./

Learning Report – Scalable Machine learning and analytics of vehicle data to derive vehicle health and driving characteristics



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ver. Rel. No.** | **Release Date** | **Prepared. By** | **Reviewed By** | **To be approved By** | **Remarks/Revision Details** |
| 1 |  | Name/PS No | Name/PS No | Module Owner Name | Comments |
| 2 |  | 99003989 | Shreyash Sabde |  |  |
| 3 |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Document History**

# Table of Contents

[Table of Contents 3](#_Toc69306958)

[Table of Figures 4](#_Toc69306959)

[PROJECT TITLE 5](#_Toc69306960)

[PROJECT DETAILS 5](#_Toc69306961)

[AIM 5](#_Toc69306962)

[Problem Statement 5](#_Toc69306963)

[INTRODUCTION 5](#_Toc69306964)

[Overview 8](#_Toc69306965)

[Data collection: - 9](#_Toc69306966)

[Data Processing 11](#_Toc69306967)

[Machine learning 11](#_Toc69306968)

[Types of Machine learning algorithms 11](#_Toc69306969)

[Common Machine learning algorithms 12](#_Toc69306970)

[Low Code Platform 16](#_Toc69306971)

[Model Evaluation 17](#_Toc69306972)

[Regression Model Evaluation 17](#_Toc69306973)

[Classification model evaluation 17](#_Toc69306974)

[Vehicle Health Characteristics 19](#_Toc69306975)

[Sub Features 19](#_Toc69306976)

[Requirements 19](#_Toc69306977)

[SWOT analysis 19](#_Toc69306978)

[Literature review 20](#_Toc69306979)

[Approach 24](#_Toc69306980)

[Driving Behavior Monitoring 26](#_Toc69306981)

[Sub Features: - 26](#_Toc69306982)

[Requirements: - 26](#_Toc69306983)

[SWOT analysis 26](#_Toc69306984)

[Literature review 27](#_Toc69306985)

[Approach 33](#_Toc69306986)

[Road anomalies 37](#_Toc69306987)

[Sub Features: - 37](#_Toc69306988)

[Requirements: 37](#_Toc69306989)

[SWOT analysis 37](#_Toc69306990)

[Literature review 38](#_Toc69306991)

[Approach 49](#_Toc69306992)

[References 52](#_Toc69306993)

[Appendix A 54](#_Toc69306994)

# Table of Figures

[Figure 1 Block diagram of the process 9](#_Toc69296244)

[Figure 2 OBD adapter and OBD II port 10](#_Toc69296245)

[Figure 3 mobile camera as photo detection camera 10](#_Toc69296246)

[Figure 14 VMMS Architecture 16](#_Toc69296247)

[Figure 15 Generic data driven FDD approach 18](#_Toc69296248)

[Figure 16 Vehicle health predication flowchart 20](#_Toc69296249)

[Figure 17 Coolant temp detection 21](#_Toc69296250)

[Figure 18 gas pedal angle signal and its difference quotient, mean, median, and standard deviation. 23](#_Toc69296251)

[Figure 19 Speed and acceleration plot over time 24](#_Toc69296252)

[Figure 20 Table showing the driving characteristics 25](#_Toc69296253)

[Figure 21 Principal Component Analysis on dataset 25](#_Toc69296254)

[Figure 22 Interpretation by principal components and by rotated components 26](#_Toc69296255)

[Figure 23 Processes of driver behavior analysis 26](#_Toc69296256)

[Figure 24 Trained decision tree based on kurtosis coefficients of velocity and acceleration 27](#_Toc69296257)

[Figure 25 flowchart of the proposed driving behavior analysis method 28](#_Toc69296258)

[Figure 26 Driving behavior flowchart 30](#_Toc69296259)

[Figure 27 Detecting over speeding behavior 31](#_Toc69296260)

[Figure 28 Classifying driving behavior 32](#_Toc69296261)

[Figure 29 Ultrasonic sensor and receive-transmit operation 34](#_Toc69296262)

[Figure 30 Flowchart for pothole detection 35](#_Toc69296263)

[Figure 31 Flow chart for speed breaker detection 35](#_Toc69296264)

[Figure 32 complete flowchart of detection 36](#_Toc69296265)

[Figure 33 Location of potholes plotted in map [9] 36](#_Toc69296266)

[Figure 34 System model of deepbus [10] 37](#_Toc69296267)

[Figure 35 Performance comparison of various machine learning algorithms [10] 38](#_Toc69296268)

[Figure 36 Flowchart of data acquisition 39](#_Toc69296269)

[Figure 37 Flowchart of low-computational data-based trigger. 40](#_Toc69296270)

[Figure 38 Flowchart of high-computational image-based trigger 40](#_Toc69296271)

[Figure 39 Pothole validation through accelerometer data 41](#_Toc69296272)

[Figure 40 Accelerator pedal position and vehicle speed through potholes 41](#_Toc69296273)

[Figure 41 Image processing system 42](#_Toc69296274)

[Figure 42 Pothole detection algorithms 44](#_Toc69296275)

[Figure 43 high level diagram 46](#_Toc69296276)

[Figure 44 Pothole and Speed breaker detection flowchart 47](#_Toc69296277)

[Figure 45 checking the road quality 47](#_Toc69296278)

# 

# PROJECT TITLE

* Scalable Machine learning and analytics of vehicle data to derive vehicle health and driving characteristics.
* Key words: - Machine learning, Vehicle data, IOT

# PROJECT DETAILS

## AIM

* The project aims at working on diagnostics stack access the various engine parameters and onboard diagnostic protocol. Using data to build a machine learning framework for vehicle and driver characteristics. Further the development towards a scalable method for multiple vehicles.

## Problem Statement

* Many vehicles have the facility to check the health of the engine. But all methods are based on one to one communication. i.e. the OBD scanner will be attached to the vehicle and one can fetch the data of the ECU.
* Utilizing mobile sensor information along with OBD data to build ML and analytics model
* Identifying features that help with maintenance, car manufacturer and driver
* Scaling the model to work with multiple vehicles

# 

# INTRODUCTION

In India, it is observed that more than 50% of the accidents occur due to bad road conditions. Road anomalies such as potholes and roughness also add up to traffic congestion in major cities. To solve such issues, it is important to monitor the condition of roads and vehicles regularly.

Vehicles having very complex structure need an effective maintenance strategy. Three types of maintenance strategies are being used in vehicle industry, predictive maintenance, corrective maintenance, and preventive maintenance. In predictive maintenance, current condition of system/vehicle is analyzed to predict what is probably going to fail.

The characterization of driving behavior is not only crucial for accident prevention, as most of car accidents are due to human mishandling, but it is also important for designing driving models, which are the core of algorithms that might make the future of self-driving cars possible. Also, we can provide image processing in vehicle that will detect road signs such as speed limits, pedestrian or animal cross boards, roadwork indications, overhead warning boards, direction boards, railway tracks, etc., need to be detected in taking a decision on changing the direction and speed of the moving vehicle. This will help in self driving vehicles.

We can sort these issues into two categories:

* Road anomalies
* Driver and Vehicle behavior

Road anomalies can include potholes, speed-breakers or just any other abnormality on roads.

An efficient method to detect such anomalies can be through an accelerometer. An accelerometer is a device used to measure acceleration forces in the x y and z directions. Most of the smartphones today have an accelerometer in-built. This makes our approach cost efficient as we do not require any additional hardware.

The idea is to measure acceleration in x y and z directions and recognize peaks to detect harsh movement of the vehicle. But how do we measure the acceleration forces? How do we visualize them?

Smartphones have applications that can be installed onto the devices that measure and display acceleration forces on a 3D plane. But is measuring acceleration enough? What can we do with that information?

Accelerometer data combined with driving style analysis or drive cycles can be of great benefit. OBD (On board diagnostics) is a simple port attachable device that records real time data from cars such as RPM, Fuel data, Pedal positions, location coordinates and much more. This data can be correlated with the accelerometer data to render amazing analytic models. The huge collection of data acquired can help us understand how the anomalies affect automobiles and how drivers react to such anomalies.

# Overview

1. We can perform predictive maintenance analysis of vehicle based on acceleration data we will get either through OBD2 sensor or mobile app. We will perform the FFT analysis then afterwards using ML algorithm we can classify whether the vibration data is normal or abnormal.
2. Pothole and speed breaker detection, by detecting them diver can be alarmed and he can reduce speed of vehicle. The same can be achieved for the autonomous driving, so the car will automatically slow down or change direction. By detecting potholes at any position we can mark that position and send information to municipality.
3. Road sign detection. By the help of image processing we can detect road signs such as speed limits, pedestrian or animal cross boards, roadwork indications, overhead warning boards, direction boards, railway tracks, etc., need to be detected in taking a decision on changing the direction and speed of the moving vehicle.
4. Driving behavior detection, with the help of speed, rpm, engine load, acceleration data, acceleration pedal position data we can detect driving behavior of driver. Now this can be used for giving driving license to person who is learning driving. So, after some driving period person’s driving behavior will be analyzed and license will be granted to that person.
5. Another driving behavior is to rank the driver to qualify him for bus driver/ school bus driver/ Ola, Uber driver.

