**Acknowledgements**

First and foremost we would like to express my immense gratitude to Ms. Savita K Shetty, Department Of Information Science, MSRIT whose sincerity and encouragement we will never forget. She has been our support as we overcame all our difficulties in the completion of this project. She had kind concern and consideration regarding our project.

We wish to express our sincere thanks to Dr. Vijaya Kumar B.P, Professor and HOD of Information Science and Engineering who guided and motivated us to complete this project. We would like to thank him for providing us with all the necessary resources for this project.

We would also like to thank our principal Dr. NVR Naidu for his support and encouragement. This work would not have been completed without the invaluable help provided by the other teaching and non-teaching staff members of the ISE department.

All sentences or passages quoted in this report from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations which are not the work of the author of this report have been used (where possible) with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amount to plagiarism and will be considered grounds for failure in this project and the degree examination as a whole.

## Abstract

To collect data from the car and use machine learning to analyse and label the driver as either ‘safe’ or ‘rash’ for the recently recorded trip. This analysis is done on a server after the trip has been completed and data is collected. The Server contains various scripts to execute and receive files and send data. Support Vector Machine (SVM) Algorithm is used to perform Classification with a threshold value to obtain the output. The final result after classification is then made available to view on the user’s android mobile phone via a Driver\_Analysis Application.

**Contents**

[**1 Introduction** 1](#_Toc3278)

[1.1 Motivation 1](#_Toc3279)

[1.2 Scope 1](#_Toc3280)

[1.3 Objectives 2](#_Toc3281)

[1.4 Proposed Model 2](#_Toc3282)

[1.5 Organisation of Report 2](#_Toc3283)

[**2 Literature Review** 3](#_Toc3284)

[**3 System Analysis and Design** 6](#_Toc3285)

[**4 Modelling and Implementation** 14](#_Toc3286)

[**5 Testing, Results and Discussion** 24](#_Toc3287)

[**6 Conclusion and Future Work** 29](#_Toc3288)

**References 7**

**List of Figures**

1. Fig 2.1 Flow Chart of driving behaviour analysis
2. Fig 2.2 Energy saving driving assessment method
3. Fig 2.3 Testing System Architecture
4. Fig 3.1 System Design
5. Fig 3.2 Torque App
6. Fig 3.3 Home Page of App
7. Fig 3.4 File browser of the App
8. Fig 3.5 Progress Bar of App
9. Fig 3.6 Result- App
10. Fig 3.7 Putty
11. Fig 4.1 Use Case Diagram
12. Fig 4.2 Sequence Diagram
13. Fig 4.3 PHP Script
14. Fig 4.4 Normalization of Data
15. Fig 4.5 Architecture of SVM
16. Fig 4.6 Visual Representation of Gamma Value
17. Fig 4.7 Pre Processing of Data
18. Fig 4.8 Testing with sample data
19. Fig 4.9 Android App Code Structure
20. Fig 5.1 SVC Model and Accuracy Computation
21. Fig 5.2 Test input after reshaping
22. Fig 5.3 Prediction of Test Input
23. Fig 5.4 Final Result along with predicted list
24. Fig 5.5 Correlation
25. Fig 5.6 Android App Result

**Chapter 1**

# Introduction

## Motivation

In [5]India, the number of registered motor vehicles were 21,00,23,289 as on 31-3-2015. The number of vehicles on the road grew at a rate of 10.01% between the year 2014 and 2015. At the same time, the number of ‘Traffic Accidents’ in the country have increased by 3.1% (from 4,81,805 in 2014 to 4,96,762 in 2015) and the fatalities in road accidents have increased by 5.1% (from 1,41,526 in 2014 to 1,48,707 in 2015).

A cause wise analysis of road accidents by NCRB revealed that most of the road accidents were due to over speeding, accounting for 43.7% of total accidents (2,02,882 out of 4,64,474 cases) which caused 60,969 deaths and injuries to 2,12,815 persons. Dangerous/careless driving or overtaking caused 1,46,059 accidents which resulted in 48,093 deaths and injuries to 1,51,231 persons during 2015.

In order to reduce the majority of road accidents in the future, machine learning can be used to analyse driving data and provide valuable feedback to the driver about the quality of their driving using a mobile application which is easily accessible to them. The result lets the driver know whether or not their driving is safe so that they can choose to change their driving behaviour and potentially help reduce accidents and even possibly prevent deaths as a result of dangerous/careless driving.

* 1. **Scope**

This project uses driving information collected from various trips and provides a simple feedback of whether the driving is ‘safe’ or ‘rash’ based on supervised machine learning models. The training is done on real world data obtained by performing various driving tests.

Using the app, the user can add several cars to their account and upload the data collected by the OBD device from their vehicle(s).

The data is then sent to the server where the machine learning algorithm runs and sends the result back to the app on the user’s phone.

## Objectives

By implementation of this project, we aim to fill these vacancies in driving analysis:

* Drivers currently have no real fixed, established standards for safe or unsafe driving.
* There are no simple and easily accessible ways to check whether a driver is good or not.

Once obtained, the driver will be able to know if they are a safe driver or not. This is done in the hope of achieving reduced road traffic accidents and hence.

## Proposed Model

The proposed model involves the following major steps:

* The user/driver can use an [1]On-Board Diagnostics device (OBD-II) to collect the raw data (comprising of up to 50 parameters) from the vehicle engine control unit (ECU).
* The collected data by the OBD.

## Organisation of Report

In order to explain the developed system, the following sections are covered:

* **Literature Review** describes the study of the existing systems and techniques taken into account prior to development of the proposed system.
* **System Analysis and Design** provides a detailed walk through of the software engineering methodology adopted to implement the model, an overview of the system and the modules incorporated into the system
* **Modelling and Implementation** provides a deeper insight into the working of the model. The various modules and their interactions are depicted using relevant descriptive diagrams.
* **Testing** the model to ensure bug/error free model along with the **Results** obtained. **Discussion** then provides detailed analysis on quality assurance measures.
* **Conclusion** about the Results obtained after successfully running the model and **Future Scope** of the model is highlighted.

**Chapter 2**

# Literature Review

[1] Shi-Huang, Jeng-Shyang Pan and Kaixuan Lu’s research paper The Driving Behaviour Analysis Based on Vehicle OBD information and AdaBoost algorithm paper implements a novel driving behaviour analysis method based on the OBD information and ADA Boost Algorithm. It collects vehicle operation information including Speed, RPM, Throttle Position, and calculated engine load via the OBD interface. It then makes use of ADA Boost algorithms to create a driving behaviour classification model. Proposed method has the potential real world applications such as assistance during driving.

Analysis method consists of driving operation acquisition module, data processing module, AdaBoost classification modules. Good driving data and Bad driving data should be collected as training set. Analysis data is divided into training and testing. Classification is done using AdaBoost.

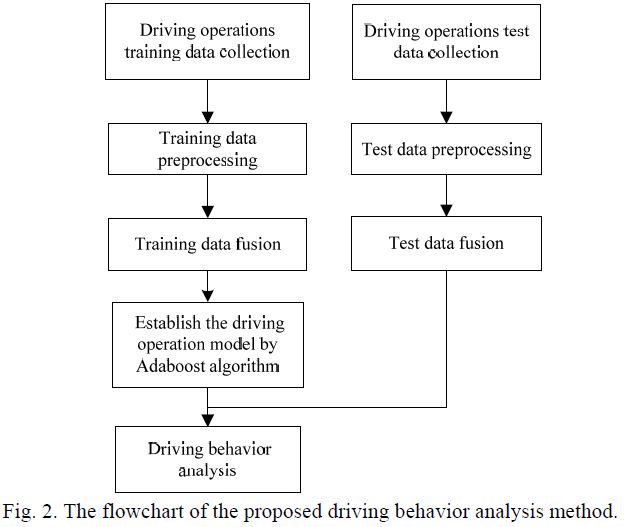


Fig 2.1 Flow Chart of proposed driving behaviour analysis method

[2] Yu-Jen Pan, Te-Cheng Yu, and Rung-Shiang Cheng ‘s research paper Using OBDII Data to Explore Driving Behaviour Model analyzes the driving behaviour data through the vehicle preloading equipment and to analyze the factors that affect safe driving. With the development in improvement of the internet huge amounts of data is constantly rising and car networking data is one of them. OBD II provides a database for analyzing user driving behaviour. The results obtained can be the basis for the study of individual driving behaviour model. Logistic Regression is used to analyze the data as calculation is small and operation speed is faster. OBD is used to establish the risk driving evaluation model and to explore the relevant factors of dangerous driving behaviour. It was inferred that maximum speed led to increased risk of driving risk.

