ASHESI

ASHESI UNIVERSITY COLLEGE

EVALUATING AND CHOOSING A MACHINE LEARNING ALGORITHM FOR CLASSIFYING ROAD SURFACE QUALITY DATA

APPLIED PROJECT

B.Sc. Computer Science

ANTHONY ANABILA ABEO

2018

ASHESI UNIVERSITY COLLEGE

EVALUATING AND CHOOSING A MACHINE LEARNING ALGORITHM FOR CLASSIFYING ROAD SURFACE QUALITY DATA

APPLIED PROJECT

Applied Project submitted to the Department of Computer Science, Ashesi University College in partial fulfillment of the requirements of the award of Bachelor of Science degree in Computer Science

ANTHONY ANABILA ABEO

April 2018

`

DECLARATION

I hereby declare that this applied project is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature:……

Candidate's Name:……

Date:……..

I hereby declare that the preparation and presentation of this applied project were supervised in accordance with the guidelines on supervision of applied project laid down by Ashesi University College.

Supervisor's Signature: ……

Supervisor's Name:……

Date:……

ACKNOWLEDGEMENT

I would like to express my deepest gratitude to everyone who contributed to the completion of this work. First, I would like to acknowledge with profound appreciation, the support of my advisor Dr. G. Ayorkor Korsah, who provided invaluable guidance from the start of this project to its completion. Further, I would like to thank Mr. Peter Okwei for driving me around during data collection exercises. Without their support this work would indeed have been difficult to finish. Finally, many thanks to the Computer Science Department of Ashesi University College for providing the equipment needed to undertake this project, and for feedback that significantly improved this work.

ABSTRACT

Considering the importance of roads to a community, stakeholders (Governments, Motorists etc) need up-to-date information about the state of roads for decision making . This problem inspired Vorgbe's (2014) work in implementing a machine learning classifier that could accurately classify roads as "good", "fair" or "bad". This information can then be visualised on Google Maps. However, with his algorithm failing to accurately classify some roads, this project seeks to evaluate five classification algorithms to determine which one is best for classifying road surface quality data.

To do this, we collected x, y, z acceleration and location data, extracted the desired features from it, performed a 10-fold cross-validation training on the data to choose the best model and then tested on a new set of examples to determine the model that accurately classifies the data. From the data available, the decision tree model produced the best performance with true positives of 97% accuracy for bad roads, 81% accuracy for fair roads and 93% accuracy for good roads. The overall accuracy on the test set is 92% with a precision of 92% and recall of 90%. This means that, this model is more likely to accurately predict a new data point as belonging to its true class. The other algorithms (Logistic Regression, Random Forests, Support Vector Machines and Nearest Neighbour) performed well when classifying the "good" and "bad" road data but instead classified the "fair" road data as "good" road.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Misclassification Error when choosing Parameters During Training

CHAPTER 1: Introduction

Roads are as important to a society as blood vessels are to the human circulatory system (Mednis, Strazdins, Zviedris, Kanonirs & Selavo, 2011). This is so because there is a need to transport human resources, goods and services from one place to another. The features of a road determines how well it is able to serve its purpose, and one important feature is the surface quality of the road. Bad roads make transportation uncomfortable for motorists, damage vehicles, cause road accidents and negatively affect economic activities (Mednis et al., 2011). For example, if employees always arrive at work late or a delivery service fails to meet delivery deadlines, revenue of the organization is affected negatively. Motorists will be able to use roads safely when they are well informed about the state of the roads. In addition, maintenance of roads can be carried out frequently if the government has access to information about the current state of roads in the country when they need it (Vittorioa et al., 2014). Having real time information about the state of roads requires that data be collected about the quality of the surface of roads (Mednis et. al., 2011). One approach is to ply the roads while manually identifying the roads and attaching the appropriate surface quality label (e.g. “Good”, “Bad” etc). However, this approach is highly inefficient, slow and rather expensive. Hence, there is a need for a better approach: an automated approach such as participatory sensing.

Participatory sensing is a large-scale, distributed type of sensing that allows citizens to contribute sensed data about their environment using mobile devices equipped with sensors (Kanhere, 2011, p. 3).

Applying participatory sensing to the task of collecting data about road surface quality is ideal for a number of reasons. First, it is cheap since the tools needed (smartphones and cellular network services/WiFi) are available and are easy to use. Second, participants carry their smartphones around which allows for the collection of data in several locations. Third, the availability of software development kits (SDKs) for the Android and iOS mobile platforms as well as the mobile application stores such as Google Play allow rapid development and deployment of applications to users (Kanhere, 2011, p.3). In this work, we apply participatory sensing to determine road surface quality by allowing participants to record accelerometer readings in the x, y and z axes from their smartphones; this data is then used to train a machine learning classification algorithm to help classify the quality of roads. This information can then be visualized on Google Maps to help motorists and other stakeholders such as the government make appropriate decisions concerning the roads.

There has been prior work by Vorgbe (2014), Doku (2014) and Boohene (2017) on the main components of this project. Vorgbe (2014) worked on collecting sensor information from smartphones and implemented and trained a logistic regression classifier to classify the quality of road surfaces. Doku (2014) investigated whether or not motorists will find it useful to have road surface quality information as part of Google Maps and Boohene's (2017) work focused on automating the collection and visualization of road surface quality data to help motorists. In Vorgbe's (2014) work, he noted that vehicles experience varying levels of vibrations depending on the surface

quality of the road on which they are travelling and if data on the levels of vibrations are recorded, a system can be developed to accurately classify the roads based on the surface quality. This was achieved by recording accelerometer and GPS data using a smartphone. This data was processed and used to train a logistic regression classifier. Boohene (2017) built on the work done by Vorgbe (2014) and Doku (2014) by building a mobile application to automate the collection of road surface quality data, using the classification algorithm developed by Vorgbe (2014) to process and classify the data. After classification, the data is transferred to a server to be displayed on Google Maps.

From Vorgbe's (2014) work, he noted some limitations of the logistic regression classification algorithm. The algorithm failed at the following set of classification tasks: **bad** versus **fair** roads, **fair** roads versus **good and bad** roads and **bad** roads versus **good and fair** roads. Considering that the classification algorithm used is the most important part of the project, it is necessary that the algorithm is able to accurately classify roads based on their surface quality. Otherwise, the information visualized on Google Maps will be inaccurate and will further frustrate the users of the system. In addition, one limitation in collecting the data stemmed from the fact that the smartphone used had to be placed in a fixed orientation in the vehicle. This approach was needed to make the phone mirror the x, y and z axes of the vehicle so that the sensor measurement of the phone in each axis will correspond to the axis of the car. This is limiting because, in trying to crowdsource data from users, it would not be ideal that users fix their phones on the dashboard. Hence it is necessary that an improved

method of collecting accelerometer data is used that does not require that the phone be in a particular orientation. To address these limitations, this work seeks to compare and evaluate various supervised machine learning classification algorithms to determine which one will provide the highest accuracy when classifying road surface quality data. After training, this model will be used as part of a road surface quality data crowdsourcing system.