General procedure we will follow is collect data from sensor. Store the data and apply our machine learning algorithm on it. Our machine algorithm will give the result we want, whether it is about vehicle health detection, pothole/speed breaker detection, driving behavior classification, etc.

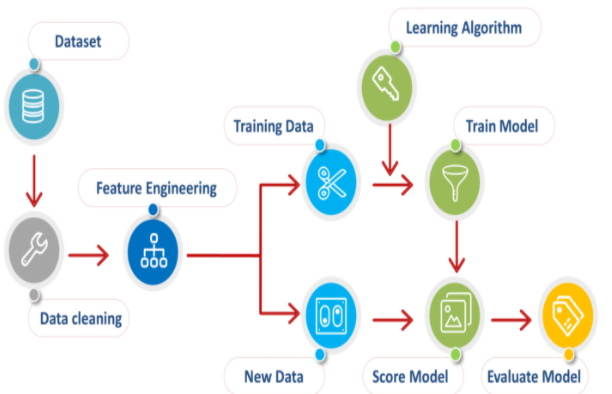


Figure 1 Block diagram of the process

## Data collection: -

The idea is to measure acceleration in x, y and z directions and recognize peaks in order to detect harsh movement of the vehicle. This can be measured with the help of in built app in phone and it will record acceleration data while driving. Accelerometer data combined with driving style analysis or drive cycles can be of great benefit.

OBD (On board diagnostics) is a simple port attachable device that records real time data from cars such as RPM, Fuel data, Pedal positions, location coordinates and much more. This data can be correlated with the accelerometer data to render amazing analytic models. The huge collection of data acquired can help us understand how the anomalies affect automobiles and how drivers react to such anomalies.



Figure 2 OBD adapter and OBD II port

Torque Pro is one of the most popular apps for reading the OBDII data.

It has both diagnosis capabilities (reading the error codes, system checkup, resetting error codes etc.) as well as real-time dashboard and graphing capabilities.

Connect to it with Bluetooth pairing and select list of sensors that we are going to need for our analysis.

For pothole/ speed breaker, road sign detection we can use mobile phone camera, no need to add extra setup to make our setup low cost.



Figure 3 mobile camera as photo detection camera

## Data Processing

Data processing means extracting data from dataset based on our application.

If we want to determine vehicle health characteristics then data related to such as acceleration in x, y and z, engine rpm, engine load. Similarly for the driving behaviour classification along with acceleration data we also want pedal position, location, etc.

Once our data is ready after processing, it is time to send to machine learning algorithm to perform next task.

## Machine learning

Machine Learning (ML) is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions. ML system is trained rather than explicitly programmed, basically it is presented with many examples relevant to a task, and it finds statistical structure that eventually allow the system to come up with rules for automating the task. To implement various ML models, we will make use of Scikit learn(sk-learn) library, which is a simple open source Machine Learning library, which provides efficient tools for data analysis, data pre-processing, model selection, model fitting, evaluation etc. It is built on Numpy, Scipy and Matplotlib libraries. We will also use some low code platforms like pycaret to create a framework for the application.

There are many machine learning algorithms to perform classification and regression analysis.

## Types of Machine learning algorithms

1. Supervised Machine Learning
2. Unsupervised Learning
3. Reinforcement Learning

Supervised Learning

This algorithm consists of a target / outcome variable (or dependent variable) which is to be predicted from a given set of predictors (independent variables). Using these set of variables, we generate a function that map inputs to desired outputs. The training process continues until the model achieves a desired level of accuracy on the training data. Examples of Supervised Learning: Regression, [Decision Tree](https://www.analyticsvidhya.com/blog/2015/01/decision-tree-simplified/), [Random Forest](https://www.analyticsvidhya.com/blog/2014/06/introduction-random-forest-simplified/), KNN, Logistic Regression etc.

Unsupervised Learning

In this algorithm, we do not have any target or outcome variable to predict / estimate. It is used for clustering population in different groups, which is widely used for segmenting customers in different groups for specific intervention. Examples of Unsupervised Learning: Apriori algorithm, K-means.

Reinforcement Learning

Using this algorithm, the machine is trained to make specific decisions. It works this way: the machine is exposed to an environment where it trains itself continually using trial and error. This machine learns from past experience and tries to capture the best possible knowledge to make accurate business decisions. Example of Reinforcement Learning: Markov Decision Process.

## Common Machine learning algorithms

1. Linear Regression: - It is used to estimate real values (cost of houses, number of calls, total sales etc.) based on continuous variable(s). Here, we establish relationship between independent and dependent variables by fitting a best line. This best fit line is known as regression line and represented by a linear equation Y= a \*X + b. Linear Regression is mainly of two types: Simple Linear Regression and Multiple Linear Regression. Simple Linear Regression is characterized by one independent variable. And, Multiple Linear Regression (as the name suggests) is characterized by multiple (more than 1) independent variables. While finding the best fit line, you can fit a polynomial or curvilinear regression. And these are known as polynomial or curvilinear regression.
2. Logistic Regression: - It is used to estimate discrete values (Binary values like 0/1, yes/no, true/false) based on given set of independent variable(s). In simple words, it predicts the probability of occurrence of an event by fitting data to a [logit function](https://en.wikipedia.org/wiki/Logistic_function). Hence, it is also known as **logit regression**. Since, it predicts the probability, its output values lies between 0 and 1 (as expected).
3. Decision Tree: -  It is a type of supervised learning algorithm that is mostly used for classification problems. It works for both categorical and continuous dependent variables. In this algorithm, we split the population into two or more homogeneous sets. This is done based on most significant attributes/ independent variables to make as distinct groups as possible.

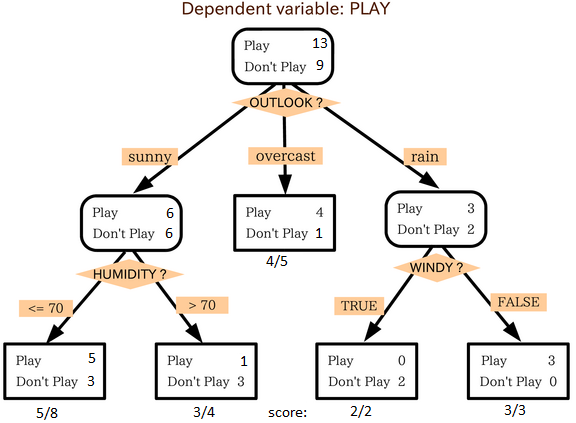


Figure 4 Decision Tree classification example

In the image above, you can see that population is classified into four different groups based on multiple attributes to identify ‘if they will play or not’.

1. SVM (Support Vector Machine): - It is a classification method. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. For example, if we only had two features like Height and Hair length of an individual, we’d first plot these two variables in two-dimensional space where each point has two co-ordinates (these co-ordinates are known as Support Vectors). Then, we will find some *line* that splits the data between the two differently classified groups of data. This will be the line such that the distances from the closest point in each of the two groups will be farthest away.

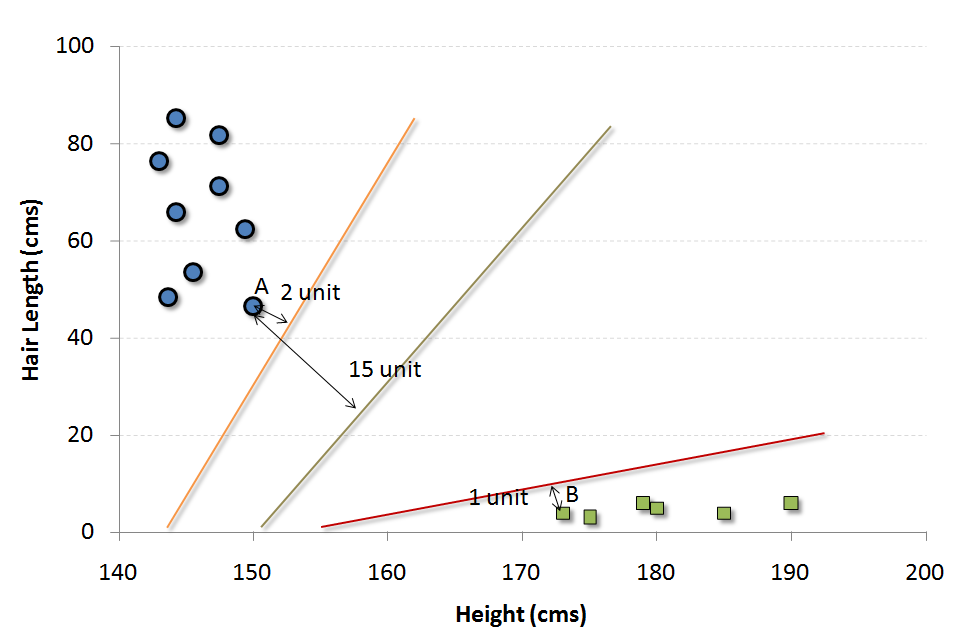


Figure 5 SVM classification example

1. Naïve Bayes: - It is a classification technique based on [Bayes’ theorem](https://en.wikipedia.org/wiki/Bayes%27_theorem) with an assumption of independence between predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. [Naive Bayesian](https://courses.analyticsvidhya.com/courses/naive-bayes?utm_source=blog&utm_medium=common-machine-learning-algorithms) model is easy to build and particularly useful for very large data sets.