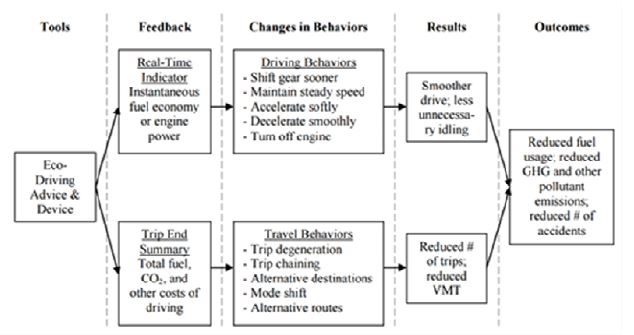


Fig 2.2 Energy saving driving assessment method

[3] Shu-LinHwang, and Yen-Jen Chen’s research paper Combining OBD Technology with Acceleration Sensor to Analyze Aggressive Driving Behaviour attempts to use smart-phones to capture OBD II data and integrate the acceleration sensor on the phone itself so that the model can easily judge whether the driver has the habit of aggressive driving. Classification is done based on of the following situations such as improper direction, over speeding, not paying attention to state of car, etc. These conditions are observed and a aggressive driving behaviour featuring model can be established.

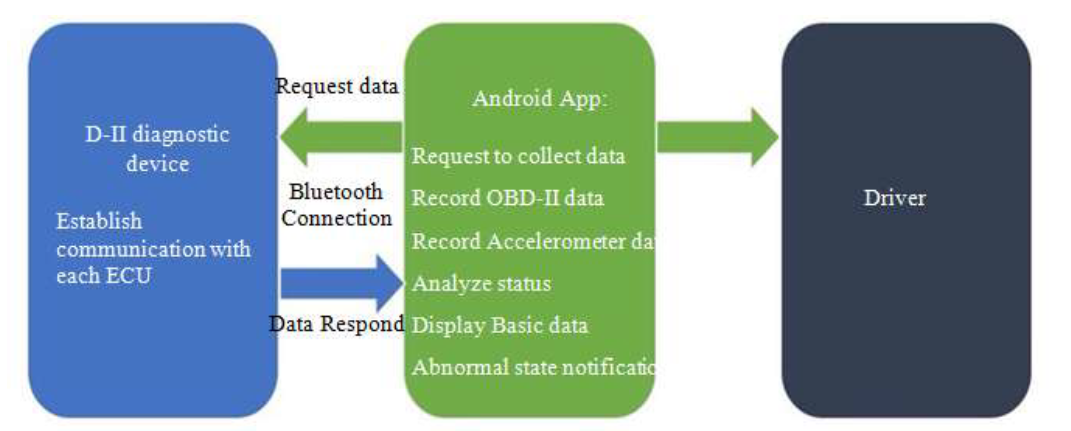


Fig 2.3 Testing system architecture

**Chapter 3**

# System Analysis and Design

**3.1 Overview**

The overview section covers the methods used to design and develop the system. There are two major components in the system: interface for data collection and classification of driver into 2 categories. System Analysis and Design consists of Analysis and Design of individual components and defining the Development Life Cycle(SDLC).

**3.2 The System**

The section covers the content specific to the system in detail starting from the requirements up to the collection of data.

**3.2.1 System Characteristics**

The Android Application is used to upload the dataset to the server and display the final result after processing. Bluetooth Connectivity must be established between the OBD2 and the Application. The interface is devoid of clutter making it easier for users who are not technically experienced.

The Machine Learning algorithm is executed as a Server side script. The complexity of the algorithm is:

* [4] SVM Classification has an efficiency of O(nSVd).

(Where SVd is the number of Support Vectors)

**3.2.2 System Assumptions, Constraints and Dependencies**

**Assumptions:**

* The car has on-board OBD or is flexible to the addition of ODB2 device.
* The driver must have an [3]Android device running the Application.

**Constraints:**

* The data collected is not entirely accurate as human and mechanical errors are possible.

**Dependencies:**

* The user device must be running Android 4.0 Ice Cream Sandwich or higher for complete execution.
* Internet Connectivity is required.
* Bluetooth Connectivity between the OBD and user device.
* External Storage permissions required on the device.

**3.3 Requirements**

The requirements are divided into Functional and User Interface Requirements.

**3.3.1 Functional Requirements**

* An algorithm that classifies the driving into one of 2 categories and also perform prediction of unknown data.
* A server of some kind for Machine Learning script execution.
* Upload of file to server.
* Requirement to view the result obtained.

**3.3.2 User Interface**

* Updates of the new car data.
* Informative UI of Application.
* Ease of use of the Application.

**3.4 System Design**

The system is designed as shown in Fig 3.1. The system consists of a client-server architecture with the bulk of processes running on the server.

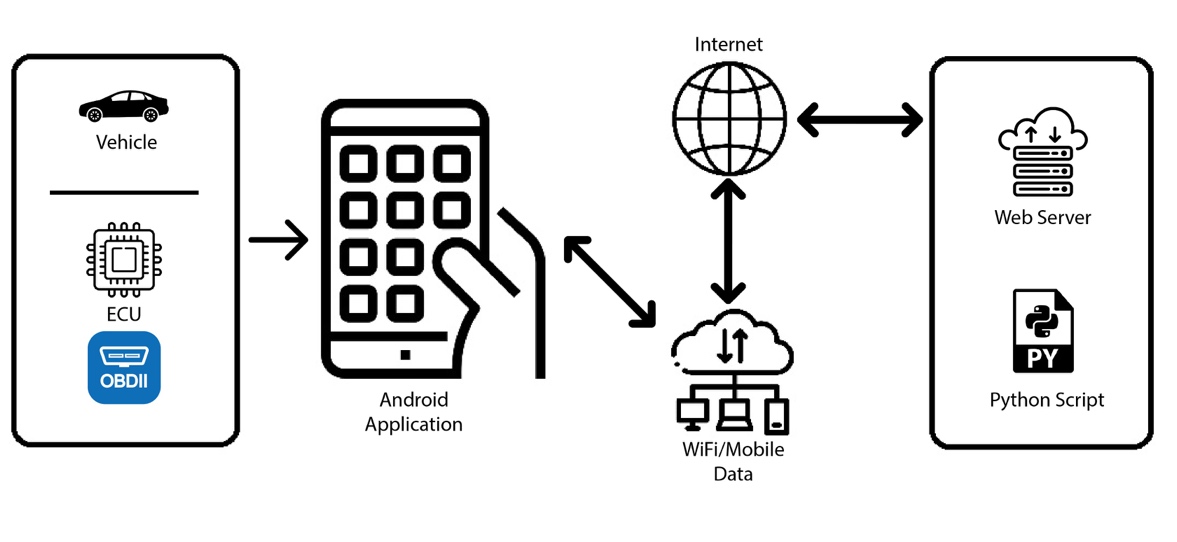


Fig 3.1 System Design

The OBD device connects to the user’s phone via Bluetooth. Once the car is started the the app sends commands to the OBD to receive the needed data. As the request is sent the response is received by the Application and stored as a .csv file. The file is uploaded to the Server.

The Server is hosted on a t2.micro instance provided by AWS EC2.

The data is then used by the machine learning algorithm to predict the output of the received input data. This output is then sent to the Application to be displayed to the user.

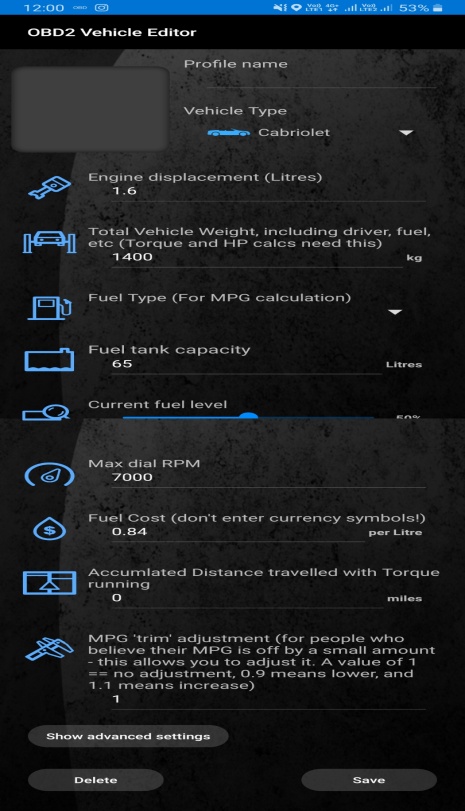
**3.5 Components of the System**

The following are the various components of the system:

* On Board Diagnostics(OBD)
* Android Application
* Server
* Classification using ML

**3.5.1 On Board Diagnostics**

OBD is refers to diagnostic and reporting capability of a vehicle. OBD gives the users various details about the vehicle that it captures. They provide real-time data that can be processed immediately or at a delayed time. OBD2 provides data from the Engine Control Unit (ECU). The data collected by the OBD2 can be used for a variety of services such as a diagnostic tool to check the health and maintenance of the vehicle. It also informs the users of possible internal flaws in the vehicle that may go undetected otherwise.

