Department of Computer Science

Submitted in part fulfilment for the degree of MEng

Travel Mode Identification based on GPS Trajectory Data

Harry Erskine

06 May 2024

Supervisor: James Yu

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

Part I

Executive Summary

Executive Summary

The aim of this project was to investigate machine learning methods that can be used in conjunction with labelled GPS data to determine the travel mode of a person. After extracting and filtering the GPS data 5 different travel modes present, those being: walk, cycle, car, bus, and train.

The machine learning models selected for this project were decision trees, support vector machines and convoluted neural networks. These 3 techniques all provide differing approaches to machine learning allowing for a wide range of techniques to be used, and for a higher likelihood of a model that produces strong results for this particular classification task to be found.

In this investigation, different versions of the dataset were used to test how well each model performed when there were inconsistencies in the data. It was found that the decision tree and convoluted neural network were able to achieve the highest accuracies, obtaining up to 81.1% and 82.6% respectively.

Part II

Main Body

1 Introduction

1.1 Motivation

The availability of GPS technology in society presents unique opportunities for data-driven decision-making across various sectors. This project is motivated by the potential to harness GPS data to determine the transportation modes of individuals, a capability that can significantly benefit urban planning and business strategy. The integration of GPS functionality in smartphones and other devices, evident with approximately 84% of the population in developed regions owning a smartphone as of 2023 [1], underscores the practicality and accessibility of this research.

Moreover, the cost-effectiveness of acquiring GPS data enhances its appeal as a resource for large-scale data analysis, enabling the development of models that interpret vast quantities of locational information to identify patterns and trends which would not be immediately obvious.

However, the widespread use of GPS technology also necessitates a careful consideration of privacy issues. It is important to acknowledge the need for individual consent and to implement ethical data collection practices that protect user privacy [2].

Labelled GPS data has diverse applications, notably in city planning where it can inform the development of more efficient public transportation systems and infrastructure. Beyond urban planning, businesses are increasingly keen to understand the travel patterns of their customers. For instance, retail companies may use this data to determine the most frequented routes that lead customers to their stores, thereby optimizing their location strategies.

1.2 Objectives

The objective of this project is to evaluate the most effective ways to identify a person's travel mode based on their GPS data. This project involves studying, creating, and analysing 3 machine learning (ML) algorithms, those being:

- Decision Tree (DT)
- Support Vector Machine (SVM)
- Convoluted Neural Network (CNN)

Where these algorithms will receive a training dataset to be used to train their models. Once the models are trained, a testing dataset will

be passed through it to determine their accuracy against data that they were not trained on.

Multiple variations of both the ML algorithms and the dataset will be used to discover the methods that provide the highest accuracy for different instances of the dataset. This project will not be proposing new state of the art ML techniques, but will look to give a good evaluation of which current techniques work best, for travel mode identification, as well looking at how different forms of training / testing data fair on each of the models.

2 Literature Review

The user guide [3] of the Geolife dataset holds a lot of useful information about the dataset including how the data is formatted using PLT files, which then contains fields for: latitude, longitude, altitude, date, and time. It also contains a description of how the labels file is formatted as well as some additional information which may be useful to consider such as describing how most of the data was created in Beijing and showing graphs of the distribution of the travel modes used, the distance and time across all the users' logs, the collection periods of users, and the number of trajectories logged by users.

Some important considerations to make are how will ML algorithms use pre-processed data and in which format should it be in to produce strong and reliable results. In the 'Inferring transportation modes from GPS trajectories using a convolutional neural network' paper [4], they show methods for identifying travel modes based on the same dataset. In their method they first split GPS data into trips if there is too long of a time gap between consecutive points. Then each trip is further divided into segments where each segment only contains one label (transport mode). Then these segments are sub-divided once more into a pre-defined number of GPS points (and padded with zero values when needed).

In that paper [4], once all the data is formatted into equal sized segments, they then calculated the motion characteristics of each point using the tuple (latitude, longitude, time). These characteristics were: change in distance (using Vincenty's formula [5]), change in time, speed, acceleration, jerk, and bearing rate.

2.1 Investigation of Data Pre-processing Methods

Effective data pre-processing is a critical step in the development of machine learning models, particularly when dealing with large and complex datasets such as the Geolife dataset.

Google Colaboratory [6], offers a cloud-based platform that provides a robust and flexible environment for machine learning and data analysis. It supports Python code execution and is integrated with Google Drive, making it accessible and convenient for collaborative projects.

At the core of data pre-processing in Python is the pandas library [7], which features DataFrame objects for storing and manipulating structured data in a tabular form. DataFrames allow for efficient data handling and are particularly adept at tasks such as filtering, grouping,

and transforming data — operations that are essential in the initial stages of analysing GPS trajectory data.

Additionally, Python, Google Colab, and pandas are all widely used for creating machine learing algorithms. This is preferable as it will keep the code base and style consistent between pre-processing and machine learning sides or this project.

2.2 Investigation of Decision Trees

Decision trees as a technique were first implemented in the 1963 AID (Automatic Interaction Detector) project by Morgan and Sonquist [8]. This method saw significant advancements with the introduction of the ID3 (Iterative Dichotomiser 3) algorithm by Quinlan in 1986 [9]. His work laid the foundation for the C4.5 algorithm [10] which further refined decision tree construction by addressing some of the limitations of ID3.