This paper is organized as follows: Chapter 2 discusses related work, Chapter 3 discusses approach, Chapter 4 reviews results and Chapter 5 is on conclusions and recommendations.

CHAPTER 2: Related Work

In recent times, there has been an increase in the amount of research aimed at automatically monitoring and reporting on road surface conditions using sensor data. Participatory sensing has helped in fueling this trend mainly because of the ubiquitousness of smartphones. Researchers have proposed and built systems that automate the collection, storage and processing of sensor information about road surface conditions in order to determine the state of such roads. This section seeks to draw attention to some relevant work done in this area.

2.1 Similar Projects

Mednis et. al. (2011) worked on automatically detecting potholes in real time as well as using smartphones for recording sensor data about the state of roads for later processing. The smartphone is equipped with a GPS and a 3-axis accelerometer with support for cellular and/or WiFi connectivity. The GPS sensor is used to record location and the accelerometer is used for recording x, y and z acceleration. The system utilizes the client-server architecture, where an Android smartphone running an application that records sensor data serves as the client. The server side consists of a Java web application and a MySQL database. When sensor data is recorded, it is processed and stored in a database which is occasionally synced with the main database. In addition, the Java web application uses the API to visualize pothole data. The authors collected data using Samsung Galaxy S, HTC Desire and Samsung i5700 at a frequency of 90 Hz, 53 Hz and 26 Hz respectively. After collecting the data, they proposed four algorithms for data processing: ZTHRESH and Z-DIFF (for real-time pothole detection) and STDEV(Z) and G-ZERO(for offline post-processing of data).

Z-THRESH classifies the z-axis values into one category of pothole or another (small pothole, cluster of potholes, large potholes) depending on some threshold value of the z-axis acceleration. The next algorithm to be tested on the dataset was Z-DIFF. This algorithm performed "a search for two consecutive measurements with difference between the values above specific threshold level" (Mednis et. al. 2011). The algorithm detected fast changes in the Z-axis measurements. The STDEV(Z) algorithm computes the standard deviation of the z-axis acceleration of a fixed window and classifies the event by comparing the standard deviation to some threshold level. G-ZERO searches for the x, y, z values where the vehicle has zero gravity. This indicates that the vehicle is either entering or exiting a pothole and that location is then marked as having a pothole. After analysing the results, the authors noted that the pothole detection algorithms produces true positive results as high as 90%.

In a similar study, Vittorioa et. al. (2014) developed an automated sensing system for monitoring road surface conditions using smartphones. The system consisted of an Android application that recorded and processed accelerometer data from an Android device and a server that records and stores this data. Vittorioa et. al. (2014) used acceleration and GPS data from smartphones to classify and localize road surface anomalies. The accelerometer records the z-axis acceleration movements above some specified threshold (high-energy event). To accurately classify road surface conditions based on z-axis acceleration and location data, the researchers had to make some adjustments to the accelerometer and GPS readings. First, they wanted the accelerometer to record data at the the same frequency as the GPS sensor (1 Hz). Hence they estimated the minimum, average and maximum z-axis acceleration values for five

readings per second since the lowest accelerometer frequency was 5 Hz. Second, Vittorioa et. al. (2014) ensured that the z-axis reading of the accelerometer corresponded to that of the vehicle. This will ensure an accurate measurement of the vertical axis acceleration even if the phone is randomly place in the car. This was achieved by computations utilizing Euler angles. Third, they filtered out low-impulse events that fell below a specified threshold value since they do not correspond to road anomalies. Essentially, Vittorioa et al.'s (2014) system works by collecting z-axis accelerometer readings, filtering out the low-impulse events and transferring the data to the server in real-time. If more than five high-impulse events are recorded at a particular location, the server marks that spot as having road anomalies and returns this new information to the users of the system. The authors carried out tests on their system by fixing three Android smartphones on the dashboard of a car and driving 14km in the city of Rende, Italy while recording and processing Z-axis accelerometer readings. The results from the tests showed a proportional relationship between the amount of impulse and the depth of potholes.

Huffman, Mock & May (2013) built a system that collects and classifies road surface conditions using smartphones mounted on bicycles. They collected acceleration and location data using the accelerometer and GPS sensors respectively. They adopted two strategies for classifying road surface quality as "smooth", "rough" and "bumpy". First they trained a supervised machine learning classification algorithm to classify road segments on a standard laptop and then transferred the model to the smartphone for classifying sensor data. They extracted the following features from the training data: <*speed, inclination, acceleration mean, acceleration variance, acceleration standard deviation*> and trained K-Nearest

Neighbour and Naive Bayes models using a 10-fold cross validation and features selection optimization. The Nearest Neighbor model performed slightly better than the Naive Bayes model. The best performance overall of the machine learning approach was 78%. The second approach involved training a binary classifier to detect bumps using some specified threshold z-axis values which led to an accuracy of 97%. If the z-axis value exceeds the threshold, it is recorded as a bump and otherwise, not a bump.

Eriksson et. al. (2011) studied the utilization of mobile sensing for automatically monitoring and reporting road surface conditions in Boston, USA. They set up embedded accelerometers and GPS sensors on seven taxis and used that for data gathering. Eriksson et al. collected acceleration and location data stored in the following format: <*time, location, speed, heading, 3 axis acceleration*>. Time, location, speed and heading were recorded using a GPS device and the acceleration data was recorded using an accelerometer. They collected data by repeatedly driving on several roads and having a passenger manually label each event encountered by pressing a key on the laptop that corresponds to the class the said event belongs to e.g smooth road, manhole, pothole, rail crossing, etc. Eriksson et. al. used speed filters to remove any high energy event that happens at low speeds such as door slams. They then determine a threshold value for each class under consideration and classify each event based on the one of the specified threshold.

Vorgbe's (2014) study was aimed at monitoring and classifying entire segments of roads as "good", "bad" or "fair". The aim of the study was to classify road segments as "good", "fair" or "bad" using a logistic regression classifier. They collected acceleration and location data

using accelerometer and GPS sensors in smartphones. Vorgbe (2014) observed that, vehicles experience alternating levels of vibrations depending on the surface on which they are travelling. Vehicles on an uneven surface may experience an exceptional amount of vibrations as compared to vehicles on a smooth road. In addition, if data on the levels of vibrations are recorded, Vorgbe (2014) noted that a system can be developed to accurately classify the roads based on the surface quality. This was achieved by fitting a smartphone to the dashboard of a moving vehicle. The accelerometer and GPS sensors recorded the acceleration and location data respectively. The accelerometer data was recorded at a frequency of 4 Hz while the GPS data was recorded at 1 Hz.

