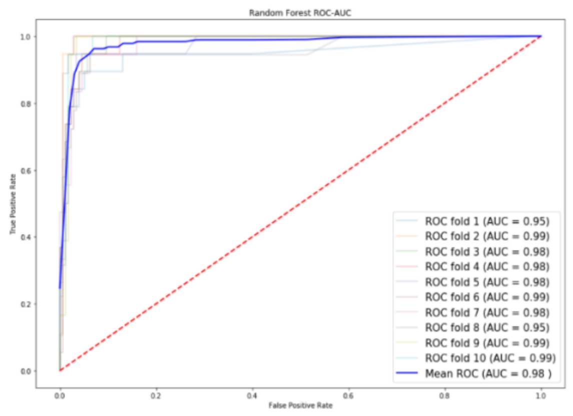
Figure 27 (a), we can infer from the AUC of 0.55, that our classifier is random, i.e., it does not have the ability to differentiate between normal and sharp corner events, whereas features obtained from our experiment 2, suggest that our classifier has a higher chance of classifying normal and sharp corner events. (Figure 27 (b)). However, since a ROC is not sensitive to class imbalance, we will look at our other evaluation metrics to evaluate our classifier.

Figure 28 (a) illustrates our random forest model with experiment 1 has high variance, which indeed means our model is overfitting, and cannot generalize new data well. A solution to this will be providing it with more training data. Figure 28 (b) with our experiment 2, our training and cross-validation learning scores converge together, with a higher '$f1 - score$', which shows our model can generalize new data and has a good trade-off between bias and variance.

Figure 29 (a)**,** illustrates a very bad average precision (which is in line with our other evaluation metrics). However, with experiment 2, Figure 29 (b)**,** our average precision increases up to 83%, which shows our classifier performs better with experiment 2 setup. This shows how only 4 relevant features helped to model our classification problem better and improve our f1 scores.

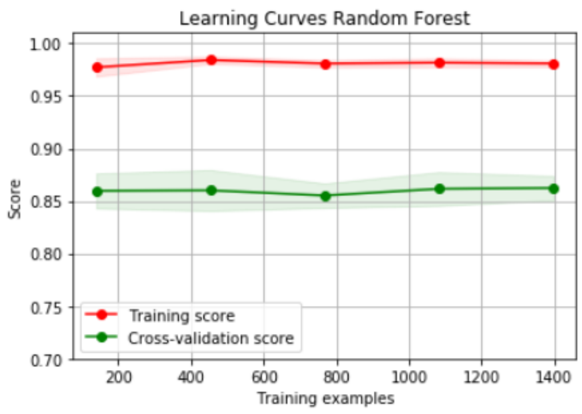


(a) : ROC-AUC with Experiment #1

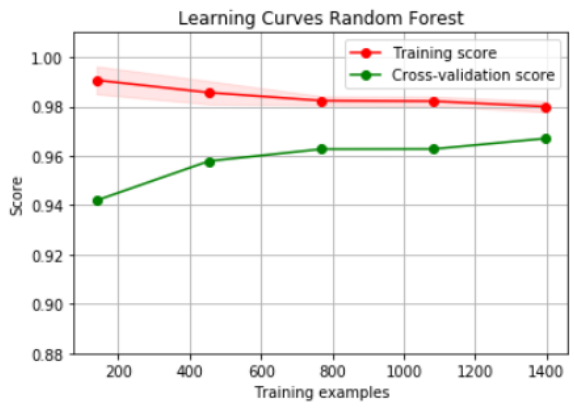


(b) : ROC-AUC with Experiment #2

Figure 27: ROC AUC on stratified 10 fold cross validation for sharp corners using random forests