Modern decision trees are powerful tools used for both classification and regression tasks in machine learning. The process begins with the entire dataset at the root of the tree. The dataset is then recursively split according to a specific criterion that measures the "best" way to separate the samples based on attribute values. A common criterion is ‘enthropy’, but others like ‘Gini impurity’ are also widely used.

In each node, the dataset is split on the attribute that results in the largest information gain. The goal is to have each branch eventually leading to subsets that are as pure as possible. To calculate information gain, the node will need to determine the entropy of the elements it holds.

Entropy is a measure of the unpredictability in the data, this is used to quantify the impurity within the dataset, entropy’s formula is given by:

$$Entropy(S) = -\sum_{i=0}^{c} \mathrm{P_i} \log_2(P_\mathrm{i})$$

Where:

- *S is the set of elements that have been passed into the node*
- P_i is the proportion of the elements in the set S that belong to class $'i'$
- *c* is the total number of different classes in set S

Information Gain is calculated as the reduction in entropy resulting from splitting the data, therefore it measures how effectively an

attribute separates the training examples into their target classes, information gain's formula is given by:

$$\text{Infomation Gain}(\text{S}, \text{A}) = Entropy(S) - \sum_{t \in T} \frac{||S_t||}{||S||} Entropy(S_t)$$

Where:

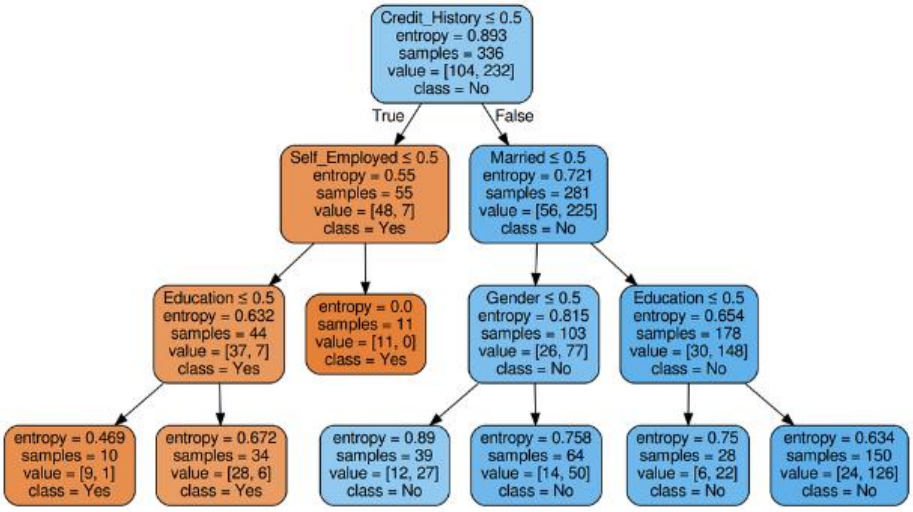
- A is the attribute (feature) used to split S
- T represents the 2 subsets created from splitting S by A
- S_t represents the elements in subset 't'

The attribute A could be any Boolean statement such as 'X ≥ Y' where 'X' is the name of a variable (e.g. weight) and 'Y' is the value for variable X (e.g. 67.89 kg) that is held by one of the elements in the subset S. The node on the DT will look to find the attribute A that will incur the greatest information gain and then create 2 sub-nodes, passing one of the two S_t subsets (which were split by A into each sub-node.

When a node has an entropy of zero or has reached the tree's max depth, then it becomes a leaf / terminal node and it will determine what class an element is based on what was the most prominent class at that node when it was trained.

It would be computationally expensive to determine the information gain for all possible attributes A, so many decision tree algorithms use a quantile-heuristic based approach to determine the best A values, in order to reduce computational time / costs.

Figure 1: Example of a Decision Tree (to classify loan eligibilty)



Because of their tree structure DTs are very interpretable compared to other ML algorithms and can therefore be visualised and understood by people without a background in ML. They are also capable of handling both numerical and categorical data, meaning they often have more access to data than other algorithms and also will not need to scale the data beforehand.

However, DTs have instability because small changes in their training data can lead to a vastly different tree being generated. They are prone to overfitting, which often leads to a high loss in accuracy compared to the data it was trained on. DTs are also very biased to classes that represent a comparatively high proportion of the data. This issue is often resolved by balancing the training dataset.

Due to these qualities, DTs are often used in areas such as credit scoring, medical diagnosis, and customer segmentation (for marketing purposes).

2.3 Investigation of Support Vector Machines

Support Vector Machines (SVMs) were first developed by Vladimir Vapnik and Alexey Chervonenkisin the 1963 [11]. The classification-regression method of SVMs was later introduced in the 1995 [12].