**

*Fig3.2 Torque App*

**3.5.2 Android Application**

The UI Interface is built on Android platform using Android Studio. The application is used to send data as .csv file to the server. The Application designed is made to be easy to use and uncluttered. The upload and download of data is handled with ease by the respective buttons that perform the aforementioned operations. The TextField in the Application displays the output of the computation to the final user. The Application supports a min SDK version of 14 and min Android version of Ice Cream Sandwich 4.0.

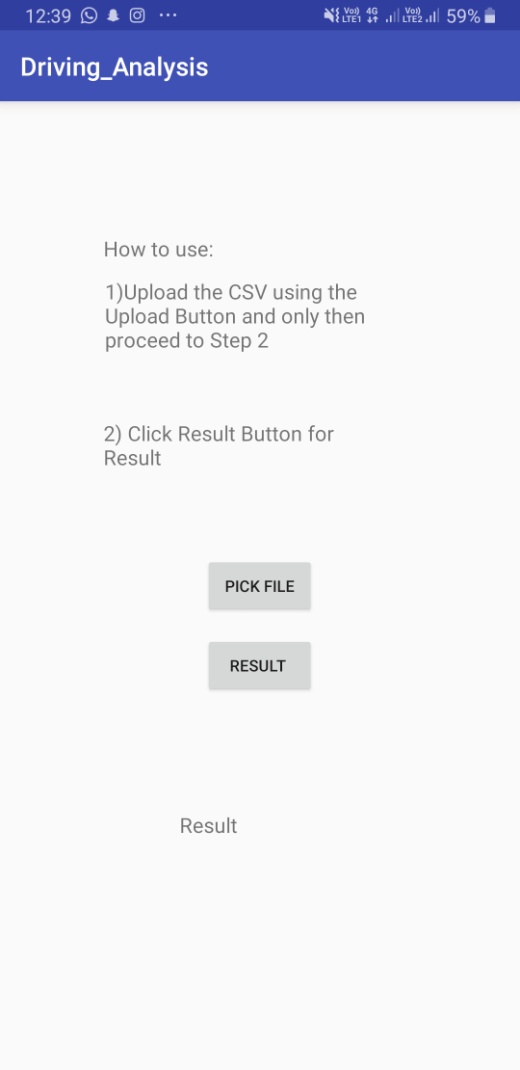
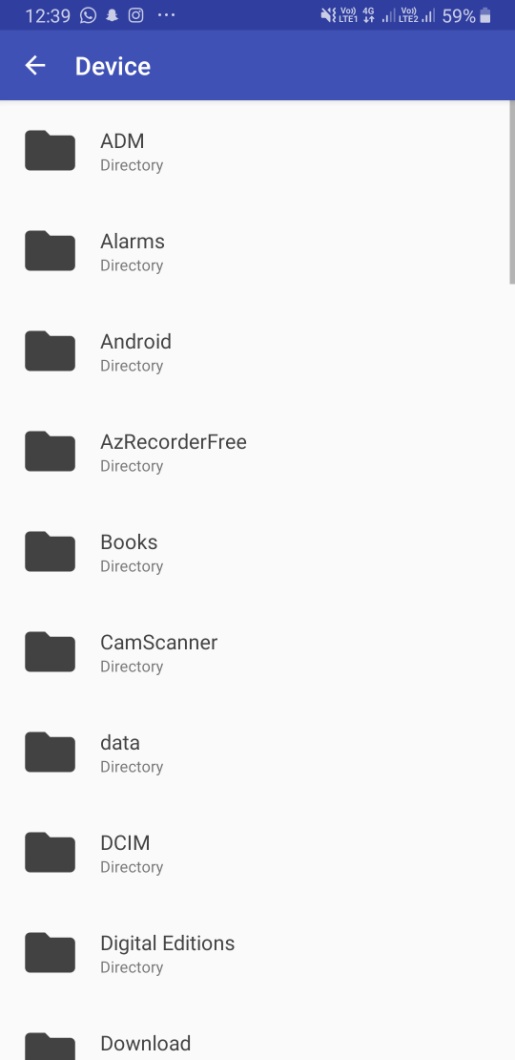
 

Fig 3.3-3.4 Homepage of the app, File browser of the application

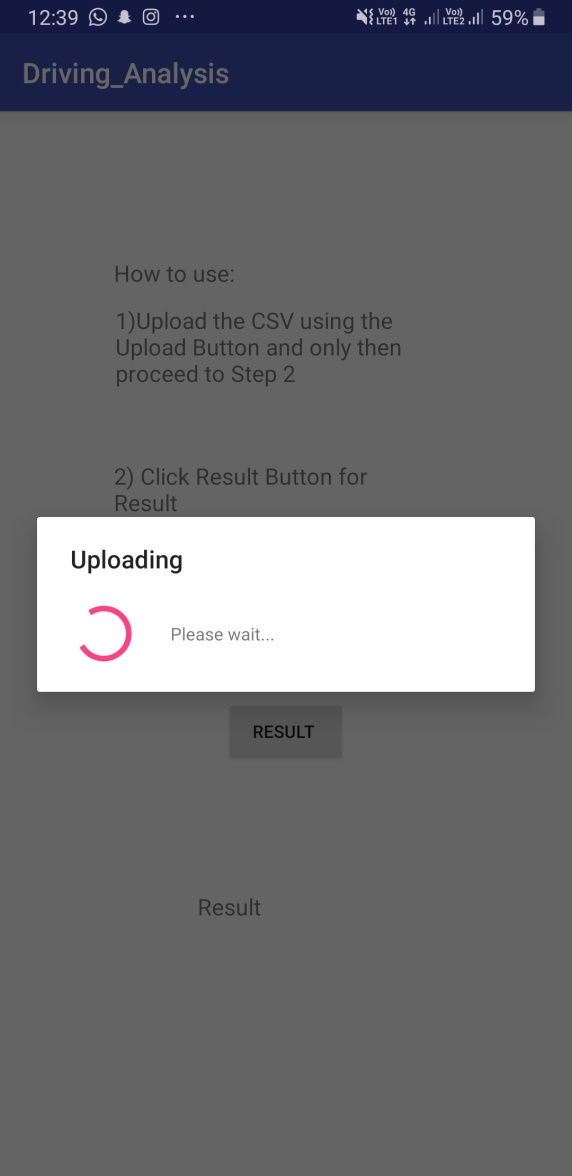
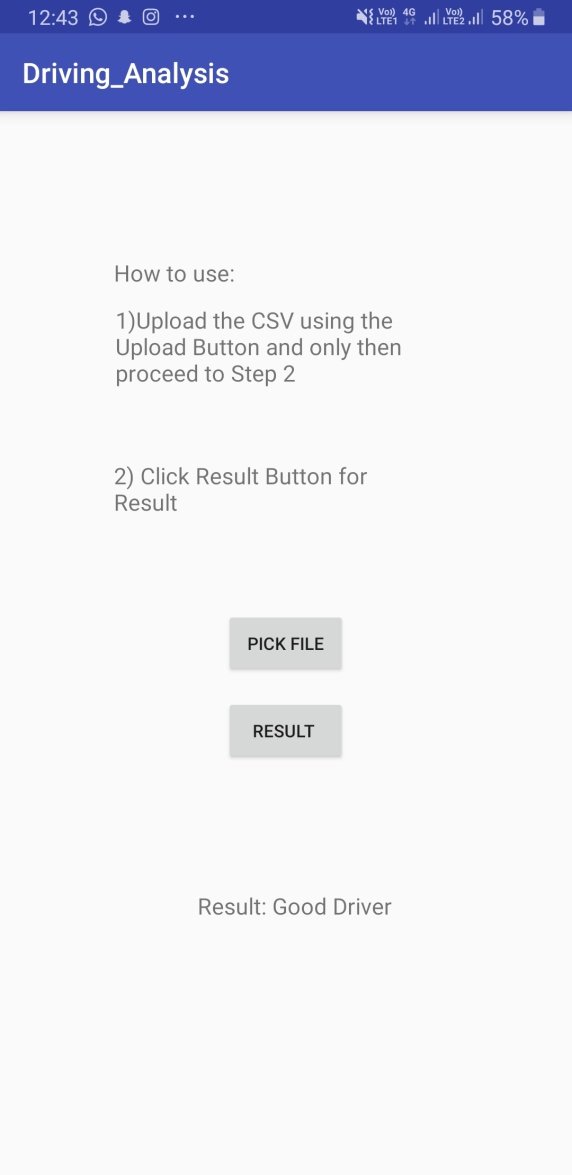
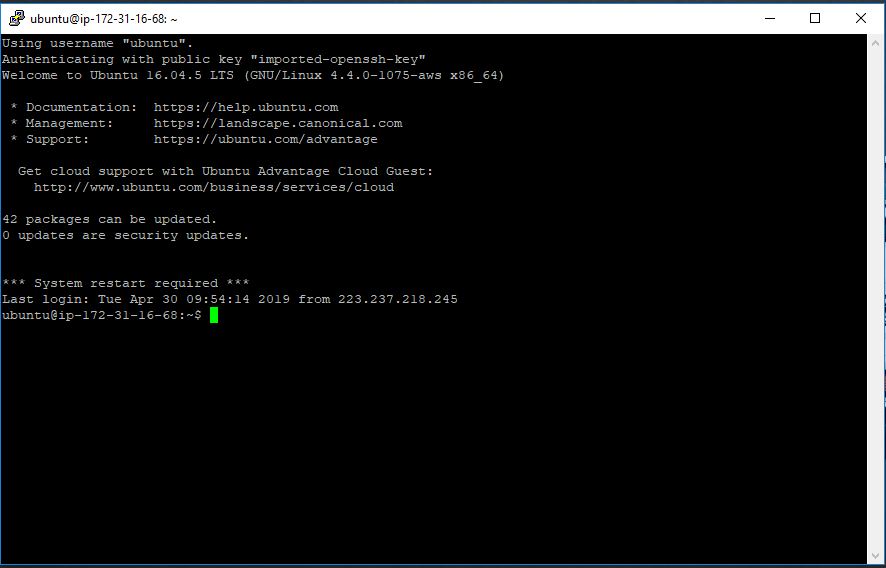
 