Vorgbe (2014) implemented and trained a LR classifier using the data they collected. The process involved extracting features from the data, generating parameters and testing the accuracy using test data. Since logistic regression is a binary classifier and with three classes under consideration (good, bad, fair), Vorgbe (2014) used the one-vs-rest approach to classify the roads. After testing, the good vs bad/fair classifier produced 87% accuracy, Bad vs. Good/Fair was 52% accurate and Fair vs. Bad/Good's accuracy was under 50%. In addition, he noted some limitations of the logistic regression classification algorithm. The algorithm was unable to accurately differentiate between bad and fair roads. In addition, the algorithm failed to clearly separate fair roads versus good/bad roads and also failed to differentiate bad roads from good/fair roads.

2.2 Participatory Sensing

It is worth noting that, in order to gather sensor data in a way that is scalable,

researchers have resorted to participatory sensing. This is a more efficient and inexpensive way of gathering data from smartphone users as compared to collecting data as an individual. Kanhere (2011) and Tilak (2013) provided a comprehensive overview of participatory sensing, especially as used in crowd-sourcing data from mobile smartphones. Traditional sensing mechanisms that involve deploying static wireless sensors in urban areas is not only expensive, it is also not scalable as not many sensor devices that can be deployed as compared to the case of participatory sensing. It is from this perspective that Kanhere (2011) considers the possibility of crowdsourcing sensor information using smartphones. Though not specifically designed to be used as sensors, smartphones come with a wide array of sensors: camera for images and videos; accelerometer for movement and direction of the phone; GPS for location etc. In addition, the pervasiveness of smartphones and the spatiotemporal coverage they provide make them an excellent tool for crowdsourcing sensor data. This, along with it numerous benefits, is fueling the adoption of participatory sensing (Tilak, 2013). Kanhere (2011) noted that, in participatory sensing, sensor data recorded by participants is transferred to a central server via cellular or wireless networks for processing. On the server, the data is analysed and made available to users by being visualized on navigation systems such as Google Maps.

The proliferation of participatory sensing has led to several applications which can be categorized under two broad categories: people-centric and environment-centric sensing. People-centric sensing focuses on collecting sensor data to help understand human behaviors and activities. It gathers sensor data about the user and can be applied in areas such as

personal health monitoring, monitoring and documenting sport experiences, enriching social media content etc. In environment-centric sensing, data is recorded about the surroundings of the user. This type of sensing requires exploiting data at a community scale and it monitors environmental factors such as air, noise, road and traffic conditions. Some challenges of participatory sensing include dealing with incomplete data, preserving user privacy and ensuring that the application is energy efficient and does not consume the participants' battery (Kanhere, 2011). Tilak (2013) discussed the various steps involved in building a participatory sensing application and the categories under which these applications fall. They defined three types of participatory sensing:

1) **collective design and investigation:** where the participants are involved in all stages of the projects from defining research objectives to deploying the application.

2) **Public contribution:** where participants are only responsible for data collection.

3) **Personal use and reflection:** where the users are in complete control of the project usually for their personal benefit. Tilak identified the process of building a participatory sensing application. He also discussed the architecture of such an application as consisting of a sensor network and a backend system and identified some of the categories of participatory sensing applications. These include environmental monitoring, health and fitness monitoring, transportation and civil infrastructure monitoring, and urban sensing.

The paper seeks to build upon Vorgbe's (2014) work by evaluating and choosing a classification algorithm that can accurately classify roads based on accelerometer and GPS readings from smartphones.

`CHAPTER 3: Approach

Machine learning algorithms perform better in some domains than others. An algorithm tends to perform much better when applied to its "favorite" domain. An example is the common use of the Naive Bayes algorithm in text classification problems. It is important to choose the algorithm that performs best in the domain one is operating in and given the data available. In addition, different performance metrics are used for different domains. Like in the case of choosing an algorithm, it is equally important to choose an appropriate metric for measuring the performance of an algorithms for the domain under consideration. Examples of performance metrics and the areas in which they are used include **Precision/Recall** for information retrieval, area under **Receiver Operating Characteristics (ROC)** curve for Medicine and **Lift** for marketing tasks (Caruana & Niculescu-Mizil, 2006). In this paper, the performance metric employed is accuracy score and standard deviation. The algorithms evaluated include Support Vector Machines (SVM), Nearest Neighbor (kNN), Logistic Regression (LR), Decision Tree (DT) and Random Forests (RF). For each algorithm, we performed a k-fold cross-validation where k = 10 and found the mean accuracy score of the 10 "folds" along with the standard deviation. In addition, certain parameters and hyperparameters were tuned to help determine the ones that produces the best accuracy. Before the performance of each algorithm can be evaluated, we collected data, extracted the desired features and trained the algorithms.

3.1 Data Collection

The data collection strategy used here is the same as the one employed by Vorgbe (2014). The data units of interest for this project were the acceleration and gravity value along the x, y and z axes, longitude and latitude corresponding to the location where the accelerometer data was recorded. The device used to record the data units is a Google Nexus 7 tablet equipped with accelerometer and GPS sensors. The tablet was running an Android application that recorded acceleration and GPS. The tablet was fixed to the armrest of a vehicle with the tablet screen facing upwards. This was to allow the axes of the tablet to be aligned with that of the vehicle so that the values recorded by the sensors can be attributed to the movement of the vehicle. The vehicle was then driven on roads of varying quality while the application recorded accelerometer and GPS data at a frequency of 4 Hz. During this process, a human was responsible for manually labelling the data as "good", "fair" or "bad" road depending of the nature of the road. The data recorded was in the following format: <*timestamp, x, y, and z-axial acceleration, x, y, z-axial gravity, latitude, longitude*>. The road segments chosen for data collection contained stretches that qualify as "good", "bad" and "fair". The road circuit used starts from Kitase junction in the Eastern Region of Ghana through the Accra-Aburi road to the Achimota overpass. From the Achimota overpass, the route joined the Nsawam-Kumasi highway, turning at the ACP junction at Pokuase to join the Kwabenya-Berekuso road then back to Ashesi University.

3.2 Feature Extraction

In addition to others, the features used for the work were the same features used by

Vorgbe (2014) . Good roads will have lower z-axis values as the vehicle would not experience severe vibrations whiles the bad roads will have higher z-axis values. With this in mind, Vorgbe (2014) wrote a program that divides the raw data collected into windows (40 data points each) and computed the **mean**, **minimum value**, **maximum value**, **standard deviation** and **variance** for each window. The program then extracts the values for the z-axis and writes them to a file as the features for each class of road. Each data point of the features has the following format: <*z-axis mean, z-axis variance, z-axis standard deviation, z-axis minimum value, z-axis maximum value*>. From here on, these features will be referred to as *z-axis features* to differentiate them from another set of features we used for training and testing. For the new features, we found the dot product of the x, y, z linear acceleration and gravity for each data point. The goal with this approach is to measure the vehicle's acceleration in the direction of gravity as this addresses the problem of fixing the orientation of the phone to match that of the vehicle. We then split the dot products into windows (40 data points per window). For each window we compute the **mean**, **maximum** and **minimum** values, **25th, 50th, 75th** percentiles and the **standard deviation** as features. These set of features will be referred to as dot-product features in the rest of this paper. Each data point is in the following format: <*mean, maximum, minimum, 25th percentile, 50th percentile, 75th percentile, standard deviation*>. Then for each class, the corresponding features are split into two sets: 70% for training and 30% for testing. This produced the following number of training and test data points:

Table 3.1 : Number of data points used for training and testing

	TRAINING	**TEST**
GOOD ROAD	1255	538
FAIR ROAD	477	204
BAD ROAD	702	303

3.3 Training

This subsection explores what each algorithm does and the parameter(s) that produced the lowest misclassification error. The training was achieved using Scikit-learn, a python machine learning library that has implementation of the algorithms under consideration.