Steps to follow for Naïve Bayes classification

1. Convert the data set to frequency table
2. Create Likelihood table by finding the probabilities
3. Use Naive Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction.
4. KNN (K – Nearest Neighbors): - It can be used for both classification and regression problems. However, it is more widely used in classification problems in the industry. K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases by a majority vote of its k neighbors. The case being assigned to the class is most common amongst its K nearest neighbors measured by a distance function.

These distance functions can be Euclidean, Manhattan, Minkowski are used for continuous function and Hamming distance for categorical variables.

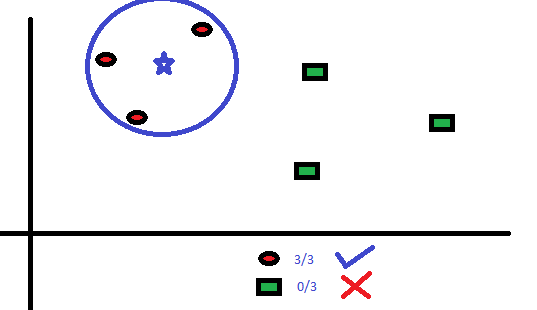


Figure 6 KNN classification example

1. K – Means: It is a type of unsupervised algorithm which solves the clustering problem. Its procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters). A cluster refers to a collection of data points aggregated together because certain similarities. Every data point is allocated to each of the clusters through reducing the in-cluster sum of squares. In other words, the K-means algorithm identifies *k* number of centroids, and then allocates every data point to the nearest cluster, while keeping the centroids as small as possible.

The *‘means’* in the K-means refers to averaging of the data; that is, finding the centroid. To process the learning data, the K-means algorithm in data mining starts with a first group of randomly selected centroids, which are used as the beginning points for every cluster, and then performs iterative (repetitive) calculations to optimize the positions of the centroids.

1. Random Forest: - With increase in computational power, we can now choose algorithms which perform very intensive calculations. One such algorithm is “Random Forest”. Random Forest is a trademark term for an ensemble of decision trees. In Random Forest, we’ve collection of decision trees (so known as “Forest”). To classify a new object based on attributes, each tree gives a classification and we say the tree “votes” for that class. The forest chooses the classification having the most votes (over all the trees in the forest).
2. Dimensionality Reduction algorithms: - Like clustering methods, dimensionality reduction seek and exploit the inherent structure in the data, but in this case in an unsupervised manner or order to summarize or describe data using less information. This can be useful to visualize dimensional data or to simplify data which can then be used in a supervised learning method. Many of these methods can be adapted for use in classification and regression.

Principal Component Analysis (PCA), Principal Component Regression (PCR), Linear Discriminant Analysis (LDA) are few examples of Dimensionality Reduction algorithms.

## Low Code Platform

PyCaret: - PyCaret is an open source, low-code machine learning library in Python that allows you to go from preparing your data to deploying your model within minutes in your choice of notebook environment.

Analyzing the performance of a trained machine learning model is very critical step in the machine learning workflow. With over 60 plots available in PyCaret, you can now evaluate and explain model performance and results instantaneously without the need to write complex code.

Whether its imputing missing values, transforming categorical data, feature engineering or even hyperparameter tuning of models, PyCaret automates all of it. It orchestrates the entire pipeline no matter how complex it is.

## Model Evaluation

### Regression Model Evaluation

1. R2 Score:  
   R^2 (coefficient of determination) regression score function. The best possible score can be 1 and it can go negative as well in worst conditions.
2. Mean squared error:  
   The mean squared error function computes mean square error, a risk metric corresponding to the expected value of the squared error or loss.
3. Mean absolute error:  
   The mean absolute error function computes mean absolute error, a risk metric corresponding to the expected value of the absolute error loss. Here is a code example which demonstrates the usage of various evaluation methods for regression models.

### Classification model evaluation

1. Confusion matrix: A Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model. This gives us a holistic view of how well our classification model is performing and what kinds of errors it is making.

For a binary classification problem, we would have a 2 x 2 matrix as shown below with 4 values:

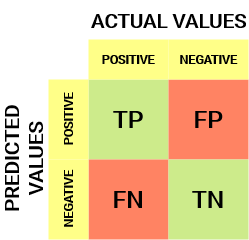


Figure 7 Confusion matrix

1. Accuracy score: - The accuracy score function computes the accuracy, either the fraction (default) or the count of correct predictions. If the entire set of predicted labels strictly match with the true set of labels, then the subset accuracy is 1.0; otherwise it is 0.
2. Precision Score: **-** The precision is the ratio tp / (tp + fp) where tp is the number of true positives and fp the number of false positives. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative.
3. Recall Score:- The recall is the ratio tp / (tp + fn) where tp is the number of true positives and fn the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples.
4. F1 Score:- The F1 score can be interpreted as a weighted average of the precision and recall, where F1 score reaches its best value at 1 and worst score at 0

# Vehicle Health Characteristics

## Sub Features

1. Coolant temperature monitoring
2. Engine oil temperature monitoring
3. Monitoring fuel economy

## Requirements

1. Hardware requirement to gather acceleration, engine load, Coolant temperature, Engine oil temperature (OBD II module) to monitor the vehicle health.
2. System should be able to upload to the data about vehicle health parameters to cloud, for future reference to the person to predict the maintenance time.
3. System should be able to tell about vehicle health characteristics in case of reckless driving.
4. System should be able to alert the driver about vehicle health.

## SWOT analysis

|  |  |
| --- | --- |
| STRENGTH   1. Vehicle health monitoring 2. Can alert the driver 3. Monitoring vehicle health | WEAKNESS |
| OPPURTUINITY   1. In case of car problem looking at data we can detect the source. 2. Car repairing or servicing can be properly schedule | THREATH   * 1. False alert can lead to unnecessary servicing or oil change. |

## Literature review

It is hard to diagnose failure in advance in the vehicle industry because of the limited availability of sensors and some of the designing exertions. It looks feasible today to analyze sensor’s data along with machine learning techniques for failure prediction. In this article, an approach is presented for fault prediction of four main subsystems of vehicle, fuel system, ignition system, exhaust system, and cooling system. Sensor is collected when vehicle is on the move, both in faulty condition (when any failure in specific system has occurred) and in normal condition. The data is transmitted to the server which analyzes the data. Interesting patterns are learned using four classifiers, Decision Tree, Support Vector Machine, K Nearest Neighbor, and Random Forest. These patterns are later used to detect future failures in other vehicles which show the similar behavior. Accuracy comparison of all classifiers is performed on the basis of Receiver Operating Characteristics (ROC) curves.

Vehicle systems are complex both in hardware and software so their maintenance is challenging and that too predictive maintenance is difficult in predictive maintenance, current condition of system/vehicle is analyzed to predict what is probably going to fail. OBD sensor is used to get the sensor data required for the analysis. On- Board Diagnostic (OBD) frameworks give the vehicle owner or a repair professional access to condition of current data.

Block diagram

An OBD2 scanner is connected with the vehicle through OBD2 port. This scanner behaves like a bridge between vehicle and portable device, that is mobile and laptop which supports Bluetooth. Data is continuously being generated and transmitted to smart phone via Bluetooth.

First step is feature selection in which data stream of DTC is filtered in feature selection process using expert’s suggestion. Then a PCA (Principle Component Analysis) is applied on data set for feature reduction. After that four classification algorithms are used in classification phase including Decision Tree, Random Forest, K-NN, and SVM. These results are stored on server for further derivation which is used for fault detection and remote monitoring of the vehicle.

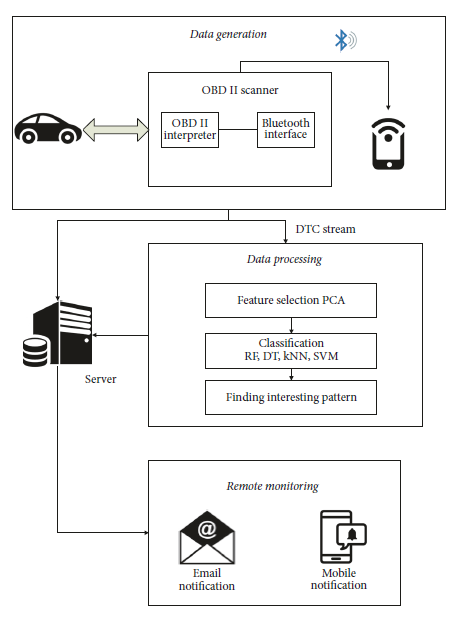


Figure 8 VMMS Architecture

The performance of each algorithm on particular system is evaluated on the basis of accuracy, precision, recall and F1 score.