Fig 3.5-3.6 Progress Bar showing Upload, Result obtained after analysis

**3.5.3 Server**

The data collected by the Application is uploaded and processed on a Server. The server used is Ubuntu instance . The service is provided by AWS EC2, which hosts the server. A t2.micro instance is created and related dependencies are installed on it like Apache, PHP and Python. This enable for the building and execution of scripts and tools on locally.

The data received is stored directly on the file server, making it easier for the python script to readily read the data and evaluate it.

**

*Fig 3.7 Putty, where the server is run*

**3.5.4 Machine Learning Classification**

The processing of the input data is done in the step. The input data is read into the python script where machine learning algorithms are invoked on it. A supervised machine learning algorithm-Support Vector Machine is used for this purpose. Supervised methods are used, as the dataset considered for training is labelled. The dataset consists of over 6000 rows and 50 features (which are reduced to remove invariant attributes).

In the Data Collection phase the collection is done individually for the two Classes we wish to classify into and target values are added accordingly. This makes it easier for the algorithm to find similarity in data and classify it efficiently. Unsupervised methods may lead to erroneous results due to a lack of a large amount of labelled data. SVC is used as the preferred model for Classification as it provides a high degree of accuracy and works well for a large dataset.

**3.6 System Performance**

In the back end the SVC algorithm has an efficiency of O(nSVd).

In the front end the time required to connect to the OBD using Bluetooth varies across mobile platforms but is completed in avg time of about 1.0 sec. The data is sent from server to Application comparatively faster (~0.5 sec).

**3.7 Data Collection and Management**

The OBD device is inserted into the port generally found beneath the dashboard of the car. After starting the car, the connection is made to the OBD via Bluetooth and the required settings are set for the necessary parameters. The Data Collection phase can then start and capture of data takes place. In this phase data collection is done by driving at slow speeds at a safe and steady manner and this data is saved as the safe-driving dataset. In a similar way collection is done by simulating rash driving by driving at high speeds carelessly and on uneven roads. This enables us to label our dataset as ‘rash’-1 or ‘safe’-0 in the ‘Target’ column. Data Collection is complete when the car is stopped and the engine is turned off. The received data is then uploaded to the server.

**Chapter 4**

# Modeling and Implementation

**4.1 Overview**

This chapter covers the following subjects:

* The Languages, Tools used to Model the System
* Class Diagram
* Sequence Diagram
* Implementation of the Modules

**4.2 Languages and Tools Used**

The following are the various Tools and Technologies used to model and develop the system comprising of the front-end and back-end services.

* Mobile Application

1. Language- JAVA
2. IDE- Android Studio

* Data Analysis

1. Language- Python
2. IDE- Jupyter Notebook

* Data Storage

1. Language- PHP
2. Tool- phpmyadmin

* Server

1. Tool- Putty (running Ubuntu)
2. Technology- AWS EC2

## 4.3 Use Case Diagram

## usecase.JPG

Fig 4.1 Use Case Diagram depicting various actors and components

As shown in Fig 4.1, the User (Actor) is central to the overall system and interacts with various components to perform different functions and achieve certain specific objectives. The User interacts with the Torque Application to configure the OBD and to collect data via Bluetooth.

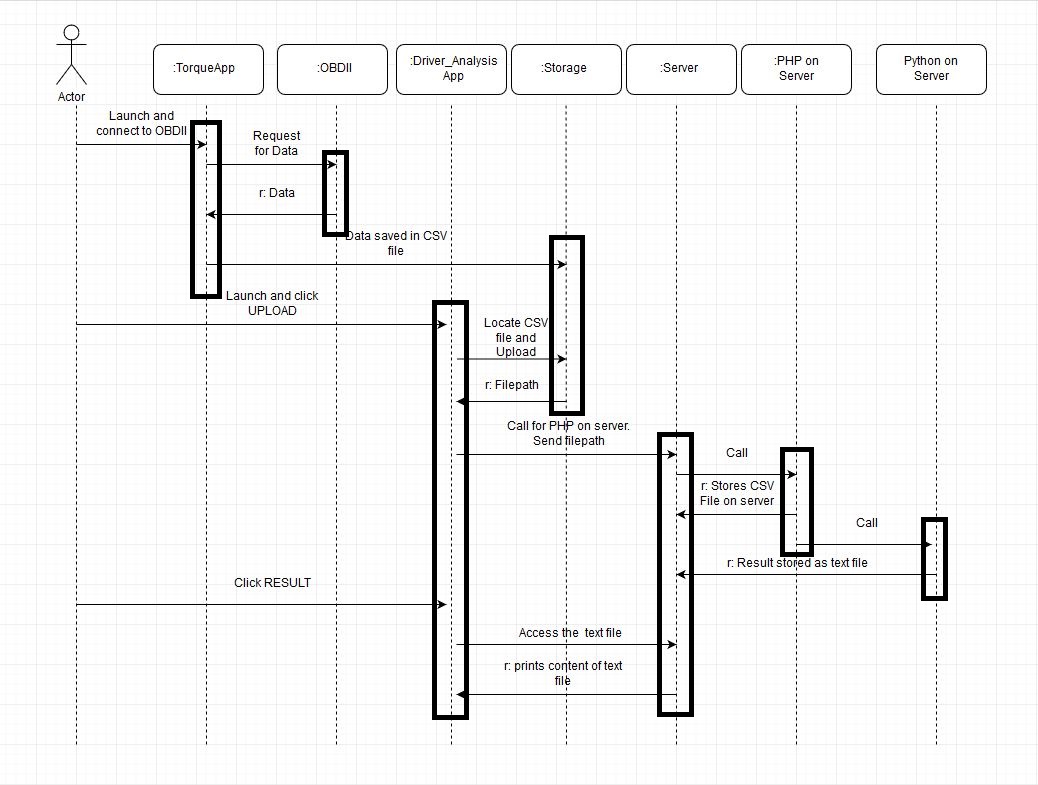
The User interacts with the Driver\_Analysis App to upload and display the final result.

Although there are more components in the system the User does not interact with them directly as they act as Back-end features running on the Server.

The Server is responsible for proving a link between the User and the final result.

Therefore the Server performs certain important functions such as running and executing PHP, Python scripts to process data develop a connectivity between Server and the Application.