3.3.1 K Nearest Neighbor

The idea behind nearest neighbor algorithms is that for some new test data point to be classified, find a set of *k* training points that are closest to the new point (neighbours). The new point is assigned the label of the most frequent class of its neighbours. The number neighbors can be specified by an integer value or can be based on the number of examples that fall within a specified radius from the new point. The distance between the points can be any standard measure of distance such as Euclidean (Scikit-learn, 2011).

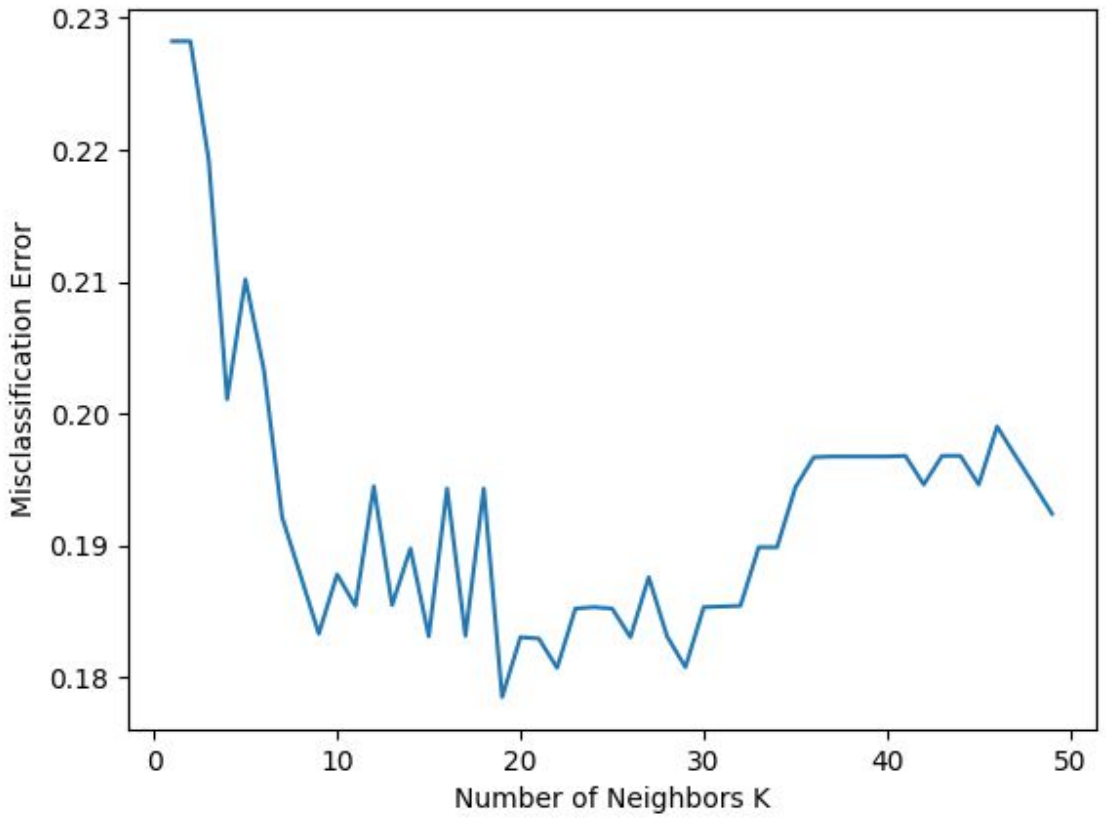

Figure 3.1: Misclassification errors for different values of K

After performing the cross-validation training for k = [1, 50], the value of k that minimizes the misclassification error is 19 as shown in figure 3.1. In addition, Euclidean distance was used as a distance metric as opposed to the radius from the new test point. This produced a mean accuracy score of 0.82 and standard deviation of 0.12.

3.3.2 Logistic Regression

Logistic regression is a linear classification algorithm that utilizes a sigmoid function to squash the value generated by the model into the range [0, 1]. A large positive value passed

to the sigmoid function will produce an output that is very close to a binary one and a large negative value will produce an output close to binary zero. A test example can then be classified as belonging to class 0 or 1. In the case of multi-label classification, the algorithm uses the one-vs-rest approach (Scikit-learn, 2011). In one-vs-rest, a model is trained to predict a data point as belonging to one class as opposed to the other classes. In the scikit-learn implementation, the *LogisticRegression* class can fit a binary, one-vs-rest or multinomial regression with L1 or L2 regularization. This work utilized L1 regularization linear regression which minimizes the following optimization cost function:

$$min_{w,\,c}\ \|w\|_1\ C \sum_{i=1}^{n} log(exp(-y_i(X^T{}_i w\ +\ c))\ +\ 1)$$

For logistic regression, our parameter of interest is the penalty value the model suffers for making a wrong prediction, which in the case of the scikit-learn library, is indicated by C.

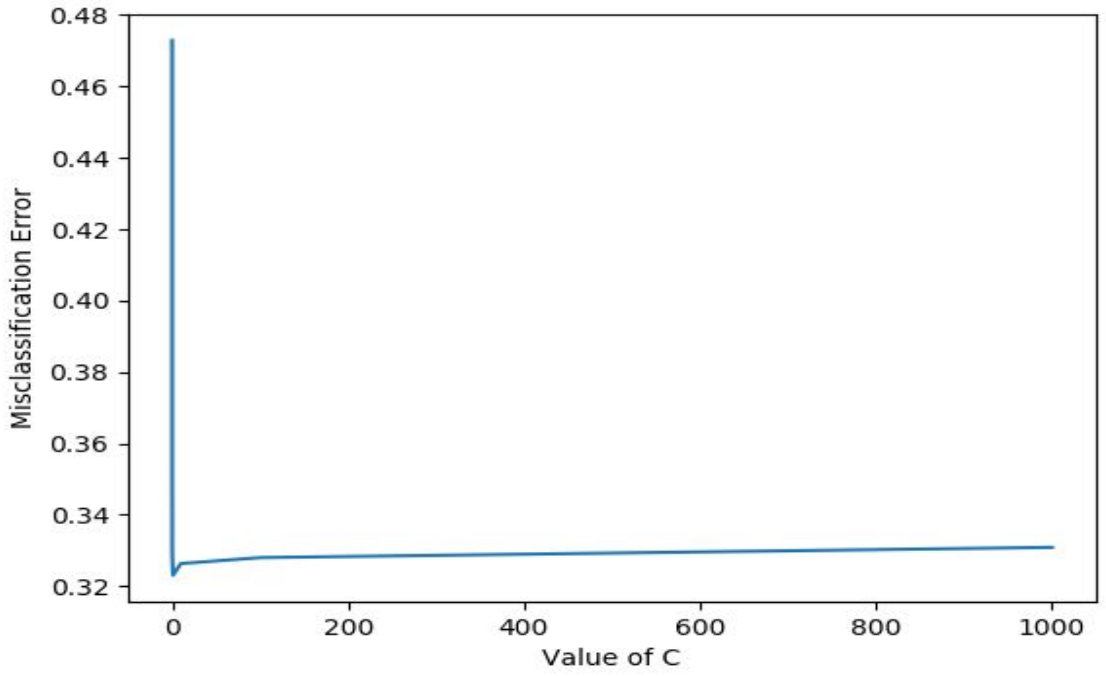

Fig.3.2: Misclassification error for different "model misclassification penalty values"