The objective of an SVM's algorithm is to find the hyperplane (which is line in n-dimensional space that best divides its training dataset into classes. The optimal hyperplane is the one that maximizes the margin (separation) between the closest data points of 2 classes, these points are known as the support vectors. The SVM constructs a hyperplane mathematically using the decision function, which is defined by:

$$\omega * x + b = 0$$

Where:

- ω is the weight vector
- x is the feature vector
- b is the bias

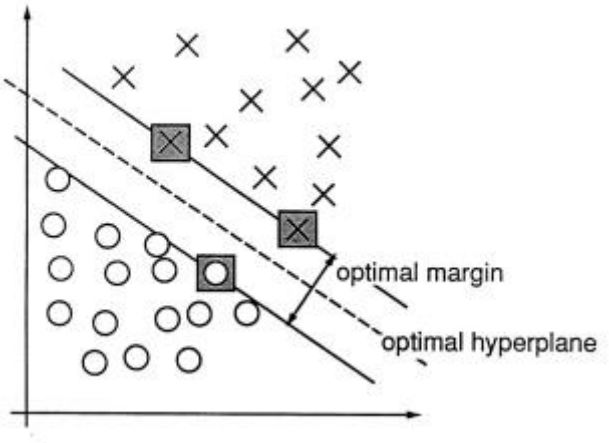
The following formula is used to classify whether a data point is being classified correctly:

$$y_{\mathrm{i}} * (\omega * x_{\mathrm{i}} + b) \geq 1, \quad for\ all\ i$$

Where:

- y_{i} is the classification of the data point (either -1 or +1)

Figure 2: Example of SVM on Linearly Separable Data



SVMs can use the kernel trick to transform the input space into a higher dimensional space when a linear separator is insufficient. They use functions such as the radial basis function (RBF) and polynomial to map the data into a higher-dimensional space [13].

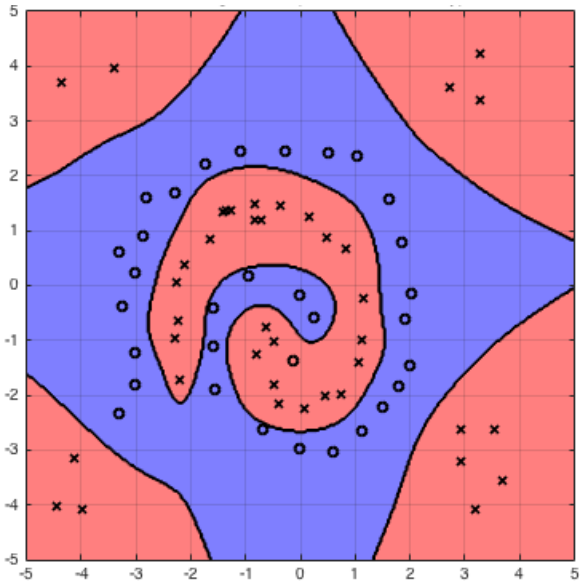
An RBF kernel uses the kernel (K) function:

$$K(x_1, x_2) = exp(-\gamma||x_1 - x_2||^2)$$

Where:

- x_1 and x_2 are feature vectors of opposing classes
- γ (gamma) is a parameter that sets the spread of the kernel

Figure 3: Example of an SVM using a Radial Basis Function



A polynomial kernel (K) function is denoted as:

$$K(x_{1,2}) = (\gamma * x_1{}^{\mathrm{T}} x_2 + r)^{\mathrm{d}}$$

Where:

- γ (gamma) is a parameter that sets the spread of the kernel
- r is a coefficient that is used to control the kernel's offset
- d denotes the degree, where higher degrees allow for the hyperplane to make curves in higher dimensional space

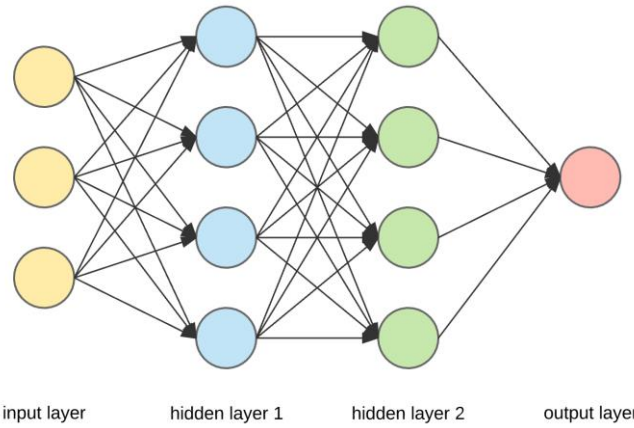
When using kernel functions, SVMs are versatile in handling various types of data and can be effectively used in pattern recognition [14]

2.4 Investigation of Convoluted Neural Networks

The concept of Convolutional Neural Networks (CNNs) was pioneered by Kunihiko Fukushima in 1980 with the introduction of the Neocognitron [15], a neural network model designed to recognize visual patterns such as handwritten characters. This early model laid the foundational principles for convolutional processing and sub-sampling layers that simulate the hierarchical processing approach of the human visual system. Later advancements by Yann LeCun et al., expanded on Fukushima's work through the development of LeNet-5 [16], a significant evolution that effectively applied backpropagation to train CNNs for practical applications like digit recognition.

CNNs are structured as many layers of interconnected nodes (neurons) where data is fed into the input layer, then there are multiple hidden layers which each apply functions to the data in the previous layer, then a final function is applied in the last hidden layer to produce the output layer, which would be a single value that corresponds to one of the labels the model is classifying.

Figure 4: Layers of a CNN



CNNs achieve high accuracies in image recognition tasks because of their unique ability to recognise small features in images, such as edges and curves, which can then be built up until the CNN is classifying objects. Features are extracted using multiple layers. First, convolutional layers use a filter (matrix of weights) to slide across an image and apply a function to the RGB (red-green-blue) values of each pixel in an image.