Our first contribution to this paper is data generation and feature selection. We selected those distinctive sensors that can cause a system to break down. The second contribution is to propose a user friendly vehicle fault prediction and vehicle remote health monitoring system. [1]

The competitive businesses' desire to provide "smart services" and the pace at which the modern automobiles are increasing in complexity, are motivating the development of automated intelligent vehicle health management systems. Current On-Board Diagnosis (OBD 11) systems use simple rules and maps to perform diagnosis. On-Board Diagnostic (OBDII) systems perform a great job in assisting technicians to trouble-shoot faulty components in the vehicle.

Fault detection and diagnosis (FDD) has mainly evolved upon three major paradigms, viz., model- based, data-driven and knowledge-based approaches. A data driven approach can provide a more systematic solution if monitored sensor data is available over a period of time for analysis.

We propose a data mining approach that utilizes the knowledge from signal-processing, statistical and pattern recognition domains. In this paper, we show its applicability to FDD in a real automotive engine, and comment on its practical application. The engine is operated under various scenarios, and the data is collected for FDD analysis.

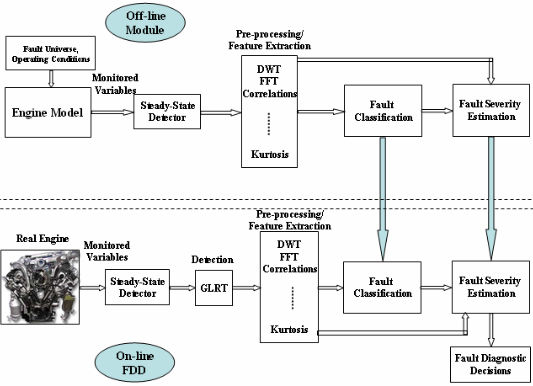


Figure 9 Generic data driven FDD approach

Above figure depicts our proposed approach for automatically detecting and diagnosing faults in automotive engines. It is a generic approach applicable to any engineering system, it requires only sensor data, and does not need detailed system models. The collected data is processed and analyzed to achieve good diagnostic accuracy.

In this paper, we proposed a systematic data- driven approach for performing fault detection and diagnosis in automotive engines and showed its applicability to Toyota Camry engine. Pre- and post - fault data is collected from the engine under various scenarios, and the proposed approach is applied to generate a D-matrix to uniquely isolate the faults into pre-defined fault classes [2]

Among the sensors which are present in modern computerized vehicles, Mass Air Flow (MAF) sensor and Oxygen (02) sensor are two of the most important sensors that crucially determine the engine performance, emissions, and other important functions. Apart from that engine coolant temperature, engine oil temperature, fuel economy are also important to monitor.

Coolant temperature, is queried from the OBD2adapter periodically and is forwarded to the CEP engine. If the averaged value is greater than the given temperature threshold (we used 104 °C), the driver is alerted immediately about the possible overheating.

If the engine oil temperature is not within the desired operating range when the average engine rpm is greater than a certain threshold. As the operating range and the rpm threshold differ from vehicle to vehicle so, here we will use the average rpm.

The fuel rate, along with the speed can be used to calculate the fuel economy.

Fuel Economy = (Avg. fuel consumption) / (Avg. speed)

If the average fuel economy becomes lower than a certain threshold, an alert is generated.

Battery voltage can be retrieved from the OBD2 adapter. If this goes below a certain threshold, the driver is alerted that there is a possible battery charging failure. [3]

## Approach

1. We will be collecting relevant data to monitor vehicle health through OBD II module.
2. For vehicle health monitoring, we need acceleration data, coolant temperature, engine oil temperature, fuel consumption data. So, out of whole dataset these required data will be extracted.
3. Then, each parameter will undergo algorithm that will check the parameter is under limit or not.
4. Coolant temperature data extracted from dataset will compare with averaged value, if it is greater than the given temperature threshold, then the driver is alerted immediately about the possible overheating. Same goes for engine oil temperature.
5. For fuel economy monitoring, fuel consumption and average speed data is extracted from dataset and fuel economy is calculated. If the average fuel economy becomes lower than a certain threshold, an alert is generated.

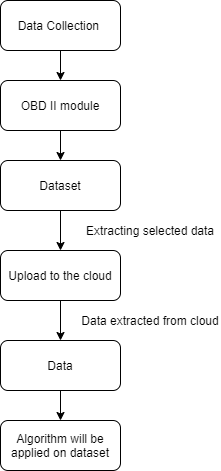


Figure 10 Vehicle health predication flowchart

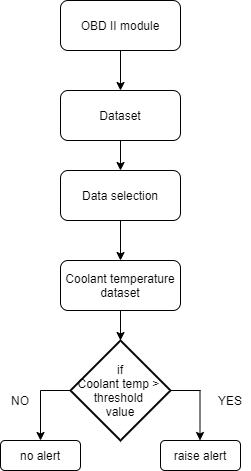


Figure 11 Coolant temp detection

Same flowchart is applicable for checking engine oil temperature, fuel economy.

# Driving Behavior Monitoring

## Sub Features: -

1. Reckless driving detection
2. Driving behavior classification
3. Effect of reckless driving on vehicle health

## Requirements: -

1. Hardware requirement to gather acceleration, engine load, engine rpm data (OBD II or mobile sensor) to classify the driver behavior is safe or not.
2. System should be able to upload to the data about driving behavior to cloud, for future reference to the person to check or improvement in driving habit.
3. System should be able to tell the effect of reckless or aggressive driving on vehicle health such as fuel consumption, vibration data, and engine load.

## SWOT analysis

|  |  |
| --- | --- |
| STRENGTH   1. Driving behavior monitoring 2. Count of over speeding can be recorded 3. Effect of reckless driving on vehicle health can be monitored. | WEAKNESS   1. For correct classifying driving behavior, large dataset is required, so immediate result will not be accurate. |
| OPPURTUINITY   1. An alternate system to driving test. 2. Insurance company can use data to give discount on policies. 3. Web interface for parents to monitor kid’s driving behavior. | THREATH |

## Literature review

The characterization of driving behavior is not only crucial for accident prevention, as most of car accidents are due to human mishandling, but it is also important for designing driving models, which are the core of algorithms that might make the future of self-driving cars possible.

Cars can nowadays record several thousands of signals through the CAN bus technology and potentially provide real-time information on the car. It can record the data gas pedal position, brake pedal pressure, steering wheel angle, steering wheel momentum, velocity, RPM, frontal and lateral acceleration. We propose an unsupervised learning technique that clusters drivers in different groups.

Data collection- For determining driving behavior data we can analyze is Brake pedal pressure (BRK), Gas pedal position (GAS), Revolutions per minute (R.P.M.), Speed (SPD), Steering wheel angle (S.W.A.), Steering wheel momentum (S.W.M.), Frontal acceleration (F. ACC.), Lateral acceleration (L. ACC.)

These signals are directly or, in some cases, indirectly related to the interaction between the driver and the vehicle.

Feature extraction: - From each considered signals we extract the following 7 indicators

Values of the signal for each sample, Difference quotient (discrete first derivative), Time interval between two singular points, Values of the local maxima, moving mean, moving median, moving standard deviation.

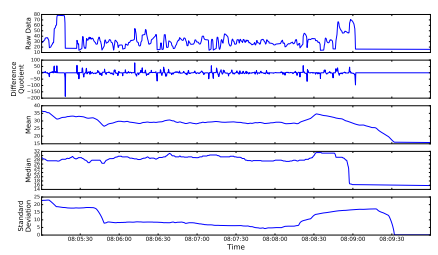


Figure 12 gas pedal angle signal and its difference quotient, mean, median, and standard deviation.

Once the full dataset is collected, we use the K-means clustering algorithm to leverage the features defined in the previous section with the aim of grouping drivers upon common similarities. In order to plot them on bi-dimensional space, therefore, a dimensionality reduction technique has to be performed. In this work we use Principal Component Analysis (PCA), a well-known statistical procedure that decreases the dimensionality of a space projecting it into another one whose dimensions (principal components) are orthogonal to each other and such that the variance of the projected data-points on the principal components is maximized. [4]

The purpose of such an endeavor is to classify the drivers according to their risk-proneness within the larger context of increasing traffic safety, which is a major concern worldwide. Cluster and principal component analyses from exploratory statistics have been used to identify and explain drivers grouping according to their driving behavior.