After performing 10-fold cross-validation for C = [0.001, 0.01, 0.1, 1, 10, 100], the value of C that produces the smallest misclassification error is 1 as shown in fig. 2 above. This produced an accuracy of 0.83 with a standard deviation of +/- 0.09.

3.3.3 Support Vector Machines

Support vector machines (SVM) are linear classifiers that accomplish classification by creating a hyperplane in a higher dimensional space that efficiently separates the data into categories (Adankon & Cheriet, 2009). The optimal separation of data is achieved by the hyperplane that maximizes the distance to the nearest training examples of the classes. That is the functional margin (Scikit-learn, 2011).

Scikit-learn's implementation solves the following dual formulation problem:

$$min_{\alpha,\, \alpha*}\ \tfrac{1}{2}(\alpha - \alpha^*)^T Q(\alpha - \alpha^*) + \varepsilon e^T(\alpha + \alpha^*) - y^T(\alpha - \alpha^*)$$
$$\text{Subject to:} \quad e^T(\alpha - \alpha^*) = 0$$
$$0 \le \alpha_i, \alpha^*_i \le C,\ i = 1, ..., n$$

where e is the vector of all ones, $C > 0$ is the upper bound, Q is an n by n positive semidefinite matrix, $Q_{ij} \equiv K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ is the kernel.

The decision function is $\sum_{i=1}^{n}(\alpha_i - \alpha^*_i)K(x_i, x) + \rho$ (Scikit-learn, 2011).

SVM performs multiclass classification using the one-vs-one (where a model is trained for each pair of class labels) or one-vs-rest (where a model is trained for each class versus the other classes) approaches. In scikit-learn, the choice between one-vs-one and one-vs-rest is specified using the "decision_function_shape" parameter of the *SVM* class. Other parameters scikit-learn's SVM accept include the degree of the polynomial kernel function used and the penalty for misclassifying a data point, C (Scikit-learn, 2011). With C=10, we get the minimum classification error. This produces an accuracy of 0.83 with a standard deviation of +/- 0.12.

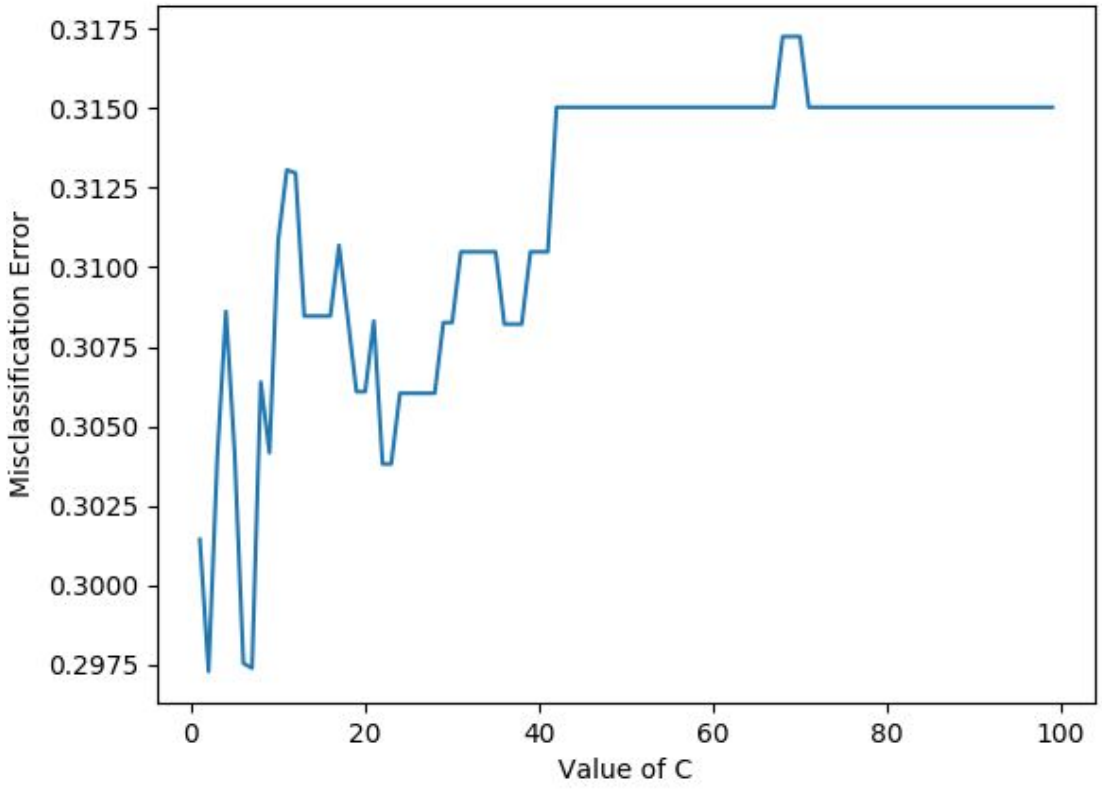

Fig. 3.3: Misclassification error for different "model misclassification penalty value"

3.3.4 Decision Tree

Decision trees (DT) are non-parametric supervised learning algorithms (Scikit-learn, 2011). The algorithm creates a tree structure where the features serve as the nodes and decision making on which branch to take happens on the nodes (figure. 3.4). For this review of DT, we considered maximum depth of the tree and minimum number of splits for each node as parameters. As per figure. 3.5 and 3.6, maximum depth of 3 and minimum splits per node of value 3, produces the highest accuracy score of 0.78 and a standard deviation of +/- 0.1