## 4.4 Sequence Diagram

****

*Fig 4.2 Sequence Diagram*

The objects involved in the system are as follows:

* Torque App
* OBD2
* Driver\_Analysis App
* Storage
* Server
* PHP on Server
* Python script on the Server

The Actor involved in the system is the end user who executes the basic steps of uploading the data and receiving and viewing the output. The sequence diagram given above shows the flow of operation between various components across a time series.

The User must first launch and connect to the OBD using the Torque Application that is used in collecting data. This is the Start State. The App sends a request for data to the OBD2 device and receives the data from the device as a response.

The data received is saved as a csv file in the local device storage.

The User then interacts with the Driver\_Analysis App. The file to upload to the server is chosen and uploaded using the Upload button that executes the underlying code to upload to the server. The file needs to be received by the server and a PHP script is run on the server to this end.

The uploaded file is now stored on the server.

A PHP script on the server calls the python script containing the machine learning algorithm.

Once the python script is executed the result obtained is saved as a text file on the server itself.

The User can then interact with the Driver\_Analysis App once again to read the text file on the server and print and display the contents on the screen. This is the End State and completion of working of the System.

**4.5 Implementation**

This section covers the detailed implementation of the system (including the code) as a whole starting from Data Collection to displaying the final result. This consists of:

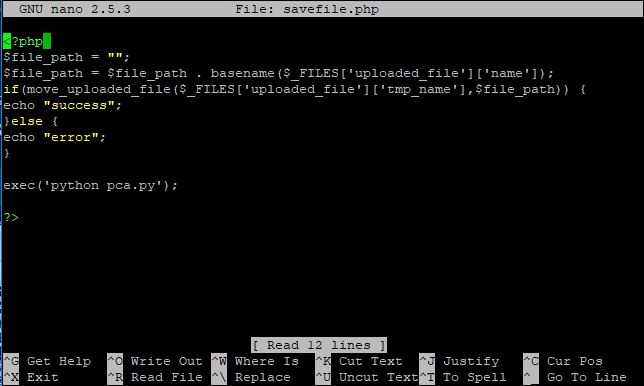
* Data Collection
* Storage
* Analysis
* Displaying the end result

**4.5.1 Collection**

The data is collected from the OBD2 device and is sent to the Android application. The data is saved in the form of a csv file containing the attributes and data needed to train and test our ML model. The dataset formed is imbalanced as it is practically difficult to simulate rash driving situations for a prolonged period of time.

**4.5.2 Storage**

The Storage of data happens at 2 places. The data collected is first saved locally on the mobile device. This data is then sent to the server via the Application. The dataset is stored here permanently and is used as input to test the SVC Algorithm. The dataset is stored on the Ubuntu AWS EC2 Server provided by Amazon. The file is uploaded by using JDBC (to connect to the database server) on the front-end (Android) and PHP at the back-end (Server) to receive the file.



*Fig 4.3 PHP Script for Receiving File and executing Python script*

**4.5.3 Data Analysis- Classification using ML Algorithms**

The objective is to determine if the driving performed by the end user is ‘Safe’ or ‘Rash’. To achieve this objective certain operations are performed on the data before classification (using Support Vector Classification algorithm) can take place.

**4.5.3.1 Data Pre-processing**

To prepare the data it must first be made suitable for use by the algorithm. This consists of extracting the required data from the dataset and then performing some pre-processing operations on them. This is an important step in Machine Learning as it greatly improves the accuracy and the prediction rate of the algorithm. Pre-processing may involve cleaning the dataset, filling up empty attributes in the dataset and setting some default values.

This if followed by Normalization (Standard Scalar method). Many machine learning methods are more effective if the data attributes have the same scale. The dataset is normalized before usage to rescale one or more attributes to the range of 0 to 1.

The Standard Score of a sample x is calculated as:

z = ( x - u )/ s

where,

u = Mean of the training samples,

s = Standard deviation of the training samples,

**

Fig 4.4 Normalization of Data

**4.5.3.2 Supervised Model using Support Vector Classifier**

Supervised Learning is a branch of Machine Learning which makes use of labels/targets in the dataset to determine the output value. Since the algorithm has known labels it uses them to ‘learn’ and predict the output of a test set. 90% of the dataset is used for training purposes and the reaming is used initially for testing.

The supervised learning algorithm used is Support Vector Machine (SVM) and the classification using this is known as Support Vector Classifier (SVC) in python’s sklearn library. SVC is a discriminative classifier formally defined by a separating hyperplane. The hyperplane classifies the values in the dataset. They can also be used for multi-class classification and regression analysis. SVC module is invoked from python’s sklearn library and the necessary parameters are set.

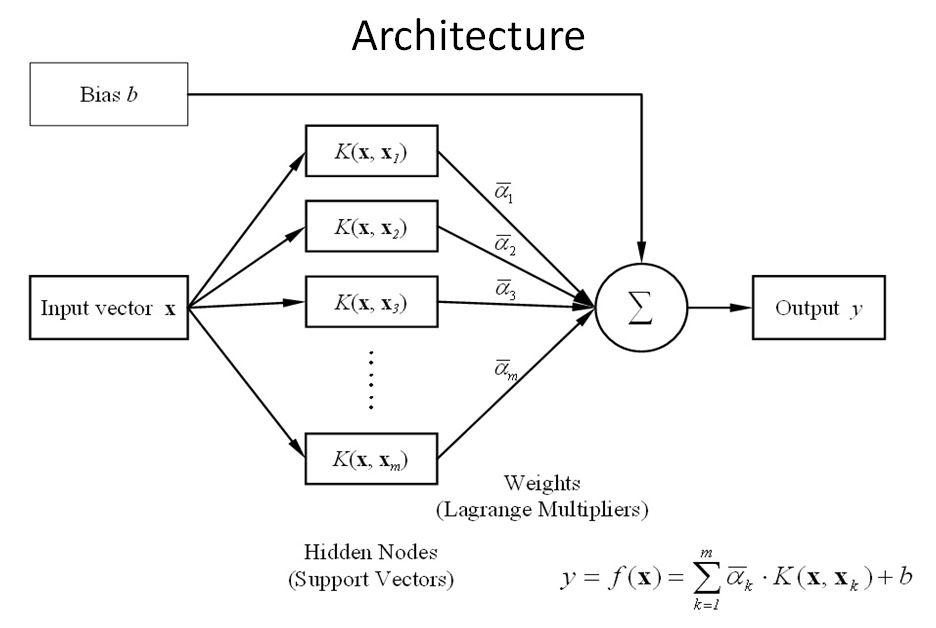
**

Fig 4.5 Architecture of SVM Algorithm

The kernel is used for dimensionality transformation in order to learn the hyperplane. This transformation is done by internally by the kernel using some liner algebra. The ‘sigmoid’ kernel is used by the SVC for transformation.

Gamma parameter determines how far the influence of a training sample reaches. Low values mean ‘far’ and high values mean ‘close’. Therefore by having a high value of gamma only points close to the estimated plane are considered.

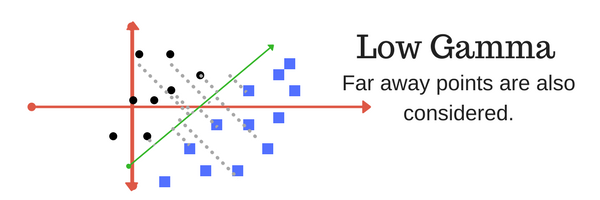
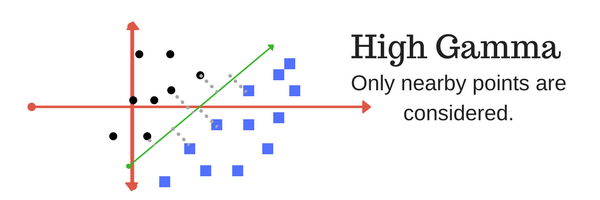
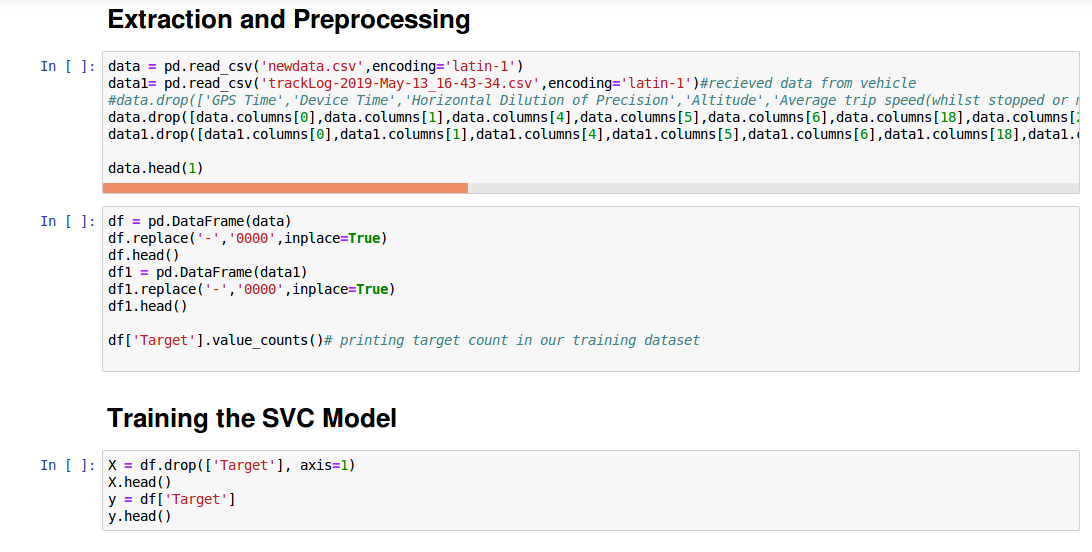