Figure 5: Input Image and Weight Matrix

INPUT IMAGE

18	54	51	239	244	188
55	121	75	78	95	88
35	24	204	113	109	221
3	154	104	235	25	130
15	253	225	159	78	233
68	85	180	214	245	0

WEIGHT

1	0	1
0	1	0
1	0	1

429

$$\begin{bmatrix} 18 & 54 & 51 \\ 55 & 121 & 75 \\ 35 & 24 & 204 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 18 & 0 & 51 \\ 0 & 121 & 0 \\ 35 & 0 & 204 \end{bmatrix} \Rightarrow$$

$$\Rightarrow 18 + 51 + 121 + 35 + 204 \Rightarrow 429$$

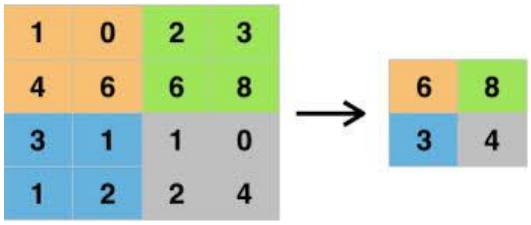
Using figure 5 as an example, the values on the input image are multiplied by their corresponding value on the weight matrix and then these values are added together to give the value at that point in the next layer. The weight matrix slides horizontally across the input image for every row of pixels, leaving behind a feature map.

An activation function is then applied to the convolutional layers by adding a bias to it. A popular activation function is ReLu (rectified linear unit), this simply applies the function:

$$ReLu(x) = \max(0, x)$$

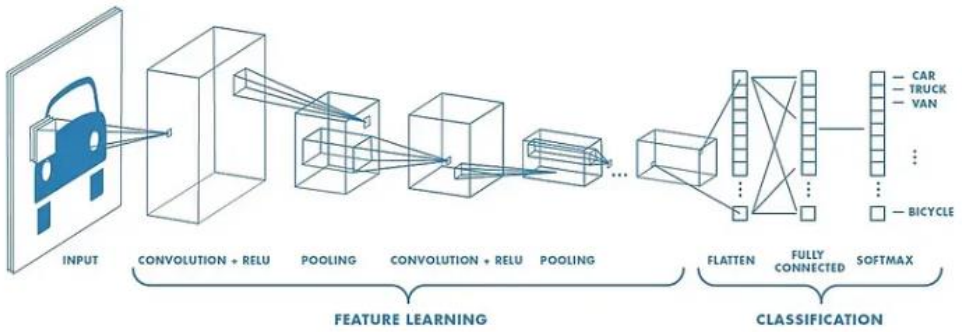
Next a max-pooling layer is often applied, this is similar to the convolutional layer, as a filter slides over the feature map in the same manner. However, it will instead return the maximum value found in the feature map.

Figure 6: Example of Max Pooling



These layers are often repeated in order to break the original input image down into multiple smaller feature maps so that there are more identifiable features, making the image easier to classify.

Figure 7: Neural Network with many Convolutional Layers



After the features have been learned, they are flattened into one dimension and the combination of a fully connected layer and soft-max layer is used to classify the image.

3 Methodology

For this project Google Collaboratory was used for the pre-processing of data and the training of the ML models as well as using pandas DataFrames to organise data, pre-process data, and pass data into the models.

3.1 Pre-processing of Data

The raw data from the Geolife dataset is structured as multiple user files ('000' to '182') where only 69 of those users had a labels file, which contained all of their labels (transport mode) and their corresponding start and end times. Each user also had numerous files for their GPS data which contained fields for Longitude, latitude, altitude, date (as a float), date (as a string), and time (as a string).

This data was then organised into 2 data frames where unlabelled users' data was discarded. These 2 data frames were then combined so that if a trajectory fell within the time period for one of the labels from its user, then that trajectory would be given its corresponding label (Mode).

Table 1: Travel Modes and Trajectories Counts

Mode	Count	Mode	Count	Mode	Count
Train	556343	Taxi	241056	Walk	1579984
Bus	1269700	Subway	309226	Airplane	9183
Car	512276	Bike	950033	Boat	3559
Run	1971	Motorcycle	338	Total	5433669

There was clearly little data with the mode 'motorcycle' so those trajectories were disregarded as it would be difficult for the ML algorithms to accurately classify that mode.

There were also 55461 trajectories that held '-777' as their value for altitude (this was a placeholder for when it wasn't recorded). These trajectories were also disregarded. The data was also thoroughly checked for impossible values, which resulted in a single trajectory (with a latitude 400.17 degrees) to be removed.

The attribute for altitude was also set in feet, so that was changed to meters in order to maintain metric values for the data.

$$Altitude\ (meters) = Altitude\ (feet) * 0.3048$$

Next, the modes were revised so that ‘taxi’ was re-classed under ‘car’, ‘subway’ was re-classed under ‘train’ and ‘run’ was re-classed under ‘walk’. With these changes, 1.027% of the data has been disregarded.

Table 2: Revised Travel Modes and Trajectory Counts

Mode	Count		Mode	Count
Train	814994		Walk	1581802
Bus	1269481		Airplane	9183
Car	748817		Boat	3559
Bike	950033		Total	5377869

From here the trajectories were organised into groups of originally 16 consecutive trajectories, however this was also done for groups of 8, 24, 32, and 64 trajectories (where these groups would also undergo any of the following pre-processing steps).