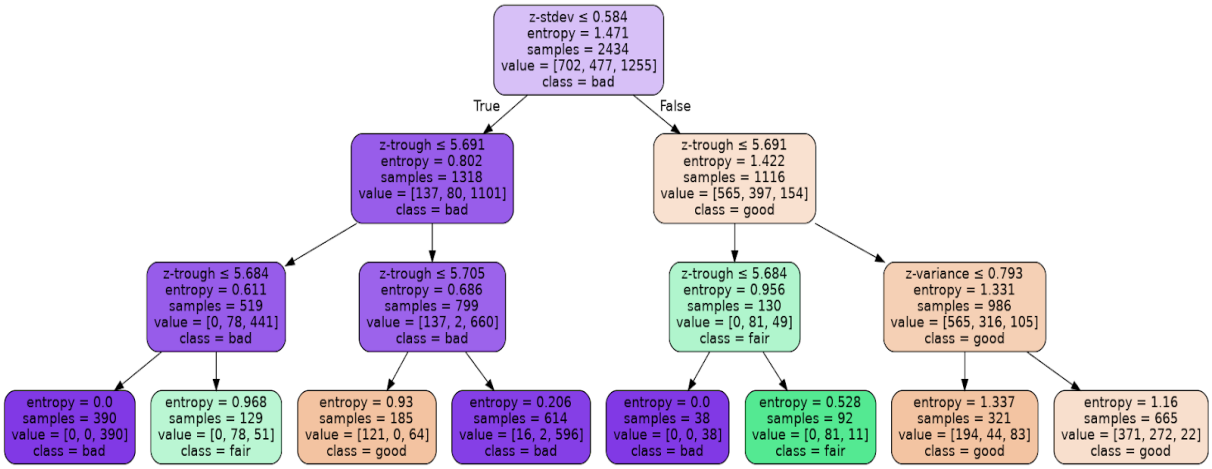

Fig 3.4. Decision tree of height, 3

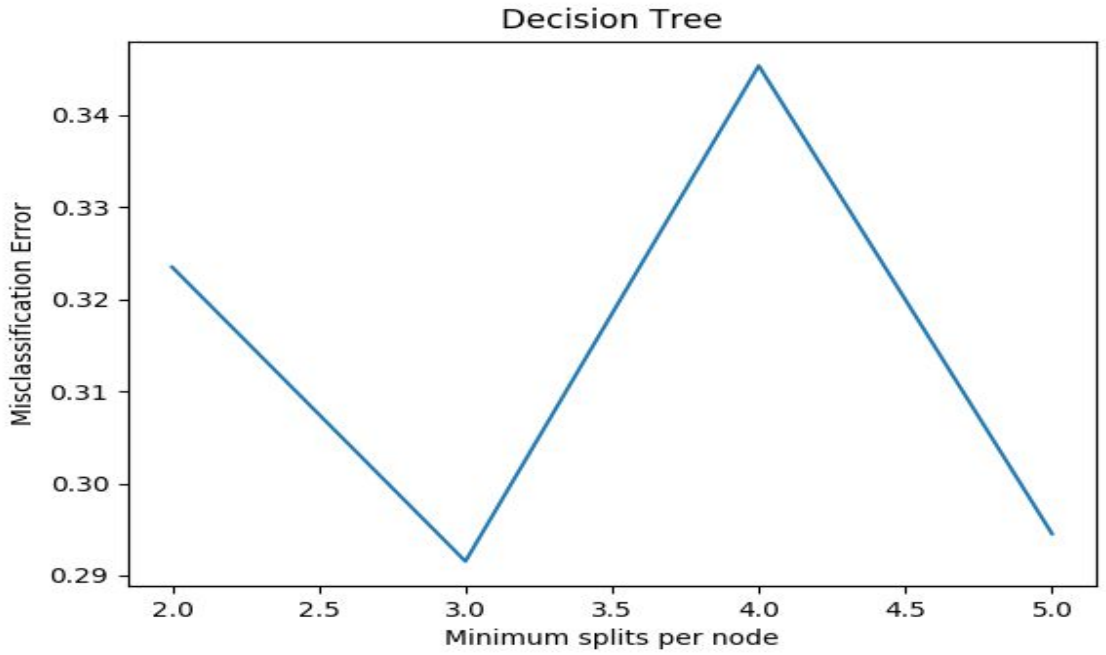

Fig. 3.5: Misclassification error for different value of number of splits per node.

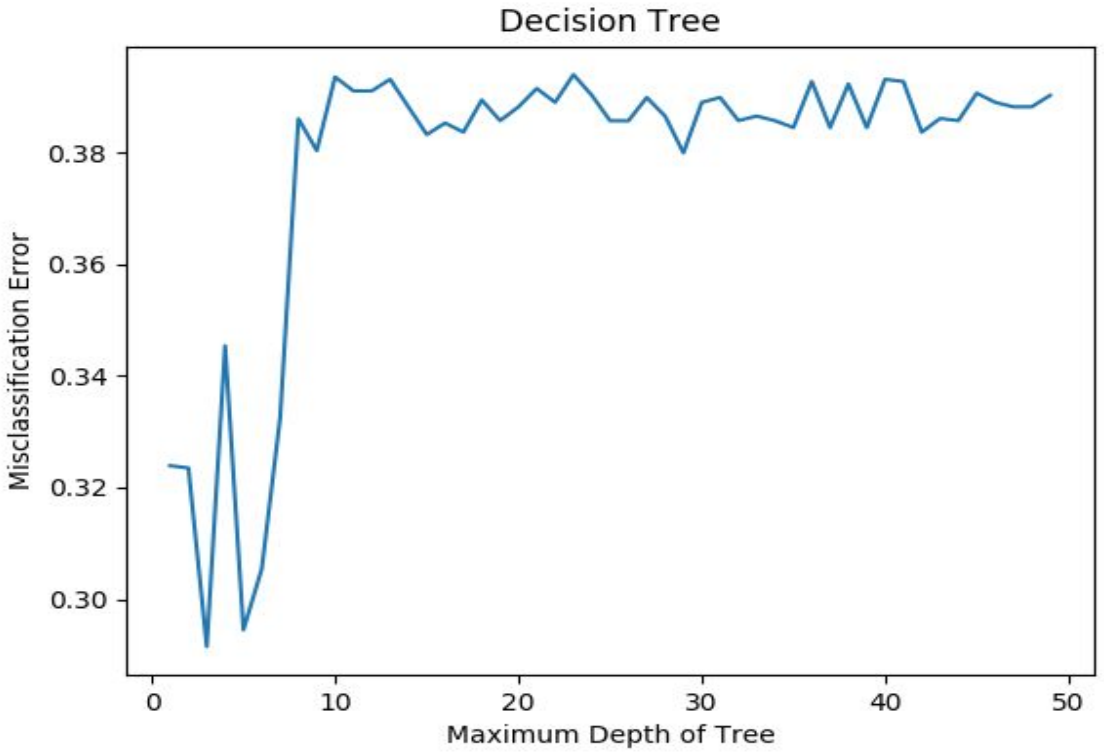

Fig 3.6: Misclassification error for different value of maximum depth of tree.

3.3.5 Random Forest

Random Forest (RF) builds *k* decision tree classifiers from the training dataset by training each classifier on a sample subset of the data. During prediction, each classifier votes and the majority class becomes the class of the new data point we want to classify. However, in the scikit-learn implementation, the average of the probabilistic predictions of all the classifiers is used to determine the class label of the new data point (Scikit-learn, 2011). The value of *k* is specified as number of estimators in the *RandomForestClassifier* class from scikit-learn and is set manually. After training, a value of 20 for the number of estimators produces the least misclassification error on the training set.