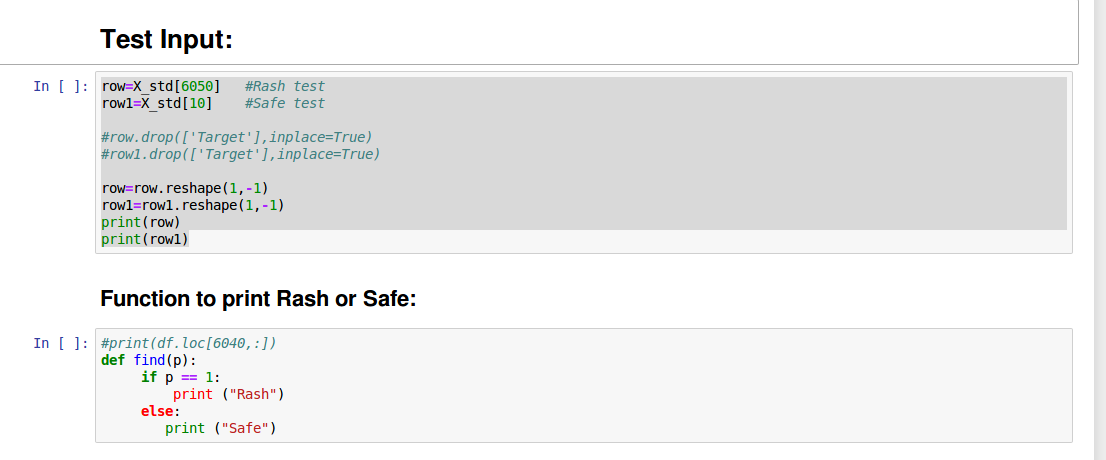
Fig 4.6 Visual Representation of Gamma value

The SVC Classifies into ‘Safe’ and ‘Rash’, with ‘Safe’ having a representation of – 0 and’ Rash’ – 1 in the Target column. Thus the output of 1 received would indicate the Class of Rash Driving.

A threshold is set to get the final prediction from the predicted list (consisting of 0’s and 1’s). Based on this threshold the final output of Rash or Safe is returned by a user defined function.

**

*Fig 4.7 Extraction and Processing of Data from the dataset*

**

*Fig 4.8 Testing with sample data and Function to print the Class*

The dataset is initially split to train and test the algorithm. The accuracy is then calculated using the score() method present in the sklearn library. This accuracy is the sum of the True Positives & True Negatives upon the sum of all values of the Confusion Matrix.

The algorithm is now trained with our training data and can be tested with the input data sent by the user. The predicted list is used to get a value of ‘Safe’ or ‘Rash’ and this is written to a ‘results’ text file on the server. This is later read and returned to the user.

**4.5.4 Result**

The end result is returned to the user on the Android Application. The output from the Analysis step is saved in a text file on the server. This file is read using a PHP script and the data contained in the file is sent to the Application. This output is displayed on a TextField on the Application and the user is made aware of the nature of the driving activity performed.

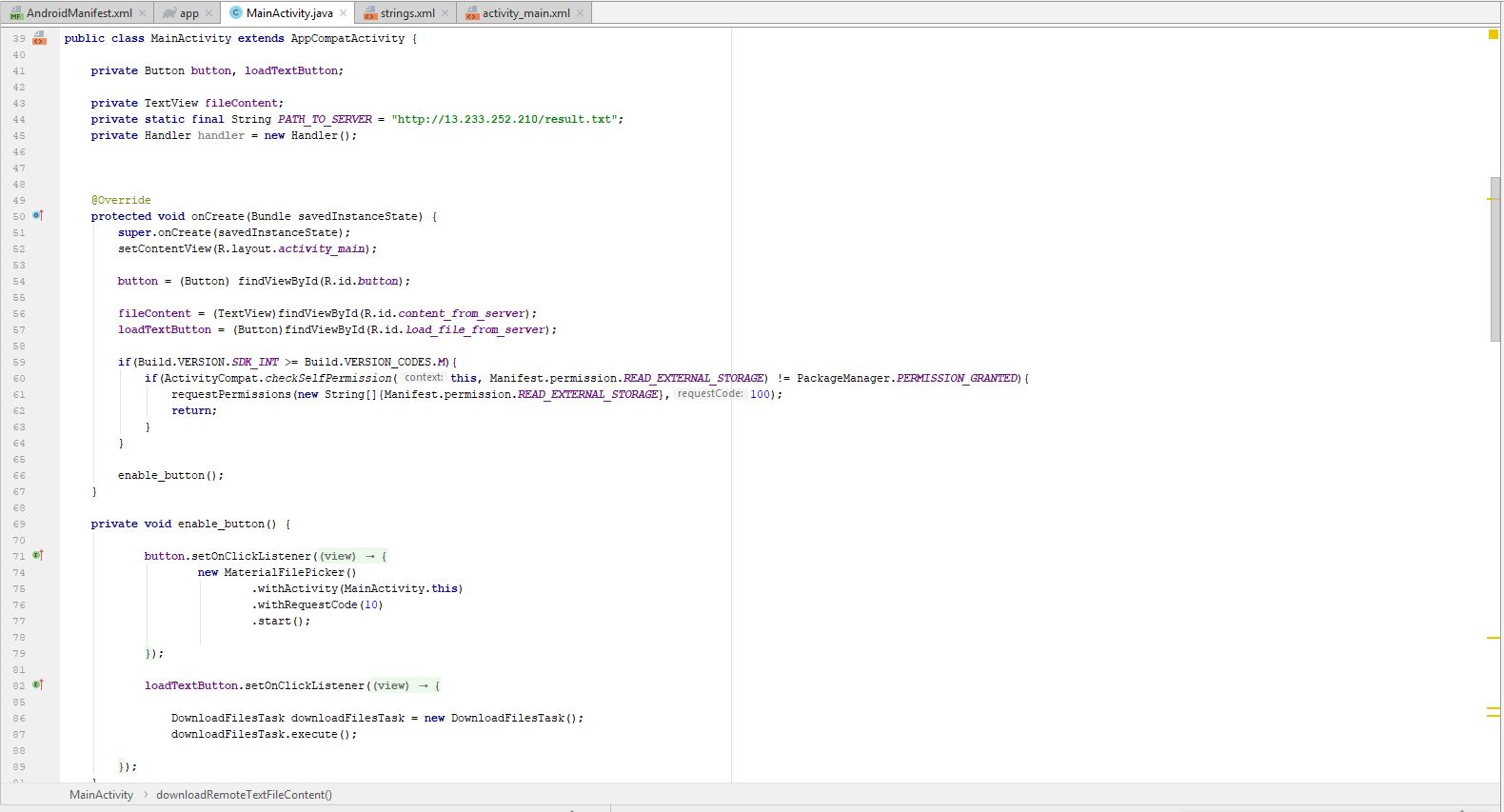


Fig 4.9 Android Application code structure

**Chapter 5**

# Testing, Results and Discussion

## 5.1 Testing

**5.1.1 Android Application**

The Application is fairly easy to use and performs without glitches. The Application has a connectivity requirement first to the Bluetooth enabled OBD and then to the Server via the internet. Therefore there maybe some delay in data transmission subject to connectivity. The Application is central to the system as it sends and receives data to display the computational output.