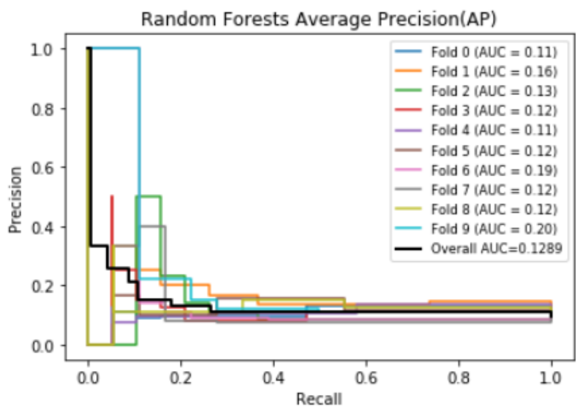


(a) : Learning Curve with Experiment #1

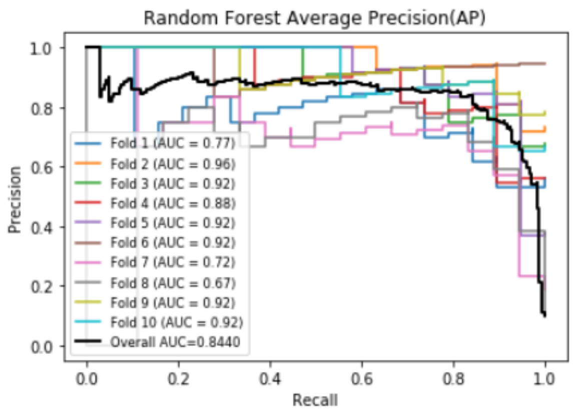


(b) : Learning curve with Experiment #2

Figure 28: Learning Curves on stratified 10 fold cross validation for sharp corners using random forests



(a) Precision Recall Curve Experiment #1



(b) Precision Recall Curve Experiment #2

Figure 29: Precision Recall Curves on stratified 10-Fold cross validation for sharp corners using random forests

Matthews Correlation Coefficient (MCC) is shown in Table 4, for both experiments using random forest classifiers.

The results for sharp corners with random forest classifier are also displayed below in Table 4.

Metric	*Experiment #1*	*Experiment #2*
Average Precision (AP)	0.12	**0.84**
ROC-AUC	0.55	**0.98**
MCC	0.13	**0.79**

Table 4: Averaged results over stratified 10-fold cross-validation using random forests for both experiments

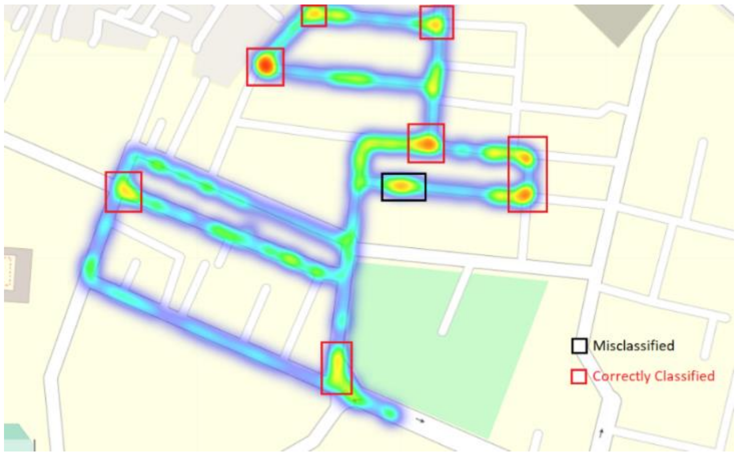
Table 4 shows a drastic improvement in all three scores for experiment 2 setup. This just further proves that a sharp corner is a harder problem to learn and hence needs a more complex model, thus more descriptive features that reflect the problem we are trying to classify.

7.2 Test on Unseen Data

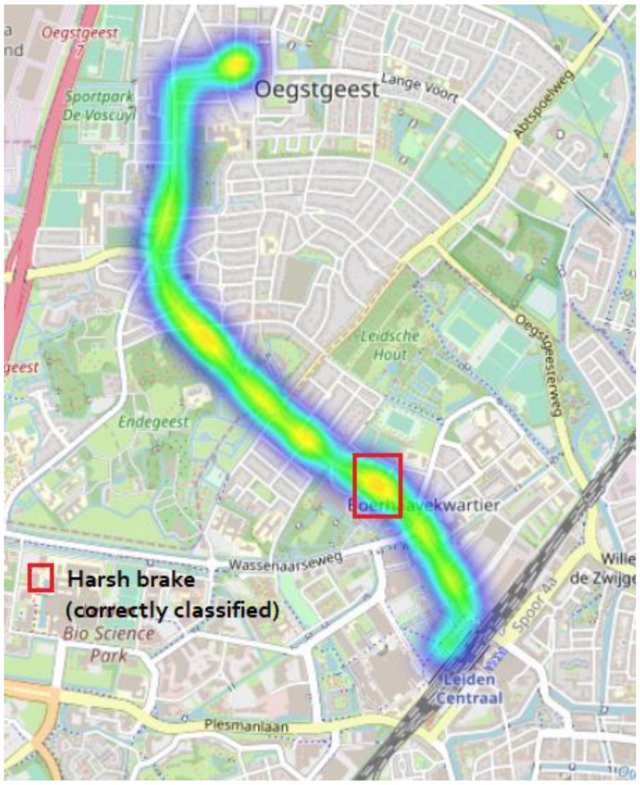
To test both our models, we subjected them to realistic driving scenarios and evaluated their performance.