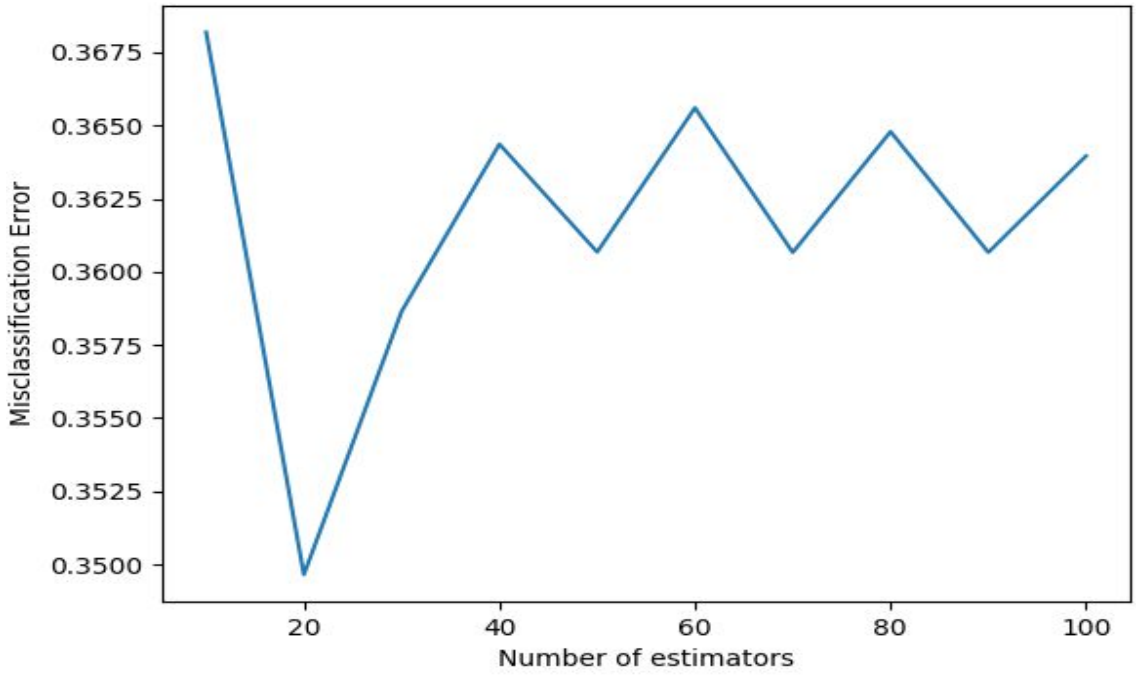

Fig. 3.7: Misclassification error for different values of number of trees in the forest

The table below gives a summary of the performance of each algorithm. Among other things, the table defines the mean score and standard deviation for each algorithm.

From the data, LR has the highest performance with a 0.83 score and +/- 0.09 standard deviation.

Table 3.2: 10 fold cross-validation training scores

ALGORITHM	PARAMETERS	MEAN SCORE	STD DEV	BEST PARAM VALS
kNN	num of neighbours weights	0.82	+/- 0.12	N_neighbors :19 weights: distance
Logistic Regression	C	0.83	+/- 0.09	C : 1
Decision Trees	max_depth, criterion,min_samples_split	0.78	+/- 0.11	Max_depth: 3 criterion: entropy Min_samples_split : 3
SVM	C, decision_function, degree	0.83	+/- 0.12	C : 2 Decision_function_shape: ovo Degree : 2
Random Forest	n_estimators	0.80	+/- 0.10	n_estimators: 20

CHAPTER 4: Testing And Results

After training, the algorithms were tested to evaluate how well each will perform on new examples. For each algorithm, the parameter values that yielded the best accuracy score were used when evaluating performance on the new examples. The tables below gives a summary of the performance of the various algorithms on the new test examples. For each algorithm, we provide its confusion matrix, the overall accuracy score, precision and recall when the z-axis features were used and when the dot-product features were used.

4.1 Summary of Test Results For Z-axis and Dot-Product Features

Logistic Regression (C = 1)

Table 5.1: **Z-AXIS**

	Predicted		
	Bad	**Fair**	**Good**
Bad	0.696	0.1287	0.175
Fair	0.176	0.014	0.8080
Good	0.078	0.0018	0.92

Accuracy	**Precision**	**Recall**
0.6583	0.4991	0.5280

Table 5.2: **DOT PRODUCT**

	Predicted		
	Bad	**Fair**	**Good**
Bad	0.6490	0	0.3509
Fair	0.1365	0.0097	0.8536
Good	0.0296	0.0148	0.9554

Accuracy	**Precision**	**Recall**
0.6816	0.5545	0.5380

Decision Trees (max depth=3, minimum sample split=3)

Table 5.3: **Z-AXIS**

	Predicted		
	Bad	**Fair**	**Good**
Bad	0.9702	0	0.0297
Fair	0.0784	0.8186	0.1029
Good	0.0613	0.0074	0.9312

Accuracy	**Precision**	**Recall**
0.9205	0.9257	0.9067

Table 5.4: **DOT PRODUCT**

	Predicted		
	Bad	**Fair**	**Good**
Bad	0.8112	0.0463	0.1423
Fair	0.3073	0.0195	0.6731
Good	0.1243	0.0333	0.8423

Accuracy	**Precision**	**Recall**
0.6567	0.4425	0.5228

Random Forests (number of estimators = 20)

Table 5.5: **Z-AXIS**

	Predicted		
	Bad	**Fair**	**Good**
Bad	0.6435	0.	0.3311
Fair	0.0343	0.6127	0.3529
Good	0.0204	0.0037	0.9758

Accuracy	**Precision**	**Recall**
0.8086	0.7834	0.7440

Table 5.6: **DOT PRODUCT**

	Predicted		
	Bad	**Fair**	**Good**
Bad	0.8940	0.0430	0.0629
Fair	0.2195	0.0829	0.6975
Good	0.0983	0.0278	0.8738

Accuracy	**Precision**	**Recall**
0.7246	0.6185	0.6169

kNN (k = 19, weights = distance)

Table 5.7: **Z-AXIS**

	Predicted		
	Bad	**Fair**	**Good**
Bad	0.6468	0.2376	0.1155
Fair	0.2549	0.1078	0.6372
Good	0.1078	0.0334	0.8587

Accuracy	**Precision**	**Recall**
0.6507	0.5245	0.5378

Table 5.8: **DOT PRODUCT**

	Predicted		
	Bad	**Fair**	**Good**
Bad	0.8013	0.0496	0.1490
Fair	0.2731	0.0243	0.7024
Good	0.0927	0.0185	0.8886

Accuracy	**Precision**	**Recall**
0.694072657744	0.571465831013	0.571465831013

SVM (C=2, degree = 2, decision function = one-vs-one)