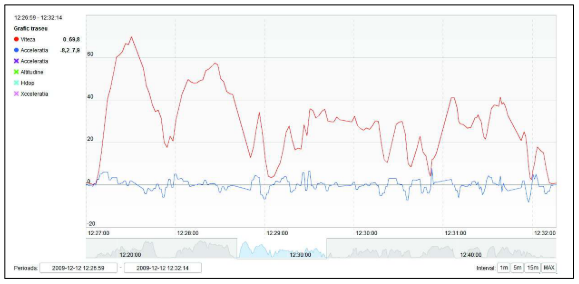


Figure 13 Speed and acceleration plot over time

For the statistical analysis, following driving parameters extracted from the raw data:

Speed over 60 km/h: percent of time (V60)

Speed: mean value (Vmn) and standard deviation (Vsd)

Acceleration: standard deviation (Asd) –

Positive acceleration: mean value (A+mn) and standard deviation (A+sd) –

Braking: mean value (Brmn) and standard deviation (Brsd)

Mechanical work (w)

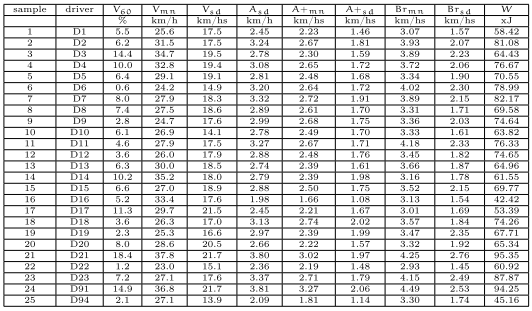


Figure 14 Table showing the driving characteristics

Principal Component Analysis is a statistical method for arranging large arrays of data into interpretable patterning match. It transforms a number of (possibly) correlated variables into a (smaller) number of uncorrelated variables called principal components.

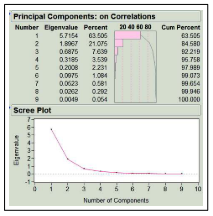


Figure 15 Principal Component Analysis on dataset

Data Interpretation: - Based on the analysis of the principal components and of their values and correlations to the clusters, we can derive an interpretation.

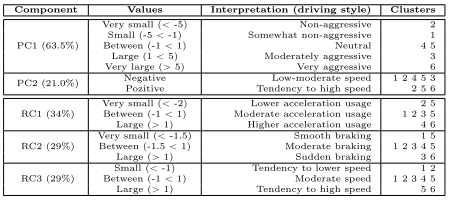


Figure 16 Interpretation by principal components and by rotated components

[5]

The driving habit index evaluation of a driver must be followed by a long term period. The long term accumulated driving information can effectively summarize the specific driver behavior by statistical analysis. Python has been adopted to as the main development tool accompanying with the Scikit-learn packages. Decision tree classification technique was applied to generate the analyzing knowledge for driver behavior analysis.

The entire analysis process is divided into iterative processes of data clean, data integration, data selection, data transformation, data mining, as well as knowledge evaluation/display. Vehicle speed and acceleration of each logged route are used as input parameters to calculate the statistical kurtosis and skewness, then decision tree classification technique was applied to generate the analyzing knowledge for driver behavior analysis.

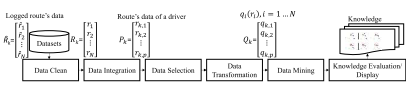


Figure 17 Processes of driver behavior analysis

The data clean process retrieves a logged route data from cloud computing platform as input using web API with an authorized user key and route key. It returns a JSON string list. Kalman filter has been implemented in data clean process to filter the noises caused by data sampling and communication. Each row

The purpose of data selection process is to rank the most useful parameters from Pk. It intended to lower the dimension of tuple to reduce the complexity of the data mining process. It ranked velocity, engine RPM, and fuel consumption rate that were the three most useful parameters in order. Once the data is ready to process with, we evaluate the skewness and kurtosis. The decision tree classifier in scikit-learn package has been adopted to realize the function of data mining process based on the kurtosis and skewness of velocity as well as acceleration.

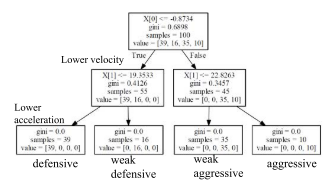


Figure 18 Trained decision tree based on kurtosis coefficients of velocity and acceleration

The left subtree from root represents lower velocity. The left most leaf represents the lower velocity and acceleration which is labeled defensive. On the other hand, the right most leaf represents the higher velocity and acceleration which is labeled aggressive. [6]

Another method to classify driver behavior is by using Adaboost algorithm.

The proposed driving behavior analysis method utilizes OBD interface to collect a number of critical driving operation data, i.e., vehicle speed, engine speed (RPM), throttle position, and calculated engine load.

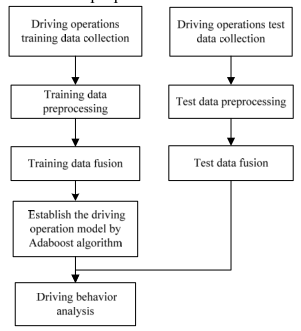


Figure 19 flowchart of the proposed driving behavior analysis method

This proposed method collects the vehicle operation information, including vehicle speed, engine RPM, throttle position, and calculated engine load and then makes use of AdaBoost algorithms to create a driving behavior classification model.

This paper uses three characteristics, i.e., the relative ratio of the vehicle speed and engine speed, the relative ratio of throttle position and engine speed, and the engine load.

The number of iterations of AdaBoost algorithm is obtained by the three different ways. They are Gentle AdaBoost, Modest AdaBoost, and Real AdaBoost. The results of Real AdaBoost are better than those of Modest AdaBoost and Gentle AdaBoost. [7]

## Approach

1. For driving behavior monitoring and classification, we need that can help us to classify driving behavior. We require data like acceleration value, engine speed, engine load, vehicle speed, throttle position. All these require data we can get from OBD II module.
2. So, to detect if driver is over speeding or not we use acceleration data. If acceleration value is more than threshold value that we set, then that will set value “1”, otherwise “0” will be set. In interval of time, if count of “1” is more, then we can say that the driver is over speeding.
3. By detecting the over speeding frequency or history, we can predict the driving behavior of driver.
4. To classify the driving behavior aggressive or not aggressive. We take the data acceleration value, engine speed, engine load, vehicle speed, throttle position. Then we clean the data i.e. remove the data when car / vehicle is idle because that value can affect the average driving speed. Then, the clean data is processed further.
5. Before using the system on real condition, we need to train and test the classification model. (Decision Tree, Adaboost classifier). Here, we use the data we already have, to train the model. Then by looking at the accuracy we finalize the classifier model we are going to use.
6. Finalized model will be uploaded on cloud.
7. Then, the classifier algorithm will be applied on preprocessed real time data to classify the driver behavior.

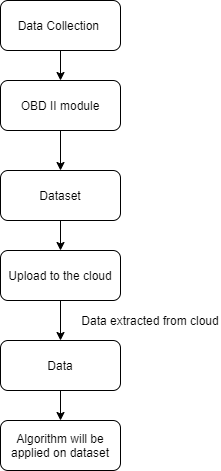


Figure 20 Driving behavior flowchart

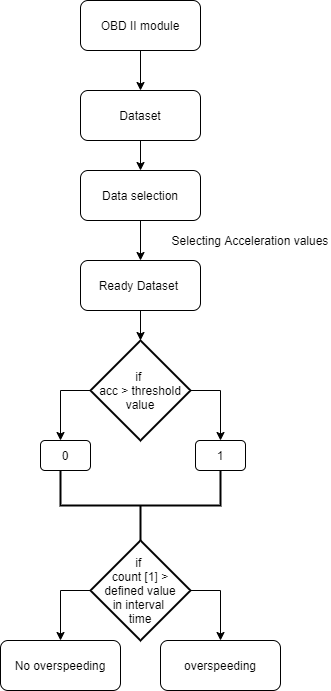


Figure 21 Detecting over speeding behavior

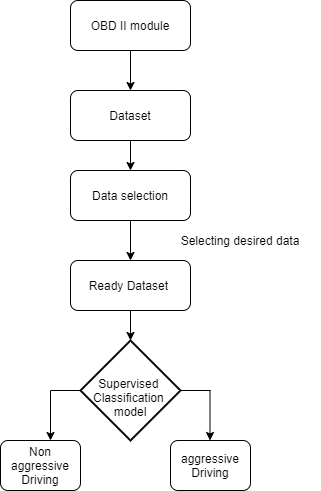


Figure 22 Classifying driving behavior

.

# Road anomalies

## Sub Features: -

1. Pothole detection,
2. Hump / speed breaker detection,
3. Road quality

Requirements: -

1. Hardware requirement to gather acceleration data (OBD II or mobile sensor) and to capture image for image processing.
2. System should be able to detect pothole or speed breaker in real time and mark the spot on map/GPS. Also able to tell about road quality by looking at acceleration data and no of potholes are there on road.
3. System should be able to upload to the image and position of detected pothole and corresponding acceleration values.
4. Pothole detection algorithm or setup should be able to run for different cars.