The test driver in the Netherlands collected realistic driving data, which was handed to us to further analyze. They used iPhone X as the phone and Volkswagen Polo car for collecting data.
For verifying the results of these drives, the driver was asked to record himself while driving and mention exactly when a sharp corner event took place. This ground truth data was then matched to the sharp turns data and further analyzed.
The intensity shown on the map (Figure 30(a) (b)) is the energy on the acceleration x-axis calculated with each event classified as a "sharp corner" by the classifier.
Figure 30 (a) shows 8 correctly classified sharp corners that were performed during the drive. However, one was misclassified as a sharp corner. Exactly why our model misclassifies it as a sharp corner is hard to pinpoint at this stage.

Figure 30 (b) shows normal driving, with one correctly classified harsh brake event. This drive was 15 minutes long and only one harsh brake event was performed. The classifier does not misclassify any events. This shows our model works well on identifying non-critical events or normal driving events and harsh brake events. This could be because of imbalance in our dataset, the model learnt the characteristics of normal driving better, and that harsh brake is easier to learn for the model than sharp corner events.



(a) Testing model on unseen data indicating correctly classified and misclassified sharp corner events



(b) Testing model on unseen data with no misclassifications

Figure 30: Visualization and classification of driving events on unseen data

7.3 Conclusion and Future Work

In this research, our goal was to train a machine learning model that can classify harsh brakes and sharp corners, using smartphone sensor data. One major limitation and challenge that came with this research was to collect ground truth data. Driving recklessly on busy roads to mimic realistic situations to collect harsh brake and sharp corner events, is dangerous as well as comes with potentials risks to the car, as well as the driver and others driving on the road.

However, in our study, we used six drivers, three different mobile platforms having different sensor calibrations as well as driving with three different cars on different types of roads to introduce as much variety in the dataset as possible.

Dealing with highly imbalanced sensor dataset comes with few challenges. On top of that, we were also dealing with noisy sensor data retrieved from smartphones.

However, we were able to mitigate these challenges by using signal smoothing techniques, state of the art signal processing algorithms, as well as using proven feature generation and selection techniques to create and select features that are the most prominent in classifying a harsh brake and sharp corner event from a normal driving event.

We chose a random forest classifier, as our classifier of choice, since it could classify harsh brakes and sharp corners with a higher average precision (0.86 and 0.83 for harsh brake and sharp corner events respectively) than a decision tree, considering there were approximately 200 harsh brake and 180 sharp corner events to learn from. We also performed various tests to evaluate the model that further gives us insight that our classifier is not overfitting or underfitting.

One important question that arises from this research is more from a psychological aspect of what is considered a harsh brake or a sharp corner. Defining a sharp corner or a harsh brake, choosing a universal threshold is naïve and cannot be applied in all scenarios and cases. During our manual data exploration, we realized that these events are different when a driver is driving in the Netherlands versus when the driver is in India. These driving events indeed depend on various factors including the type of roads the car is driven on, the amount of traffic on roads, as well as the type of driver behind the wheel.
Although we experimented with many variables, some seem to have more effect on the data than the others, like type of road and type of driver. For some drivers we observed the harsh brakes were very prominent, and indeed dangerous.

One interesting aspect that could take this research further, could be to learn country-based thresholds for these driving events, that could help define a harsh brake or sharp corner event with respect to the country the driver is driving in. This indeed could be a starting point to get a baseline for thresholds of what is considered a harsh brake or sharp corner event.

The model presented in this thesis can be improved with more realistic driving data, as well as tested with more driving data. More research and experimentation can be done with the position of the phone in the car, and the orientation of the phone, i.e., in landscape mode.