**5.1.2 OBD**

Some of the features obtained by the OBD2 device are either not of use (in Classification) or do not have any values and are empty as some cars do not have the inbuilt capability to read same feature data. Such features are omitted and/or transformed to default values in the pre-processing step prior to Classification. To do so we plotted a correlation graph against all features, this gave a better insight on the dependency of the features for our chosen Model. Selecting the features with high correlation values contributed to better accuracy in predicting the driving analysis.

**5.1.3 Server**

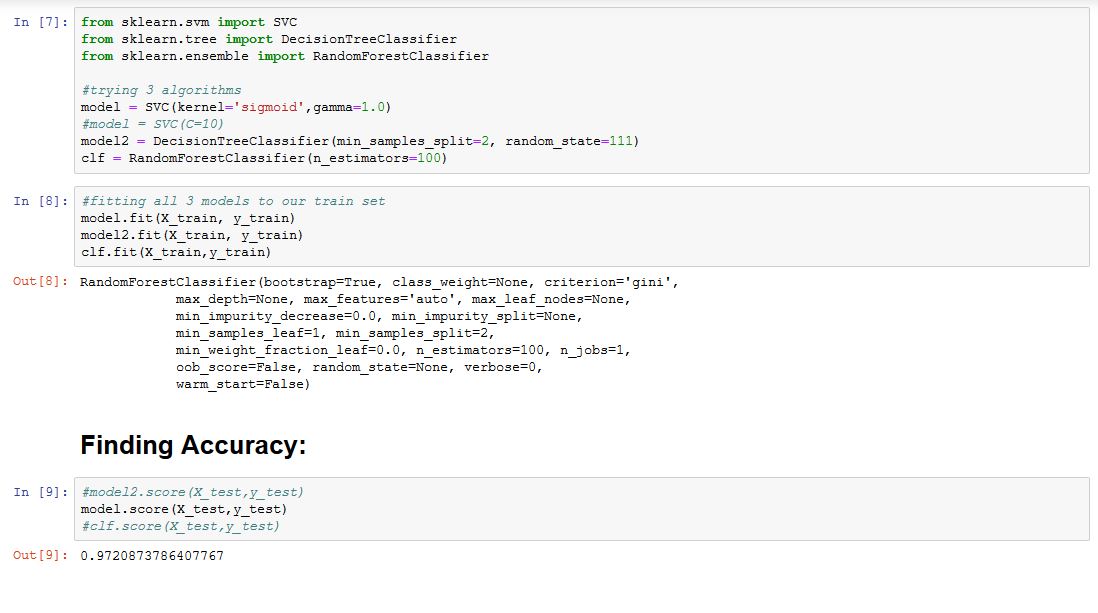
Ubuntu Server hosted by AWS EC2 works well as minimal amount of data and processing is required but a constant connection between the computer system and the remote terminal is needed or else it may cause the server to crash. Overuse of the server resources over prolonged periods may cause it to freeze. The data transfer speeds are fast but for large files there is a greater dependency on the data transfer speeds of the network.

**5.1.4 Classification Algorithm**

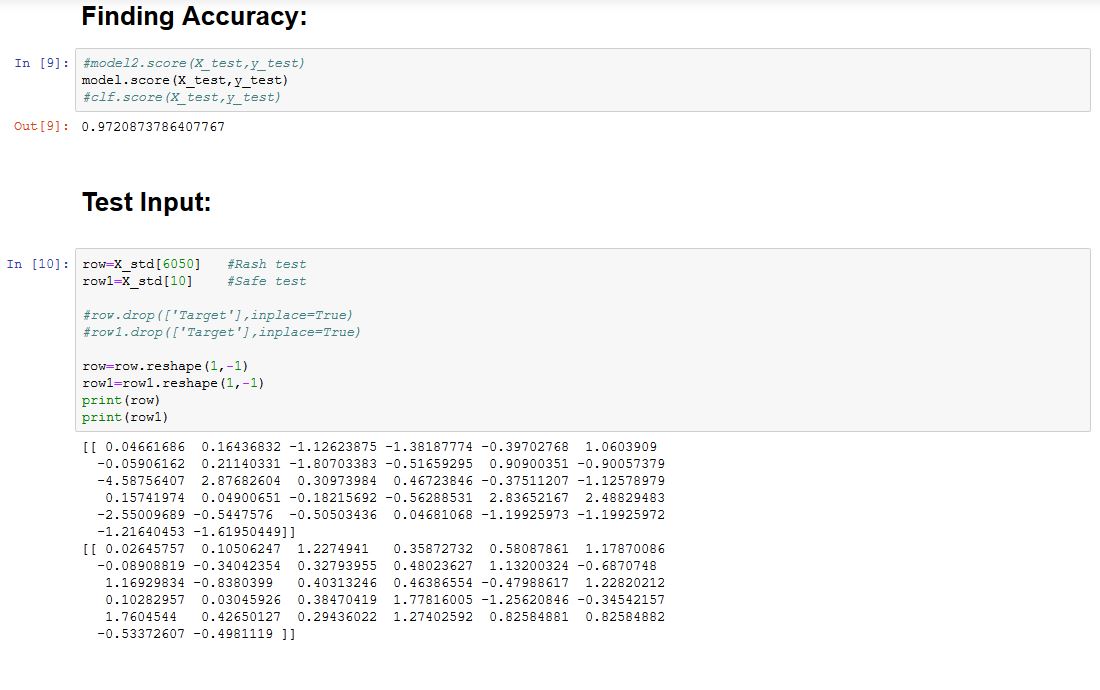
The SVC Algorithm was chosen as it is a powerful Supervised Learning algorithm and provides a high degree of accuracy. This was chosen as the primary model over others such as Decision Trees or Naïve Bayes (cannot contain negative values) as it provides the most optimal output by transformation (using kernel trick) of data.

## 5.2 Results

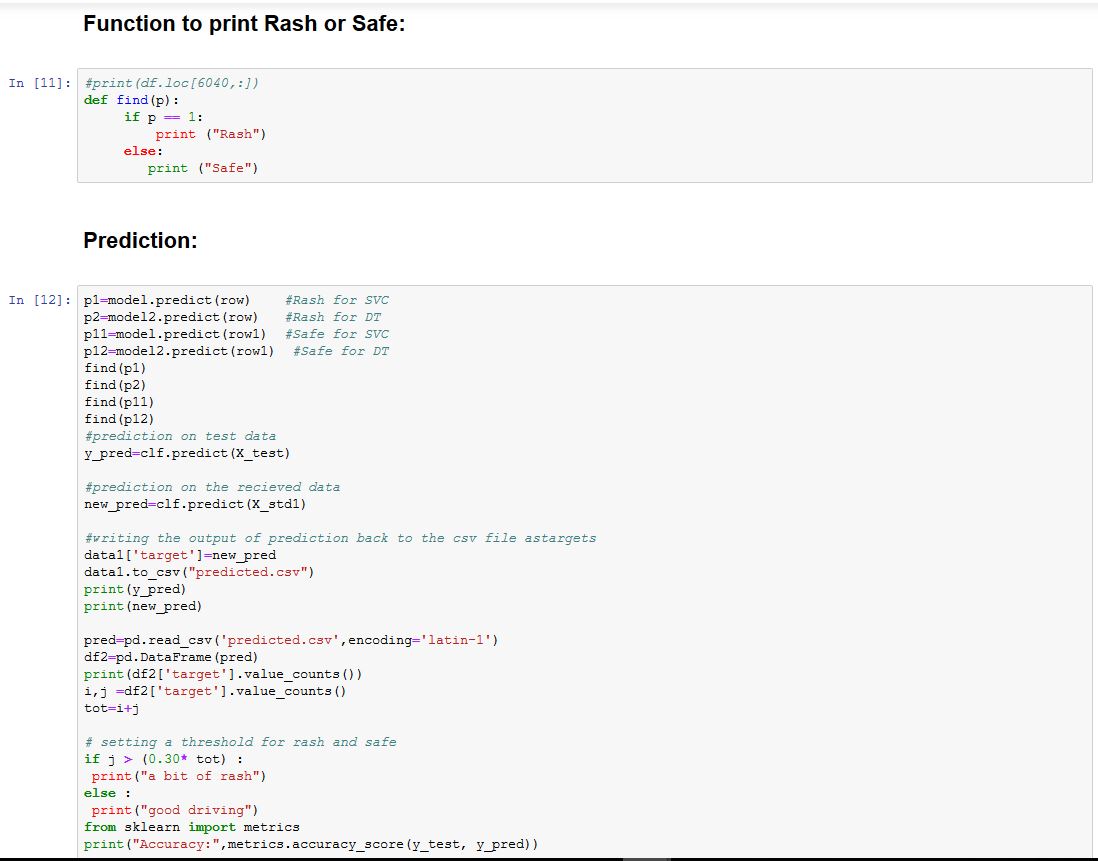
SVC Supervised Learning is applied to the dataset and the following results are obtained upon execution.