## SWOT analysis

|  |  |
| --- | --- |
| STRENGTH   1. Real time pothole detection. 2. Acceleration data and image processing, two steps are occurring simultaneously will give better efficiency. | WEAKNESS   1. In image processing presence of shadow or in night time, it may not give high accuracy. 2. In normal driving condition, if the pothole is covered by car in front. |
| OPPURTUINITY   1. We are alerting about pothole location to respective authority so they can act and other user can also see how many potholes are there on route to select the route. | THREATH   1. False true can be generated that can mislead other users about route. |

## Literature review

It is essential to identify pothole and humps in order to avoid accidents and damages to the vehicles that is caused because of distress to drivers and also to save fuel consumption. In this regard, this work presents a simple solution to detect potholes and humps and hence avoid accidents and help driver. And also in assisting in self driving for cars.

Potholes and humps are detected using ultrasonic sensors and their information like location is provided to the authorities with the help of GPS and GSM and alerts the driver regarding the bumpy roads and potholes on its path. With this information the authorities could take measures to repair the roads which in turn help to avoid accidents caused due to nontechnical humps and hence lower the percentage of damaged roads and avoid road accidents. [8]

Potholes are detected using Image Processing Technique and Ultrasonic Sensors are used to detect humps. Use of ultrasonic sensors to detect the speed breakers, it works on principle that transmitter triggers the frequency from transmitter and the receiver will wait for the wave to return. The wave returns after getting reflected by any object. If there is a speed breaker on the road, waves will return sooner than usual and alarm will be triggered.



Figure 23 Ultrasonic sensor and receive-transmit operation

Pothole detection: - The camera captures the image of the road continuously through Open CV in RGB form. Then it converts the image from RGB form to HSV by image processing technique. It is then dilated for image enhancement of the concentrated region. The obtained image is then compared with the stored pothole reference image in the database. If, the captured image matches with stored image, then the Raspberry Pi sends the location of the pothole to the municipal officials through E-mail along with the captured image.

The location can also be saved and shared in the test tool. It also generates alert to the driver through LCD display and speaker that a pothole is detected. This process is repeated.

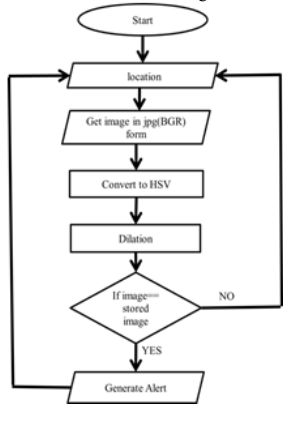


Figure 24 Flowchart for pothole detection

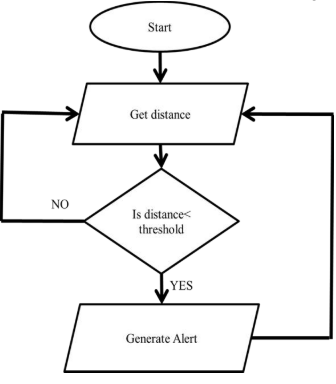
 [8]

Figure 25 Flow chart for speed breaker detection

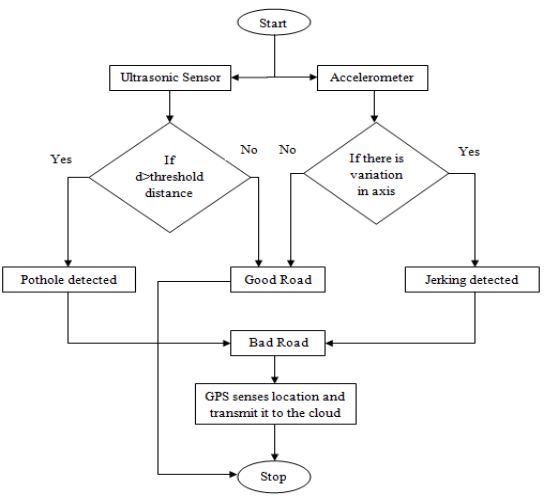
 [9]

Figure 26 complete flowchart of detection

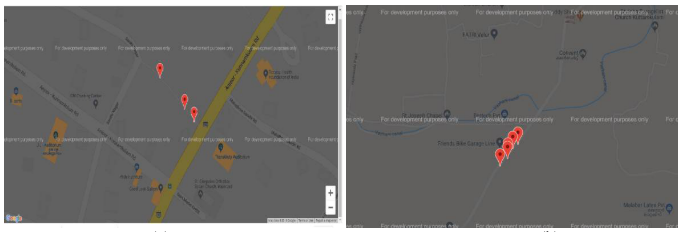


Figure 27 Location of potholes plotted in map [9]

While comparing both, figure (a) shows less number of potholes and figure (b) shows more number of potholes. The user can identify the better roads from this. The authority can also do the maintenance of roads which have pathetic conditions.

The location of these potholes would be available on a centrally hosted map which can be accessed by both end users and civic authorities. So that, next time while selecting the route end user will also see the factor how many potholes are there on his/her route.

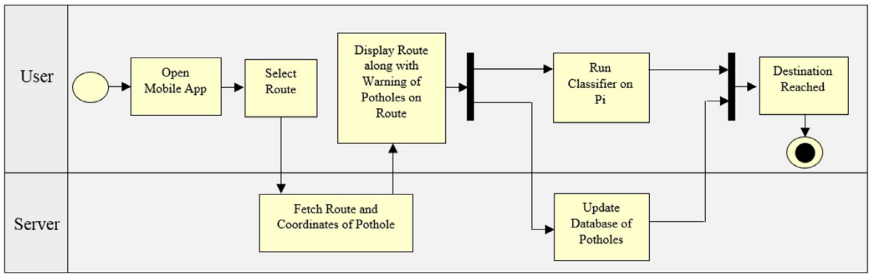


Figure 28 System model of deepbus [10]

User can find routes with information about potholes and selects the best route for destination by running the classifier on Raspberry Pi. The overall process is divided into three main phases:

1. Data Collection and Preprocessing: Using the sensors available to collect data on road information. This will involve manually labelling timestamps with presence of potholes and bumps. This can be done by using a smartphone or a Raspberry Pi with appropriate sensors. After data is collected, it will need to be preprocessed which includes removal of missing data, outlier detection, grouping data etc. It is to be noted that the suspension, steering, braking, acceleration of all vehicles are different. So, it is possible but not necessary that multiple models will have to be trained in the future for different vehicles. For our domain, we have done this research work on a single vehicle.
2. Training: Training various machine learning models on the obtained datasets. This poses questions like which model will help us achieve the best Accuracy, and least False Positive and False Negative rate.
3. Classification/Testing: After the first two phases, the model is deployed on phones and collects data from the smart- phone sensors in real time and classify the presence of a pothole/bump. Misclassifications can be minimized by methods like Weighted Polling, Majority Voting etc. The locations will then be marked on a centrally hosted map which will be available to all users.

After running a script on the raw data, we obtained the final dataset which was in .csv format with the following eight columns: [Time, Ax, Ay, Az, Gx, Gy, Gz, Pothole]. Label 1 denotes “pothole” and label 0 denotes “no pothole”

This dataset will undergo the classification algorithm to detect, whether it is pothole or not.

We have compared the performance of various machine learning models (Logistic Regression, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Naive Bayes, Decision Tree, Random Forest and Ensemble Voting) based on different parameters (Accuracy, F-score, Precision and Recall) and identified that Random Forest is the best model for pothole detection.

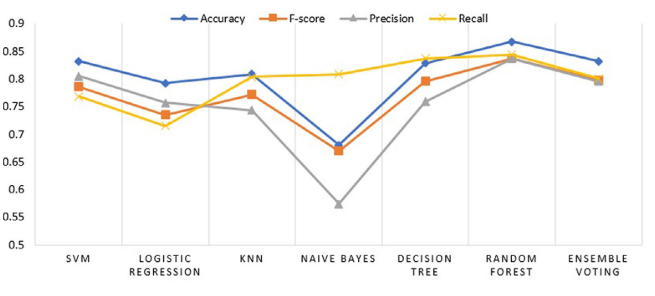


Figure 29 Performance comparison of various machine learning algorithms [10]

We can approach pothole detection problem by approach of image processing for the detecting and positioning of potholes on roadways. Several edge detection and machine learning algorithms such as Canny edge and K-means have been tested and implemented on images to render satisfactory results. However, image-based pothole detection methods suffer from challenges such as obstructions, quality of cameras used and varying conditions of light that result in shadows. These drawbacks often cause inaccuracy of results leading to false positives or false negatives.

This paper proposes a low-cost hardware setup that combines low-computational accelerometer data-based method and high-computational image-based approach. The system would be a low-cost implementation as it would make use of a smartphone and an OBD-II. The two approaches combined would increase the accuracy and efficiency of the system when compared their independent implementation.