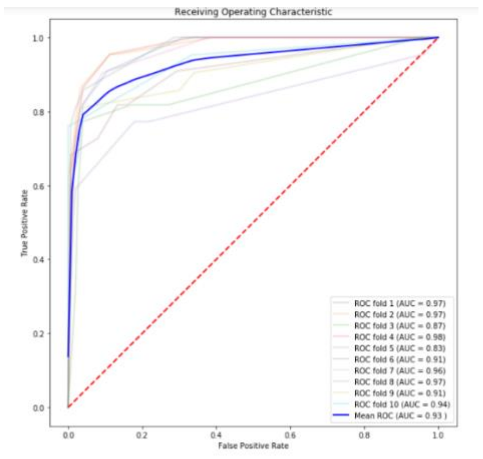
Since getting labelled ground truth data is expensive and time-consuming, unsupervised learning can be considered on the dataset, and these abnormal driving events can be treated as anomalies. Unsupervised learning is cheaper than supervised learning in terms of getting ground truth data. However, unsupervised learning comes with its own challenges, and since the data generated from smartphone sensors are noisy, it could be that we detect anomalies that are not necessarily just harsh brakes and sharp corners. Other critical events also take place on the roads. Due to time constraints of this thesis, unsupervised learning could not be experimented with.

Reproducibility

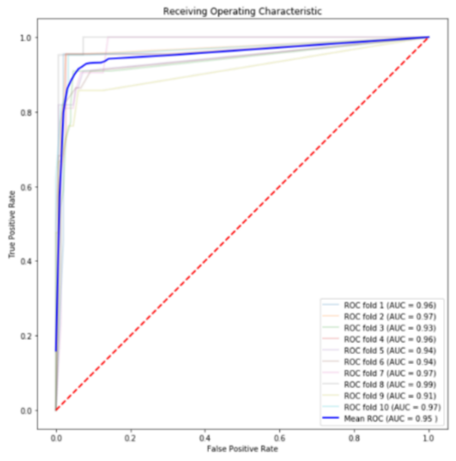
The scripts used for the experiments of this thesis are owned and controlled by TomTom B.V.(Amsterdam). The data used for this thesis is under the ownership of TomTom B.V.(PU Driver Apps Unit), for accessing the data they may be contacted.

APPENDIX

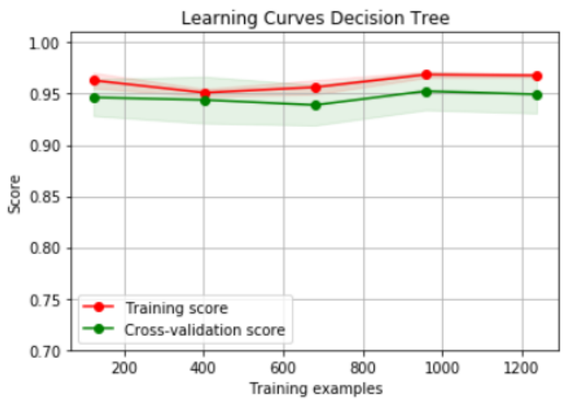
(a) Harsh Brakes - Results from Decision Tree Classification (Experiment 1 and 2)