Groups containing trajectories of differing modes or users were disregarded as well as any groups containing consecutive trajectories with a time difference of over 6 seconds.

The distance between 2 consecutive GPS points is calculated by passing the latitudes and longitudes of the 2 points into Vincenty’s formula [5]. Using the distance and time ‘t’ between 2 positions (p1 and p2) the speed ‘S’, acceleration ‘A’, and jerk ‘J’ can be calculated as follows:

$$S_{\mathrm{p1}} = \frac{Vincenty(P_1,\ P_2)}{\Delta t}$$

$$A_{\mathrm{p1}} = \frac{S_{\mathrm{p2}} - S_{\mathrm{p1}}}{\Delta t}$$

$$J_{\mathrm{p1}} = \frac{\mathrm{A_{p2}} - \mathrm{A_{p1}}}{\Delta \mathrm{t}}$$

This project will also utilise the altitude features in the dataset to determine the altitude-based velocity ‘AV’ and altitude-based acceleration ‘AA’ of consecutive positions, using the similar equations:

$$AV_1 = \frac{\mathrm{Altitude_2} - \mathrm{Altitude_1}}{\Delta \mathrm{t}}$$

$$AA_2 = \frac{\mathrm{AV_2} - \mathrm{AV_1}}{\Delta \mathrm{t}}$$

Next, the bearing rate ‘BR’ is calculated using these calculations:

$$y = sine[P_2(long) - P_1(long)] * cosine[P_2(lat)]$$

$$x = \ cosine\ [P_1\ (lat)] * sine\ [P_2\ (lat)] - sine\ [P_1\ (lat)] * cosine\ [P_2\ (lat)] * cosine\ [P_2\ (long) - P_1\ (long)]$$

$$Bearing(P_1) \ = \ arctan(y, x)$$

$$BR_{\mathrm{p1}} = |Bearing_{\mathrm{p2}} - Bearing_{\mathrm{p1}}|$$

Due to the formatting of the groups in this project, the last 3 positions on each group was not calculated a value for Jerk, meaning the group size will be reduced by those 3 trajectories as they are disregarded.

As individual points of motion data may be hard for a decision tree or support vector machine to achieve high accuracies, 2 additional sections of data will be introduced.

The first being 'derived motion features' which will be the minimum 'Min', 25th quartile '25th Q', medium 'Med', mean, 75th quartile '75th Q', maximum 'Max', range 'Range', and interquartile range 'IQR' for all of the motion features (speed 'S', acceleration 'A', jerk 'J', altitude-based velocity 'AV', altitude-based acceleration 'AA', and bearing rate 'BR') over each pre-defined group of trajectories.

Table 3: Table of all Derived Motion Features

S Min	A Min	Jerk Min	AV Min	AA Min	BR Min
S 25th Q	A 25th Q	J 25th Q	AV 25th Q	AA 25th Q	BR 25th Q
S Med	A Med	J Med	AV Med	AA Med	BR Med
S Mean	A Mean	J Mean	AV Mean	AA Mean	BR Mean
S 75th Q	A 75th Q	J 75th Q	AV 75th Q	AA 75th Q	BR 75th Q
S Max	A Max	Jerk Max	AV Max	AA Max	BR Max
S Range	A Range	J Range	AV Range	AA Range	BR Range
S IQR	A IQR	J IQR	AV IQR	AA IQR	BR IQR

The second being 'range counts', this feature involves counting the number of times each motion feature fell between a pre-determined range (see table 3). Because the bearing rate is rotational data (ranging from -π radians to π radians), the range counts for it are based on its absolute value (so that the range count will be more precise data on how much a person's turning, rather than focusing on which direction they are turning).

Table 4: Table of Specified Range Counts

Speed	Accel'	Jerk	AV	AA	BR
S < 0.25	A < -5.0	J < -5.0	AV < -2.0	AA < -2.0	BR < 0.125
0.25 ≤ S S < 1.0	-5.0 ≤ A A < -2.5	-5.0 ≤ J J < -2.5	-2.0 ≤ AV AV < -1.0	-2.0 ≤ AA AA < -0.75	0.125 ≤ BR BR < 0.25
0.0 ≤ S S < 2.5	-2.5 ≤ A < -0.5	-2.5 ≤ J J < -0.5	-1.0 ≤ AV AV < -0.25	-0.75 ≤ AA AA < -0.25	0.25 ≤ BR BR < 0.5
2.5 ≤ S S < 5.0	-0.5 ≤ A < 0.5	-0.5 ≤ J J < 0.5	-0.25 ≤ AV AV < 0.25	-0.25 ≤ AA AA < 0.25	0.5 ≤ BR BR < 1.0
5.0 ≤ S S < 10.0	0.5 ≤ A < 2.0	0.5 ≤ J J < 2.0	0.25 ≤ AV AV < 1.0	0.25 ≤ AA AA < 0.5	1.0 ≤ BR BR < 1.75
10.0 ≤ S S < 20.0	2.0 ≤ A A < 4.0	2.0 ≤ J J < 4.0	1.0 ≤ AV AV < 3.0	0.5 ≤ AA AA < 1.5	1.75 ≤ BR BR < 2.5
20.0 ≤ S	4.0 ≤ A	4.0 ≤ J	3.0 ≤ AV	1.5 ≤ AA	2.5 ≤ BR

The final step of pre-processing involves labelling groups of trajectories shall be deemed either 'Impossible' (due to significantly large motion values) or 'stationary' (roughly), these will be useful for altering the data used to train our models in order to show how well they perform when given poor data.