The proposed method focuses on a low-cost option and reuse of existing hardware resources, it utilizes smartphone and OBD II module. The method can be called a low-cost implementation as it makes use of minimal hardware components. The role of the smartphone in this implementation with an inbuilt IMU is to provide visual data, as well as an integrated dataset that involves accelerometer data and other vehicle parameters obtained from the OBD-II module.

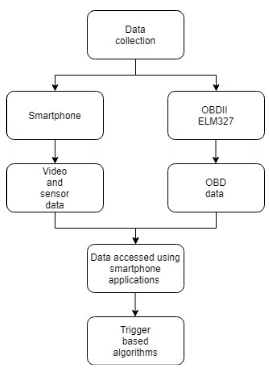


Figure 30 Flowchart of data acquisition

The vehicle used for data acquisition are swift desire (2019) and XUV 500 (2013). The data is recorded at 1Hz sampling rate. The vehicle is stationary for 30 seconds in order to acquire idling conditions. Datasets have been collected for different durations of time mostly at an interval of 20-30 minutes, accessed through a mobile application to gather OBDII data (Torque Pro).

The data captured by the Tri-axial accelerometer is filtered using a high-pass filter to eliminate vehicular vibrations and noise. The flow chart of the method is shown in figure 3. The filter function consists of a low-pass filter which is then converted into a high-pass filter by the means of spectral inversion. The output is then convoluted with the accelerometer signal to obtain only a band of high frequency signal from the data. As a part of the preliminary approach, the Z-Thresh method has been implemented. The Z-Thresh algorithm confirms that peak acceleration in the vertical axis (X-axis) is a prime characteristic that determines anomalies such as potholes on roads.

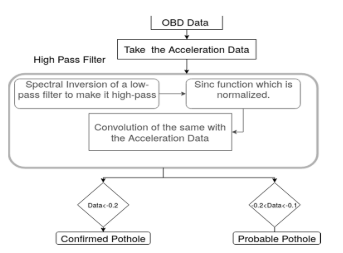


Figure 31 Flowchart of low-computational data-based trigger.

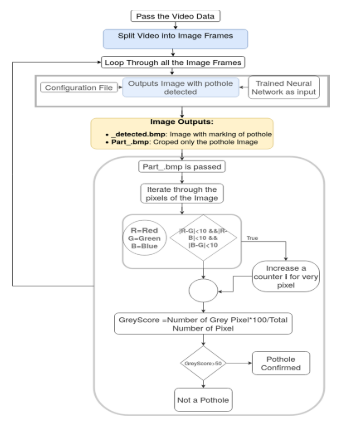


Figure 32 Flowchart of high-computational image-based trigger

The result of the image-based trigger approach positively validates potholes encountered by the vehicle. The algorithm is expected to detect potholes using the R- CNN method. The result also displays several parameters such as vehicle speed and accelerometer data at the location. The matched time frame is extracted, and thresholding methods are implemented on the corresponding accelerometer data.

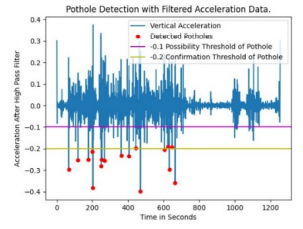


Figure 33 Pothole validation through accelerometer data

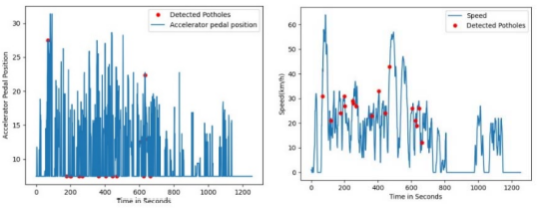


Figure 34 Accelerator pedal position and vehicle speed through potholes

The data undergoes filtering and thresholding methods to render negative peaks where the vehicle encounters the pothole. This data can be considered to validate the results acquired by the visual data. This method can be regarded as the image triggered method. The trigger-based approach also imposed a few drawbacks, Shadows caused by trees and buildings impose false positives that act as a trigger which will be further investigated using accelerometer data. [11]

Integration of several image- processing schemes has been used to produce an algorithm using Python Language from the OpenCV library that can detect and report potholes automatically from a moving vehicle. With a rate of about 8 frames per second, images were processed per frame to detect potholes by analyzing its color, depth, and area. With cars rapidly getting “smarter”, sophisticated sensors are installed allowing the vehicle to profile the road surface under the wheels and identify different road distresses.

However, vehicles are still vulnerable to damages caused by these road distresses like bumps, patches, and especially Potholes. Existing pothole detection can be categorized into three methods: vibration-based method, 3D reconstruction-based methods, and vison-based methods. To date, most automotive technology engineers have turned their implementations of a camera-based installations, which provide visual effects and movement recognition.

The system should be able to detect potholes and send report to the main server that includes the image and the location of the pothole. A unit that was installed in the vehicle with camera module, GPS system, and three LEDs that would act as indicators if the device was turned on, if a pothole was detected, and if the report was sent. The camera module was placed securely in the upper portion of the car, behind the rear mirror, to obtain a good range and to acquire reliable and accurate results. For the pothole detection, the camera module would capture eight images per second in front of the windscreen of the car.

Image-processing is done by first removing other objects in the image such as the sidewalks and the pedestrian, the algorithm would extract first the pavement or the road itself and the extracted image would be the only part that would undergo the main process. Potholes then would be detected based on its color and area using Canny edge detection, and finally contour detection and final filtering.

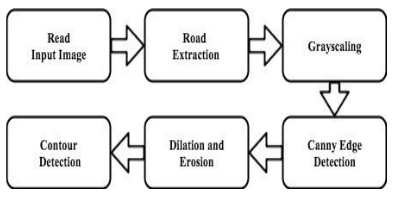


Figure 35 Image processing system

Another key element of the system is the automatic reporting system. If a pothole was detected, the same microcomputer would connect to the Internet to send the image and the location of the said pothole to the main database. Once the system is powered on, the camera would start to capture images in front of the moving car. The microcomputer processes the captured images and analyzes it through image processing. Once detected, the microcomputer connects to the GPS module to acquire the location of the occurrence of the pothole.

The system has two testing set-ups namely, ideal and non-ideal. The ideal set-up included areas that have no other vehicles nearby and do not have other objects, such as trashes, rocks, other road distresses, and even no pedestrians showing in the camera. The non-ideal set-up is a set-up constituted in a normal driving condition.

Accuracy, sensitivity, specificity are the parameters calculated using data we got from tests to check the accuracy and reliability of the system. Response time is also recorded to measure the fast processing of system. [12]

The paper is describing a mobile sensing system for road irregularity detection using Android OS based smart-phones. Selected data processing algorithms are discussed and their evaluation presented with true positive rate as high as 90% using real world data. The optimal parameters for the algorithms are determined.

Pothole detection algorithm is based on simple machine-learning approach using X and Z axis acceleration and the vehicle velocity data as input. The algorithm consists of five consecutive filters: speed, high-pass, z-peak, xz-ratio and speed vs. z ratio. Simple threshold based algorithms such as z-sus, z-peak etc. undoubtedly are suitable for implementation on Android based smart-phones.

The technical requirements for the system are 1) the system should be able to detect events potholes in real time, 2) System should be able to detect events while driving in different four-wheel vehicle types such as passenger cars, minivans and buses.

Analog Devices 3-axis accelerometer ADXL335 is used to record acceleration of vehicle. Mans OS based software was used for raw acceleration data acquisition (sampling rate 100Hz) and transmission through USB interface to a laptop computer. After the acquisition of the first test data set, a search for potential event related features was performed.

Some algorithms to detect potholes: -

1. Pothole detection algorithm Z-THRESH: - Events are represented by measurements with values exceeding specified threshold levels.
2. Pothole detection algorithm Z-DIFF: - Events are represented by consecutive measurements with difference value above specific threshold level.
3. Pothole detection algorithm STDEV (Z): -Events are represented by measurements with standard deviation value above specific threshold level.
4. Pothole detection algorithm G-ZERO. Events are represented by tuple of measurements with all three axis values below specific threshold level.