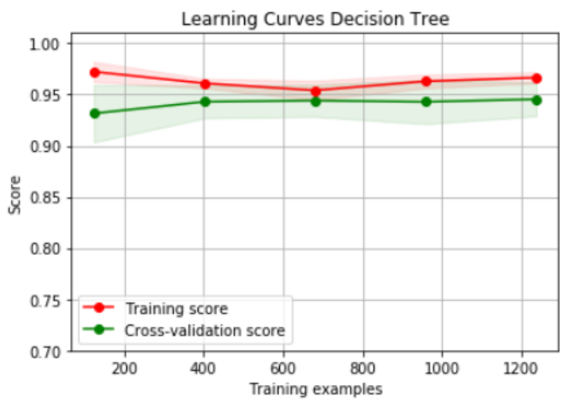
(a) : ROC-AUC with Experiment #1



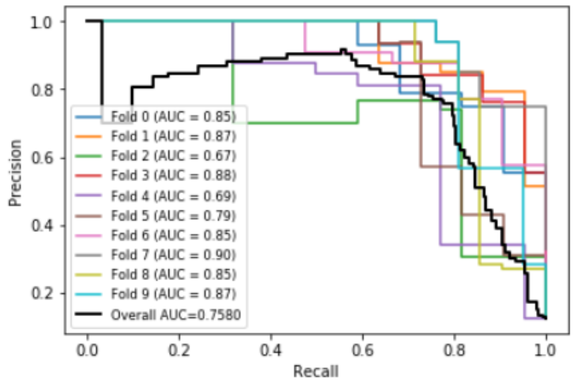
(b) : ROC-AUC with Experiment #2



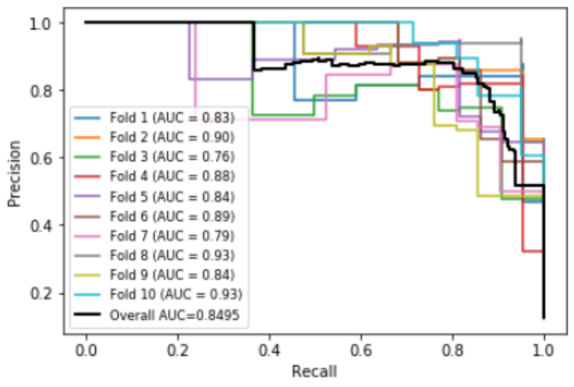
(c): Learning Curve with Experiment #1



(d): Learning Curve with Experiment #2



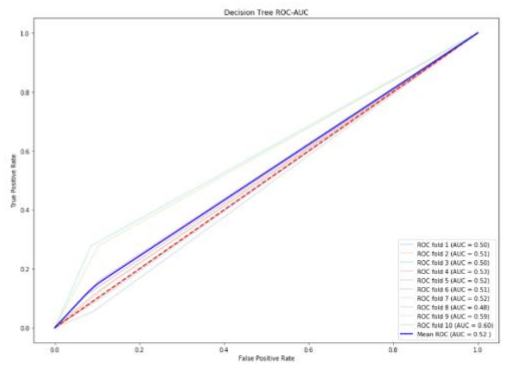
(e): AP with Experiment #1



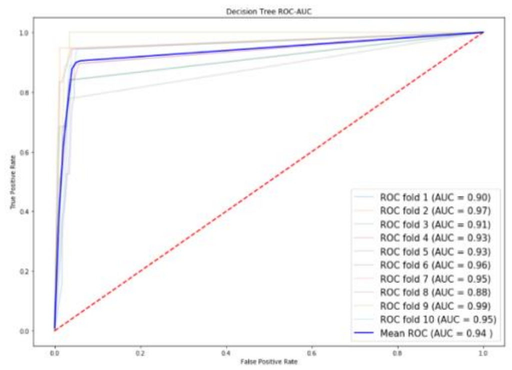
(f): AP with Experiment #2

Figure 21: Model Evaluation with Decision Trees on stratified 10-fold cross validation

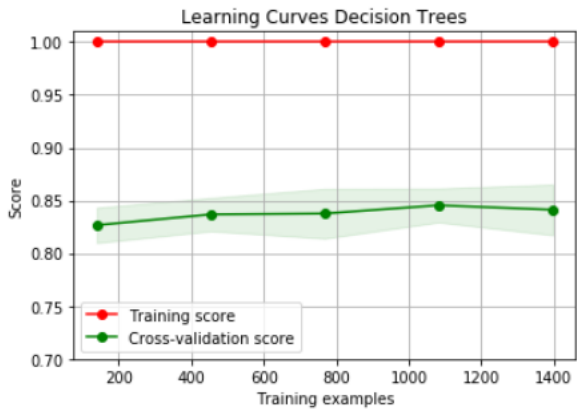
(b) Results from Decision Tree Classification (Experiment 1 and 2)