Table 5.9: **Z-AXIS**

	Predicted		
	Bad	**Fair**	**Good**
Bad	0.795	0.059	0.145
Fair	0.23	0	0.769
Good	0.083	0	0.91

Accuracy	**Precision**	**Recall**
0.7023	0.4780	0.5705

Table 5.10: **DOT PRODUCT**

	Predicted		
	Bad	**Fair**	**Good**
Bad	0.8112	0	0.1887
Fair	0.1902	0	0.8097
Good	0.0834	0	0.9165

Accuracy	**Precision**	**Recall**
0.7065	0.4778	0.5759

4.2 Discussion

From the results above, decision trees, with a tree depth of 3 and minimum node split of 3 is the best performing model overall with true positives of 97% accuracy for bad roads, 81% accuracy for fair roads and 93% accuracy for good roads. The overall accuracy on the test set is 92% with a precision of 92% and recall of 90%. This means that, this model is more likely to accurately predict a new data point as belonging to its true class. These results apply for the z-axis features. In the instance where the dot-product features were used, the model produced true positives of 81% for bad roads, 0.02% for fair roads and 84% for good roads. From the table, the model mostly predicts fair roads as good roads while it is more likely to predict bad and good roads accurately. The other models (LR, SVM, RF, kNN) produced a pattern for both the z-axis and dot-product features where they had high true positives for bad and good roads, but were more likely to predict the fair roads as good roads. For example, the

SVM model had high true positives for bad (80%) and good (91%) roads but is more likely (76%) to classify fair roads as good roads when using the z-axis features. It has a accuracy of 70% with only 47% precision and 57% recall. In the case of the dot-product features, we have true positives of 81% bad road and 91% good road and 0% for fair roads. The model is more likely to predict fair roads as good road (80%) and then bad road (20%).

For the kNN model, z-axis features produces true positives of 64%, 10% and 85% for bad, fair and good roads respectively. Again this model is mostly classifying fair roads as good 63% of the time. It has 65% accuracy with 52% precision and 53% recall. For the dot-product features, this model produces true positives of 80%, 0.02% and 88% for bad, fair and good roads respectively. The model has an accuracy of 69% with 57% precision and recall while also more likely to predict fair roads as good 70% of the time. The RF model (z-axis features) performs with an accuracy of 83%, 81% precision and 77% recall. It however, fails to accurately predict any of the fair road instances and rather predicts them as good roads. We see similar result for the dot-product features as well.

From the analysis, we see the decision tree model is more likely to accurately classify each type of road with high precision whereas the other fail to classify fair roads based on the current data.

CHAPTER 5 : Conclusion And Recommendations

5.1 Summary

In evaluating the performance of and choosing an algorithm that will produce the highest accuracy score on test examples, we used 10-fold cross-validation training while tuning some parameters of the models. The algorithms yielded various performance results for different parameters as shown in the "**10 fold cross-validation training scores"** table above. After training and testing, decision tree had the highest accuracy for classifying data belong to "good", "bad" and "fair" roads. All the other algorithms (LR, kNN, RF and SVM) had similar performance. They are more likely to accurately classify data belonging to "good" and "bad" roads but fail to accurately classify data from "fair" roads. Instead data from "fair" roads are more likely to be classified as belonging to "good" road. A summary of the performance of all the algorithms and further discussion can be seen the in section 4 above. Looking at this, Decision Tree is the clearcut winner based on the data available as it is more likely to accurately classify data belong to each class of road.

5.2 Limitations

The similarity among the other models (classifying "fair" road data as "good") is due to a lack of a clear distinction between the data especially the fair and good road data. One way to improve performance will be to precisely define what differentiates a "fair" road from a "good" one as this could help distinguish the classes. Another problem this project could not address is the issue of participants having to place their phone in fixed positions as was done

for data collection. As mentioned earlier, this will be inconvenient for participants and it will be more appropriate if a model can be trained that can accurately classify data gathered in instances where the phone is not fixed to the dashboard or armrest of the vehicle.

5.3 Future Work

There is a need to evaluate the performance of other algorithms not captured in this work. A good example will be Artificial Neural Networks (ANN). ANN can learn features from the raw data that best describe the data instead of the manual feature extraction utilized in the work. This will however require having a good understanding of how to set up the ANN's architecture and what parameters to tweak for improved performance. In addition, if the classes of interest are distinct enough, then applying a clustering algorithm such as K-means to the raw data could yield a better performance since the we would expect that data points belonging to the same class will be much closer to each other than data points belonging to different classes. Another options for improving performance could be to try out various performance metrics with data that is clearly separable to identify the metric that is ideal for this particular situation. In addition, trying out a different set of features that define the data properly could also improve performance. An option could also be to create more features by generating quadratic features from the existing data.

References

Adankon, M. M., & Cheriet, M. (2009). Support vector machine. In *Encyclopedia of biometrics* (pp. 1303-1308). Springer US.

Boohene, K. G. (2017, April). Automated Collection And Visualization Of Road Quality Data To Aid Driver Navigation. Ashesi University.

Caruana, R., & Niculescu-Mizil, A. (2006, June). An empirical comparison of supervised learning algorithms. In Proceedings of the 23rd international conference on Machine learning (pp. 161-168). ACM.

Doku, A. G. (2014, April). Embedding Information About Road Surface Quality Into Google Maps To Improve Navigation. Ashesi University

Eriksson, J., Girod, L., Hull, B., Newton, R., Madden, S., & Balakrishnan, H. (2008, June). The pothole patrol: using a mobile sensor network for road surface monitoring. In *Proceedings of the 6th international conference on Mobile systems, applications, and services* (pp. 29-39). ACM.

Hoffmann, M., Mock, M., & May, M. (2013, August). Road-quality classification and bump detection with bicycle-mounted smartphones. In Proceedings of the 3rd International Conference on Ubiquitous Data Mining-Volume 1088 (pp. 39-43). CEUR-WS. Org.

Kanhere, S. S. (2011, June). Participatory sensing: Crowdsourcing data from mobile smartphones in urban spaces. In *Mobile Data Management (MDM), 2011 12th IEEE International Conference on* (Vol. 2, pp. 3-6). IEEE.

Scikit-learn: Machine Learning in Python,Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.

Strazdins, G., Mednis, A., Kanonirs, G., Zviedris, R., & Selavo, L. (2011, April). Towards vehicular sensor networks with android smartphones for road surface monitoring. In *2nd international workshop on networks of cooperating objects, Chicago, USA*.

Tilak, S. (2013). Real-world deployments of participatory sensing applications: Current trends and future directions. *ISRN Sensor Networks*, *2013*.

Vorgbe, F. D. (2014, April). Classification Of Road Surface Quality Using Android Smartphone Devices. Ashesi University.

Vittorio, A., Rosolino, V., Teresa, I., Vittoria, C. M., & Vincenzo, P. G. (2014). Automated sensing system for monitoring of road surface quality by mobile devices. *Procedia-Social and Behavioral Sciences*, *111*, 242-251.