The condition for a group to be classed as stationary is that, if the combined movement of all of the trajectories is less than 3 meters, then that group is deemed as 'stationary'.

The conditions for a value to be deemed 'impossible' varies for each transport mode (remember that all values are in meters and seconds):

- For 'walk' - max speed > 12.0
- For 'car' - max speed > 90.0
- For 'bus' - max speed > 90.0 or
- For 'bus' - min accel' < -15.0 and max accel' > 15.0
- For 'train' - max speed > 150.0 or
- For 'train' - min accel' < -3.0 and max accel' > 3.0
- For 'bike' – max speed > 30.0 or
- For 'bike' – min accel < -10.0 and max accel' > 10.0

The reason for using opposing minimum and maximum acceleration values to classify data as impossible is because there is a likely scenario where one point of data is mis-collected, causing a sharp acceleration value and then a sharp deceleration value immediately afterwards when the data returns to where it should be.

Additionally, the groups of trajectories labelled as ‘airplane’ and ‘boat’ only account for 0.237% of groups. This statistic, coupled with the fact that they are not common modes of transport in urban environments, means that their data is not necessary to the projects aims.

Another method that will be used to alter the data before training the ML models, is to reduce the data frame to only include groups where there is always exactly 1 second between trajectories, this data would be expected to perform well due to the consistency between points, however this data frame will be reduced to 13.5% of its original size under this condition.

The cleanest form of the data will only include data that is not impossible, not stationary, not labelled airplane, not labelled boat, and will only include consecutive trajectories with a consistent 1 second interval between them. A table showing the data’s distribution is below.

Table 5: Distribution of Travel Modes

Travel Mode	Number of Groups of Trajectories (when group size = 16)	
	Cleaned groups (1s ≤ Δ time ≤ 6s)	Cleaned groups (where ‘Δ time’ = 1s, for all trajectories
Walk	66567 (26.3%)	6622 (19.5%)
Bike	41616 (16.5%)	8029 (23.6%)
Car	35612 (14.1%)	2721 (8.0%)
Bus	65333 (25.8%)	5168 (15.2%)
Train	43759 (17.3%)	11503 (33.8%)
Total	252878	34043

When passing data Into ML models a proportion of data must be retained for the model to be tested on. This is necessary as, although you can test a model against its own data, the model would have often overfitted to its training data – meaning that it would not be as accurate on new data. To split the data that will be passed into the ML models, a train-test-split function [17] has been used where it will randomly shuffle the data and split it so that 25% of the data is set aside for testing.

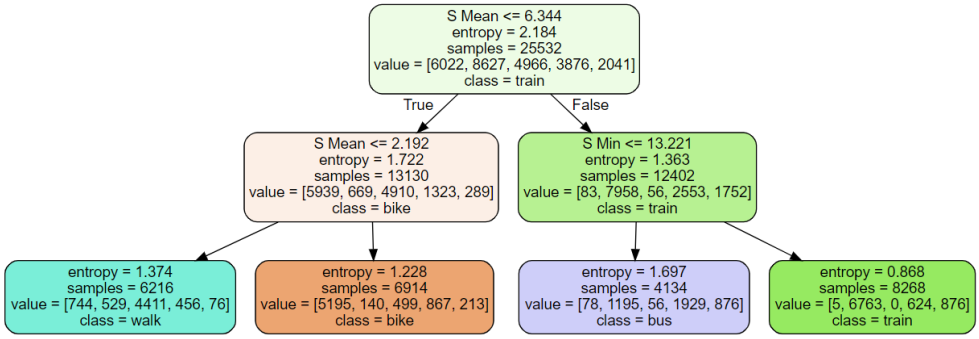
3.2 Implementation of the Decision Tree

For decision trees (DTs), this project has used a solution from Scikit Learn [18]. This implements the ‘entropy’ and ‘information gain’

calculations (as described in the literature review) to determine how to split the data at each node.

Below is a decision tree that was trained / tested on the cleanest form the data with a group size of 16 it and also had its maximum depth limited to 2.

Figure 8: Decision Tree with a Maximum Depth of 2

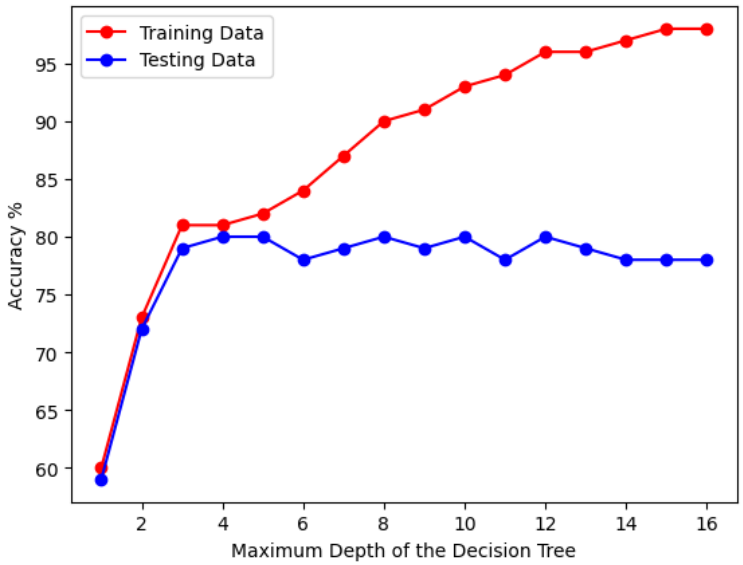


This DT was able to achieve an accuracy of 72% on the test data (despite its limitations in depth) by using:

- 'Mean Speed <= 6.344 m/s' as the condition (attribute) for its root node.
- If that condition was true, then the next condition is 'Mean Speed <= 2.192 m/s'
- If that is true, then that group of trajectories is predicted to have 'walk' as its transportation mode
- If false, then it is predicted as 'bike'
- If the root node was false, then the next condition is 'S Min <= 13.221 m/s'
- If that is true, then it is predicted 'bus'
- If false, then it is predicted 'train'

This is a strong result considering that the maximum depth can be increased to much higher amounts. However, when this was experimented it was found that the testing accuracy was only able to reach a highest accuracy of 74.6%, where the max depth was 4.

Figure 9: Accuracies of the Training and Testing Data Against the Maximum Depth of their Decision Tree



The results from the table above were achieved when the DT was able to train on derived motion features (table x) and specific range counts (table y). When these features were not used - meaning the DT was trained using the raw motion features of individual positions in each group – the DT was not as effective as it was only able to achieve a highest accuracy of 74% (which was achieved when the DT had a max depth of 6). The reason why that DT model has a drop-in accuracy is because it can only split the groups by the attributes of a single position of the data at each node. This means that it is unable to make more rounded decisions of how to split the data based on all of the positions' data for each motion. This also leads to the model overfitting early as it focuses on individual positions in the data – which have high variance compared to when averaging out the data.

The highest Accuracy achieved by a DT was 81.1% when it was trained on a fully cleaned version of the data where the group size was set to 64 and the maximum depth was set to 5. Figure 4 shows the confusion matrices for how the model performs on both the training and test data sets (also note that the ratio of training data to testing data is 3 to 1)

Figure 10: Confusion Matrices for Decision Tree of the Highest Accuracy

This model performed well because the large group size for each data entry meant that each entry covered a longer period of time – meaning entries were more well-rounded, so that there is less variance in the data for each travel mode because variations movement data for each mode would have less significance when it is averaged out over a data entry that covers a longer period.

Moreover, a max depth of 5 is relatively low and the training set also had an accuracy of 82%, meaning that the model did not overfit to the training set, causing the testing set to be just as accurate.

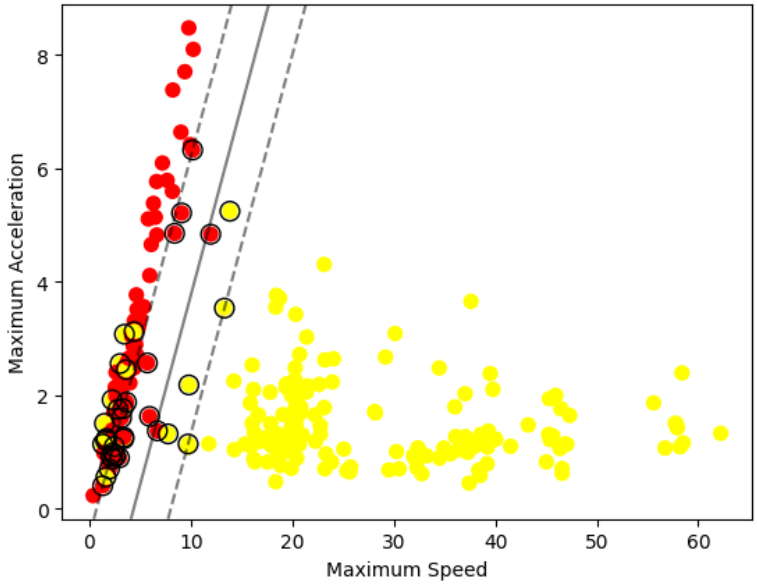
3.3 Implementation of the Support Vector Machine

For support vector machines (SVMs) this project is using solutions from Scikit Learn [19]. As described in the literature review, SVMs divide data into their classes by finding hyperplanes that best split the data when it is represented in multi-dimensional space.

Appendix A and Appendix B each show a pairplot for the data, this helps give a 2d visualisation to show how different features are linearly separable, which should help visualise why data SVMs are able to achieve high accuracy when using hyperplanes to split data.

Figure 11 shows how an SVM would use a linear separator to classify a reduced version of the data frame where there are only 2 classes – ‘walk’ (red) and ‘train’ (yellow) – and only 2 features being passed into the model – ‘maximum speed’ and ‘maximum acceleration’.