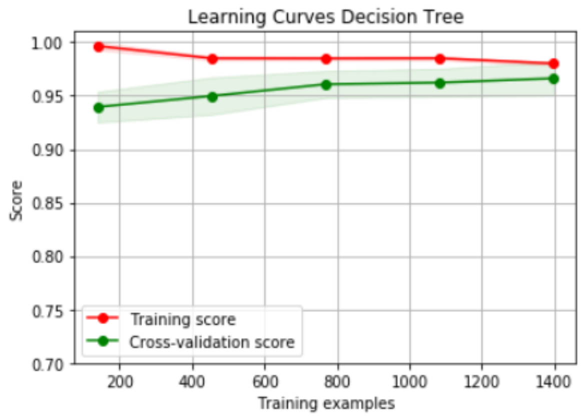
(a) : ROC-AUC with Experiment #1



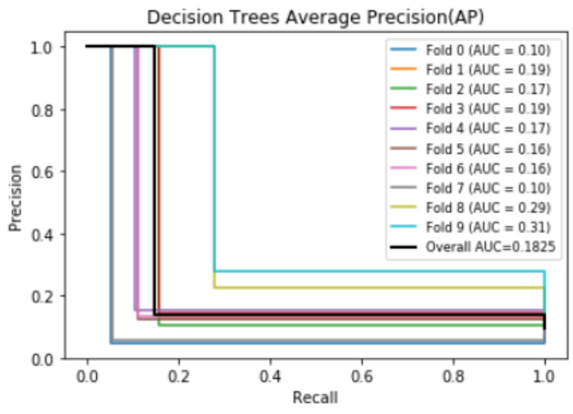
(b) : ROC-AUC with Experiment #2



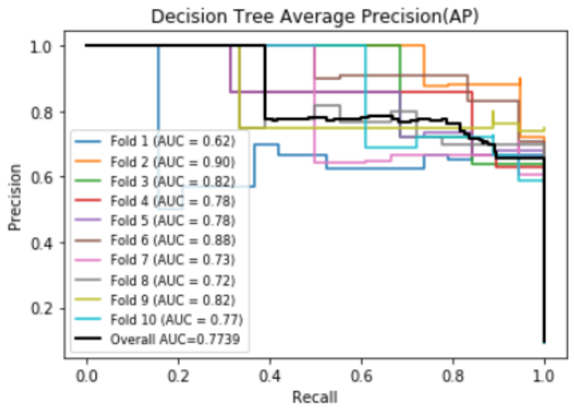
(c): Learning Curve with Experiment #1



(d): Learning Curve with Experiment #2



(e): AP with Experiment #1



(f): AP with Experiment #2

BIBLIOGRAPHY

[1] Road Safety Facts. (2018). Retrieved from https://www.asirt.org/safe-travel/road-safety-facts/.

[2] Aljaafreh, Ahmad & Alshabatat, Nabeel & Najim Al-Din, Munaf. (2012). Driving style recognition using fuzzy logic. 2012 IEEE International Conference on Vehicular Electronics and Safety, ICVES 2017. 460-463. 10.1109/ICVES.2012.6294318.

[3] Fazeen, Mohamed & Gozick, Brandon & Dantu, Ram & Bhukhiya, Moiz & Gonzalez, Marta C.. (2012). Safe Driving Using Mobile Phones. Intelligent Transportation Systems, IEEE Transactions on. 13. 1462-1468. 10.1109/TITS.2017.2187640.

[4] Johnson, Derick & Trivedi, Mohan. (2015). Driving Style Recognition Using a Smartphone as a Sensor Platform. Intelligent Transportation Systems (ITSC), 2015 14th International IEEE Conference On. 10.1109/ITSC.2015.6083078.

[5] Senin, Pavel. (2009). Dynamic Time Warping Algorithm Review.

[6] Ma, Chunmei & Dai, Xili & Zhu, Jinqi & Liu, Nianbo & Sun, Huazhi & Liu, Ming. (2017). DrivingSense: Dangerous Driving Behavior Identification Based on Smartphone Autocalibration. Mobile Information Systems. 2017. 1-15. 10.1155/2017/9075653.

[7] Ferreira, J., Júnior, Carvalho, E., Ferreira, B. V., de Souza, C., Suhara, Y., Pentland, A., & Pessin, G. (2017). Driver behavior profiling: An investigation with different smartphone sensors and machine learning. *PloS one*, *12*(4), e0174959. doi:10.1371/journal.pone.0174959

[8] Priyadarshini, M. (2019, April 20). Which Sensors Do I Have In My Smartphone? How Do They Work? Retrieved September 3, 2019, from https://fossbytes.com/which-smartphone-sensors-how-work/.

[9] Broder, P. (2014, October 4). Sensor Play - Data Recorder. Retrieved from https://apps.apple.com/us/app/sensor-play-data-recorder/id921385514.