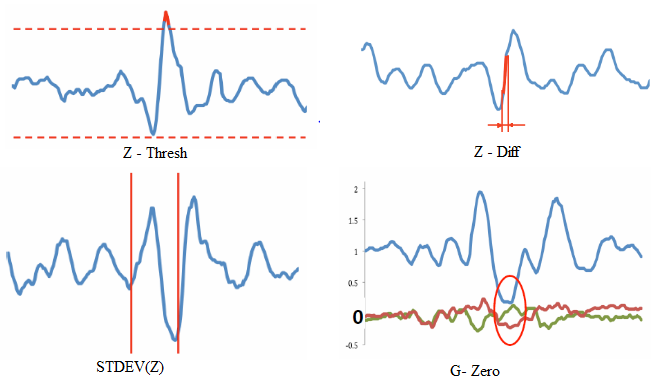


Figure 36 Pothole detection algorithms

This paper describes accelerometer data based pothole detection algorithms for deployment on devices with limited hardware/software resources and their evaluation on real world data acquired using different Android OS based smart-phones. [13]

## Approach

1. We are going to use acceleration data and images of road to detect road anomalies.
2. For acceleration, we can use smartphone sensor to record acceleration values or acceleration data generated by OBD II module. Here, we are proceeding with OBD module data for acceleration values and smartphone camera to capture images while driving.
3. For pothole and hump / speed breaker detection one way we are using is analysis of acceleration value. Now, passing through pothole or speed breaker, acceleration value in vertical direction is gets impacted, so our major focus will be on that axis.
4. Acceleration data from OBD II will be filtered to remove vehicle vibration and noises. Afterwards, dataset is ready to proceed with. Now, out of algorithms from Z-Thresh, Z-Diff, STDEV (Z) based on accuracy we will select one algorithm. Which will help in finding pothole on road.
5. Another way to find pothole is by image processing. While driving video is getting captured by the smartphone camera. Images will undergo image processing using open CV library in python. Where supervised model will be detecting the potholes or speed breakers. This will work on comparing the grey score of image, if it more than the threshold value that we set, then it will detect the pothole.
6. Frequency of data collection and image capturing should be same so that data validation can be done, we can see when pothole is detected by acceleration value if it is detected by image processing. Also, drawbacks of each other will be overcome by using them together and synchronized.
7. Initially, we will take the test on road where we know how many potholes are there and compare them the result to find the accuracy of the system.

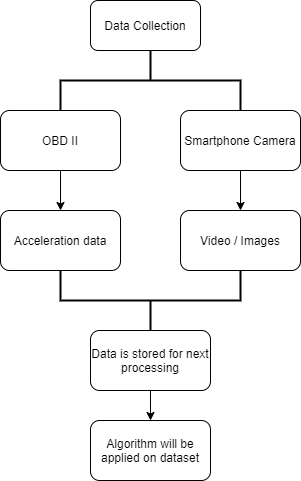


Figure 37 high level diagram

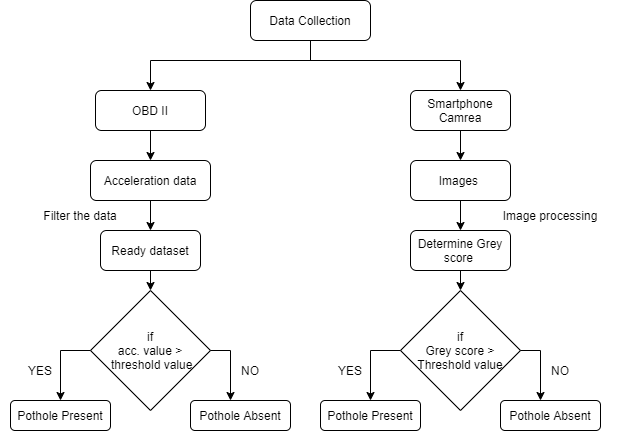


Figure 38 Pothole and Speed breaker detection flowchart

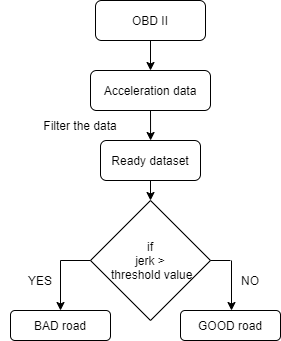


Figure 39 checking the road quality

# References

[1] U. Shafi, A. Safi, A. R. Shahid, S. Ziauddin, and M. Q. Saleem, “Vehicle remote health monitoring and prognostic maintenance system,” *J. Adv. Transp.*, vol. 2018, 2018, doi: 10.1155/2018/8061514.

[2] S. M. Namburu, M. Wilcutts, S. Chigusa, L. Qiao, K. Choi, and K. Pattipati, “Systematic data-driven approach to real-time fault detection and diagnosis in automotive engines,” *AUTOTESTCON (Proceedings)*, pp. 59–65, 2006, doi: 10.1109/AUTEST.2006.283654.

[3] M. Amarasinghe, S. Kottegoda, A. L. Arachchi, S. Muramudalige, H. M. N. Dilum Bandara, and A. Azeez, “Cloud-based driver monitoring and vehicle diagnostic with OBD2 telematics,” *IEEE Int. Conf. Electro Inf. Technol.*, vol. 2015-June, pp. 505–510, 2015, doi: 10.1109/EIT.2015.7293433.

[4] U. Fugiglando *et al.*, “Driving Behavior Analysis through CAN Bus Data in an Uncontrolled Environment,” *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 2, pp. 737–748, 2019, doi: 10.1109/TITS.2018.2836308.

[5] Z. Constantinescu, C. Marinoiu, and M. Vladoiu, “Driving style analysis using data mining techniques,” *Int. J. Comput. Commun. Control*, vol. 5, no. 5, pp. 654–663, 2010, doi: 10.15837/ijccc.2010.5.2221.

[6] C. P. Hwang, M. S. Chen, C. M. Shih, H. Y. Chen, and W. K. Liu, “Apply Scikit-learn in python to analyze driver behavior based on OBD data,” *Proc. - 32nd IEEE Int. Conf. Adv. Inf. Netw. Appl. Work. WAINA 2018*, vol. 2018-January, pp. 636–639, 2018, doi: 10.1109/WAINA.2018.00159.

[7] S. H. Chen, J. S. Pan, and K. Lu, “Driving behavior analysis based on vehicle OBD information and adaboost algorithms,” *Lect. Notes Eng. Comput. Sci.*, vol. 1, pp. 102–106, 2015.

[8] B. G. Shivaleelavathi, V. Yatnalli, Chinmayi, V. S. Yamini, and S. Thotad, “Design and development of an intelligent system for pothole and hump identification on roads,” *Int. J. Recent Technol. Eng.*, vol. 8, no. 3, pp. 5294–5300, 2019, doi: 10.35940/ijrte.C5936.098319.

[9] A. Terish, S. J. T. D, S. Valsan, P. Sooraj, and N. V Indrasena, “Real Time Pothole Detection and Road Monitoring Sysytem,” pp. 6740–6744, 2019, doi: 10.15680/IJIRSET.2019.0806034.

[10] K. Bansal, K. Mittal, G. Ahuja, A. Singh, and S. S. Gill, “ DeepBus : Machine learning based real time pothole detection system for smart transportation using IoT ,” *Internet Technol. Lett.*, vol. 3, no. 3, pp. 2–7, 2020, doi: 10.1002/itl2.156.

[11] K. S. Ashwini, G. Bhagwat, T. Sharma, and P. S. Pagala, “Trigger-Based Pothole Detection Using Smartphone and OBD-II,” *Proc. CONECCT 2020 - 6th IEEE Int. Conf. Electron. Comput. Commun. Technol.*, 2020, doi: 10.1109/CONECCT50063.2020.9198602.

[12] M. M. Garcillanosa, J. M. L. Pacheco, R. E. Reyes, and J. J. P. San Juan, “Smart Detection and Reporting of Potholes via Image-Processing using Raspberry-Pi Microcontroller,” *2018 10th Int. Conf. Knowl. Smart Technol. Cybern. Next Decad. KST 2018*, pp. 191–195, 2018, doi: 10.1109/KST.2018.8426203.

[13] A. Mednis, G. Strazdins, and R. Zviedris, “Real Time Pothole Detection using Android Smartphones with Accelerometers Research domain • Road infrastructure as blood vessels,” 2011.

# Appendix A

Domain: - AI and ML learning tools

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sr. No | Tech tools | Description | Deadline | Status |
| 1 | AI and Machine learning tools | Unsupervised model – clustering model | 16/4/21 | In - process |
| 2 | Literature survey | Reviewing and implementing the process they used to decide what process we should follow | 16/4/21 | In - process |

Process: -

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sr. No | Tech tools | Description | Deadline | Status |
| 1 | GIT | Day 1 course |  | Completed |
|  |  | Week 1 course |  | Completed |
| 2 | Word | TOC/TOF/ reference |  | Completed |
| 3 | Mendeley | Account open, group create, use to add reference in report |  | Completed |

Tech and tools: -

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sr. No | Tech tools | Description | Deadline | Status |
| 1 | Pycharm | Download and practice python code in it. |  | Completed |
| 2 | Sololearn Python | Practice classes and complete regular expression |  | Completed |
| 3 | Pep8 | Practice Pep8 on code |  | Completed |
| 4 | Low code platform(pycaret) | Practice pycaret library on unsupervised model | 16/4/21 | In - Process |
| 5 | Kaggle | Practice pycaret library on one of the data set from Kaggle | 16/4/21 | In - Process |
| 6 | Python | Reimplement the existing code | 16/4/21 | In - Process |