**

*Fig 5.1 SVC model declaration and Accuracy computation*

**

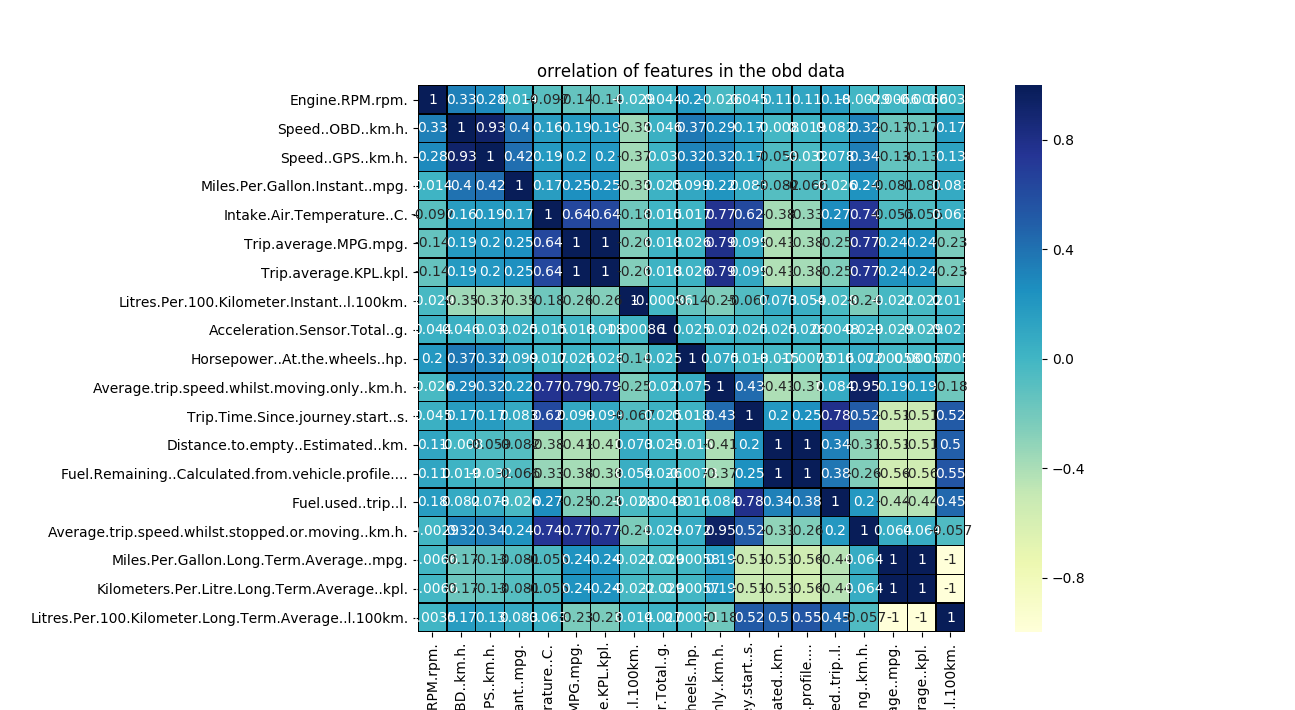
*Fig 5.2 Test Input after reshaping*

**

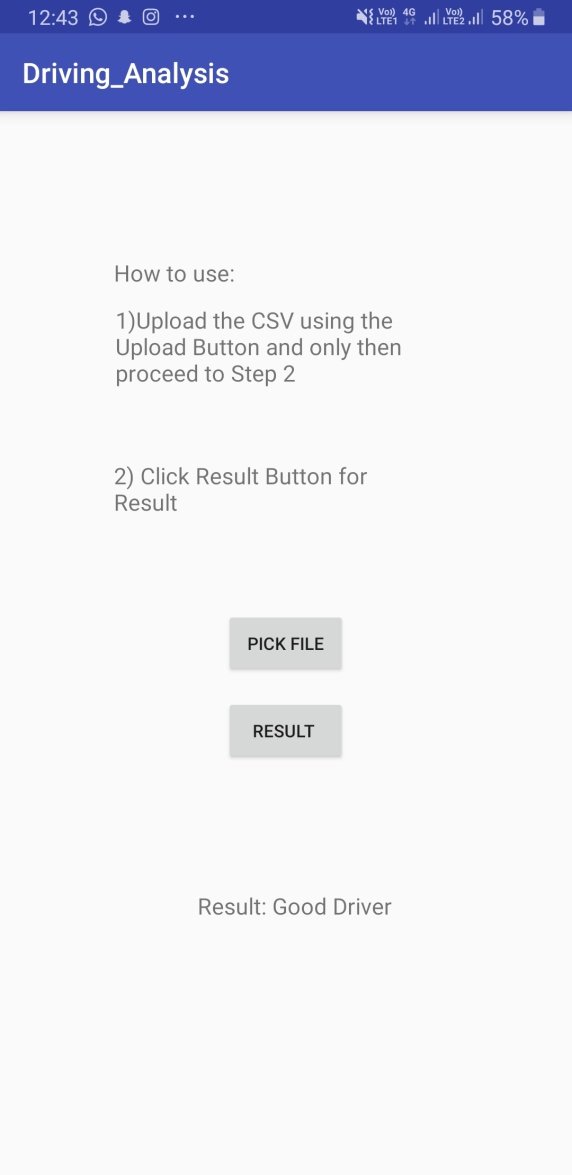
*Fig 5.3 prediction of test input*

**

*Fig 5.4 Final result obtained along with the predicted list*

**

*Fig 5.5 Correlation Plot*

**

*Fig 5.6 Android Application showing the Final Result*

## 5.3 Discussion

The SVC algorithm used classifies very accurately as a large amount of data is provided to it for learning. The drawback found is that it may tend to overfit the data or wrongly classify it as the model is not sensitive to sudden changes in driving conditions from Rash to Safe or vice-versa. This gives the user an overall idea/perspective of the quality of driving but can never be 100% accurate. Random Forest algorithm is also considered as SVM is sensitive to noise in the target classes and training of SVM takes a considerably longer amount of time.

**Chapter 6**

# Conclusion and Future Work

**6.1 Conclusion**

The confluence of IoT and Machine Learning is still being explored in the world of Information Technology. Research is being conducted into the possible ways to reduce accidents caused due to human faults and how to rectify them by providing a smart solution, thus making life easier for each individual and the society as a whole. By performing simple data analysis on data retrieved we can predict the quality of a driver with nothing more than past information. The application of the combination of hardware and intelligent software are numerous and can be applied to almost any field of work and is guaranteed to produce results. This provides a gateway to a variety of future applications not just in the automotive industry but also numerous others.

**6.2 Future Work**

There is scope for improvement in the Software Development section. The application is only supported on Android platform; this can be extended to other platforms as well. The UI of the Application can be upgraded and scaled to add support for more functionality.

The addition of a database can be used to add a login for each user and past history of tests conducted can be stored and accessed as when required.

Similarly, the nature of the system can be extended to support various other detection and prediction systems like potholes, speed bumps, polluted and highly congested areas. Only the methodology needs change as the data is already available.

A notification mechanism in the application to notify the user about an event that may occur. To provide accurate analysis other features may be considered that are not returned by the present sensor such as weather, peak hour traffic, road construction work etc.

**Bibliography**

[1] Driving Behaviour Analysis Based on Vehicle OBD Information and AdaBoost Algorithms. Shi-Huang Chen, Jeng Shyang Pan, and Kaixuan Lu. Proceedings of the International MultiConference of Engineers and Computer Scientists 2015 Vol 1, IMECS 2015, March 18 – 20, 2015, Hong Kong

[2] Using OBD-II Data to Explore Driving Behaviour Model. Yu-Jen Pan, Te-Cheng Yu and Rung-Shiang Cheng. Proceedings of the 2017 IEEE International Conference on Applied System Innovation IEEE-ICASI 2017 – Meen, Prior & Lam (Eds)

[3] Combining OBD Technology with Acceleration Sensor to Analyse Aggressive Behaviour. Shu-LinHwang and Yen Jen Chen. American Journal of Engineering Research (AJER), E-ISSN:2320-0847 P-ISSN:2320-0936, Volume 7, Issue 12, PP-139-144

[4] <http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>

[5] <https://community.data.gov.in/registered-motor-vehicles-in-india-as-on-31-03-2015/>