[10] Apple iPhone 6 and iPhone 6 Plus Teardown. (1970, September 27). Retrieved from https://www.techinsights.com/blog/apple-iphone-6-and-iphone-6-plus-teardown.

[11] Samsung Galaxy S8 Teardown (SM-G950W). (1970, September 27). Retrieved from https://www.techinsights.com/blog/samsung-galaxy-s8-teardown-sm-g950w.

[12] Ashraf, I., Hur, S., Shafiq, M., & Park, Y. (2019). Floor Identification Using Magnetic Field Data With Smartphone Sensors. *Sensors (Basel, Switzerland)*, *19*(11), 2538. doi:10.3390/s19112538

[13] (n.d.). Retrieved from https://wiki.aalto.fi/display/ITSP/Windowing.

[14] P. Duhamel, M. Vetterli, Fast fourier transforms: A tutorial review and a state of the art, Signal Processing, Volume 19, Issue 4, 1990, Pages 259-299, ISSN 0165-1684, https://doi.org/10.1016/0165-1684(90)90158-U.(http://www.sciencedirect.com/science/article/pii/016516849090158U)

[15] Safavian, S.R., & Landgrebe, D.A. (1991). A survey of decision tree classifier methodology. *IEEE Trans. Systems, Man, and Cybernetics, 21*, 660-674.

[16] Louppe, Gilles. (2014). Understanding Random Forests: From Theory to Practice. 10.13140/2.1.1570.5928.

[17] Raschka, S., & Mirjalili, V. (2017). *Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow*. Birmingham: Packt Publishing.

[18] Green, D. M., & Swets, J. A. (1974). Signal detection theory and psychophysics. Oxford, England: Robert E. Krieger.

[19] Davis J, Goadrich M. The relationship between precision-recall and ROC curves. In Proceedings of the 23rd international conference on machine learning (2006;pp. 233–240). ACM.

[20] T. Fawcett, “ROC graphs: Notes and practical considerations for re-searchers,” Machine learning, vol. 31, no. 1, pp. 1–38, 2004.

[21] Anzanello, Michel & Fogliatto, Flavio. (2011). Learning curve models and applications: literature review and research directions. International Journal of Industrial Ergonomics, 41, 573-583. International Journal of Industrial Ergonomics - INT J IND ERGONOMIC. 41. 573-583. 10.1016/j.ergon.2011.05.001.

[22] Boughorbel, S. & Jarray, Fethi & El-Anbari, Mohammed. (2017). Optimal classifier for imbalanced data using Matthews Correlation Coefficient metric. PLOS ONE. 12. e0177678. 10.1371/journal.pone.0177678.

[23] Overview on extracted features. (n.d.). Retrieved from https://tsfresh.readthedocs.io/en/latest/text/list_of_features.html.

[24] Kursa, Miron & Rudnicki, Witold. (2010). Feature Selection with Boruta Package. Journal of Statistical Software. 36. 1-13. 10.18637/jss.v036.i11.

[25] sklearn.feature_selection.RFE. (n.d.). Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html#sklearn.feature_selection.RFE.

[26] sklearn.manifold.TSNE. (n.d.). Retrieved from https://scikitlearn.org/stable/modules/generated /sklearn.manifold.TSNE.html.

[27] Maaten, L.V., & Hinton, G.E. (2008). Visualizing Data using t-SNE.

[28] sklearn.manifold.TSNE. (n.d.). Retrieved from https://scikit-learn.org/stable/modules/generated /sklearn.manifold.TSNE.html.

[29] Jolliffe, I. T., & Cadima, J. (2016). Principal component analysis: a review and recent developments. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, *374*(2065), 20150202. doi:10.1098/rsta.2015.0202

[30] A. Al-Thubaity, N. Abanumay, S. Al-Jerayyed, A. Alrukban and Z. Mannaa, "The Effect of Combining Different Feature Selection Methods on Arabic Text Classification," *2013 14th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, Honolulu, HI, 2013, pp. 211-216.doi: 10.1109/SNPD.2013.89

[31] H. Eren, S. Makinist, E. Akin, and A. Yilmaz, “Estimating driving behavior by a smartphone.” in Intelligent Vehicles Symposium (IV). IEEE, 2012, pp. 234-239

[32] Narkhede, S. (2019, May 26). Understanding AUC - ROC Curve. Retrieved from https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5.