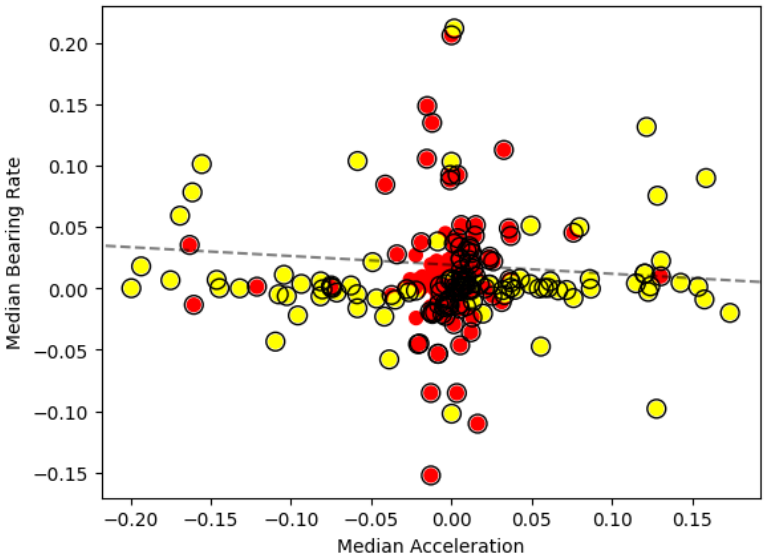
Figure 11: Linear SVM on 2D Data



That SVM achieved an accuracy of 94% (against its training data). However, the modes and attributes for this example were cherry picked to show the effectiveness of a linear SVM when the data suits that model.

The next examples will show a different reduced data frame where the 2 classes are ‘walk’ (red) and ‘bus’ (yellow), and the 2 attributes are ‘Medium acceleration’ and ‘median bearing rate’. Figure 12 shows how a linear separator would classify the data.

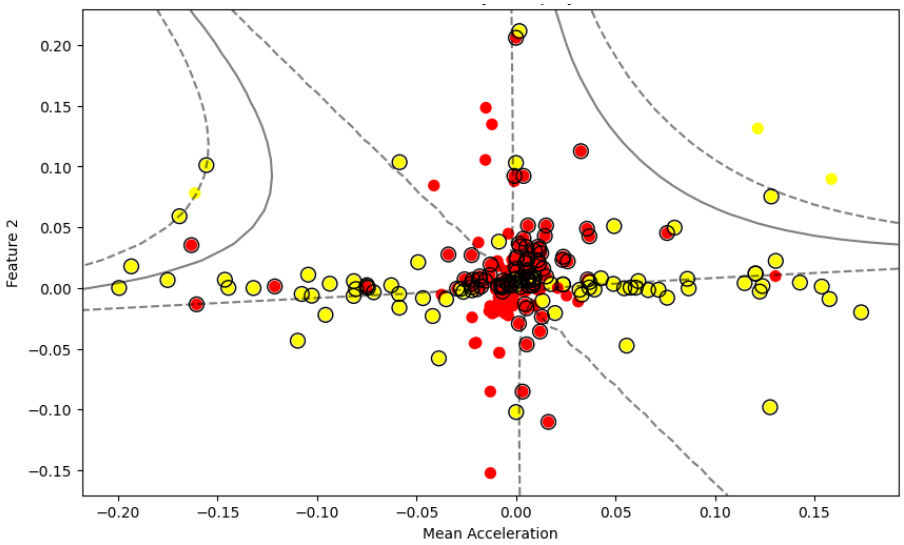
Figure 12: Linear SVM on Different 2D



The accuracy of this SVM is 66.4% (on the data it was trained on) and clearly shows the short comings of linear separators. To accurately classify data that appears in clusters, but is not linearly separable, SVMs can use the kernel trick that was described in the literature review, to find a complex way of segmenting data.

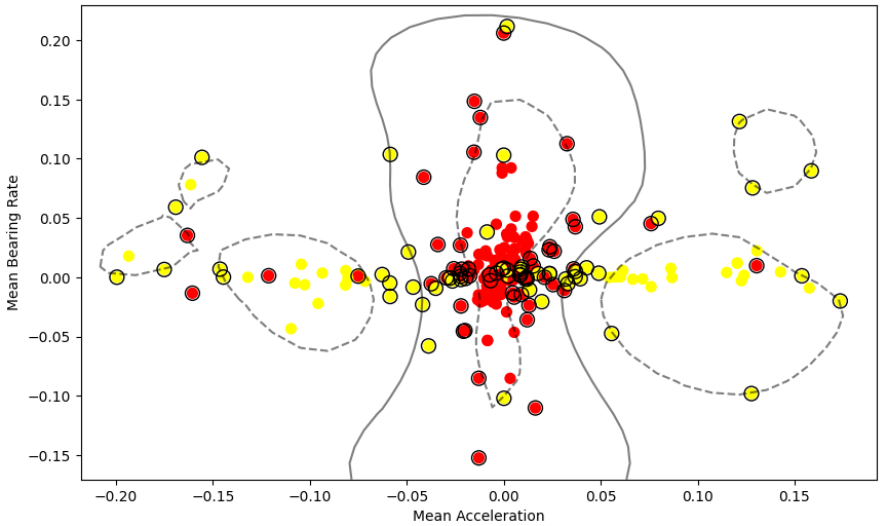
When the SVM uses a polynomial kernel (as described in the literature review) on this data (see figure #y) it is able to achieve an accuracy of 68.8% (on the data it was trained on).

Figure 13: SVM with Polynomial Kernel on 2D data



When the SVM uses an RBF kernel (as described in the literature review) on this data (see figure #z) it is able to achieve an accuracy of 86.0% (on the data it was trained on).

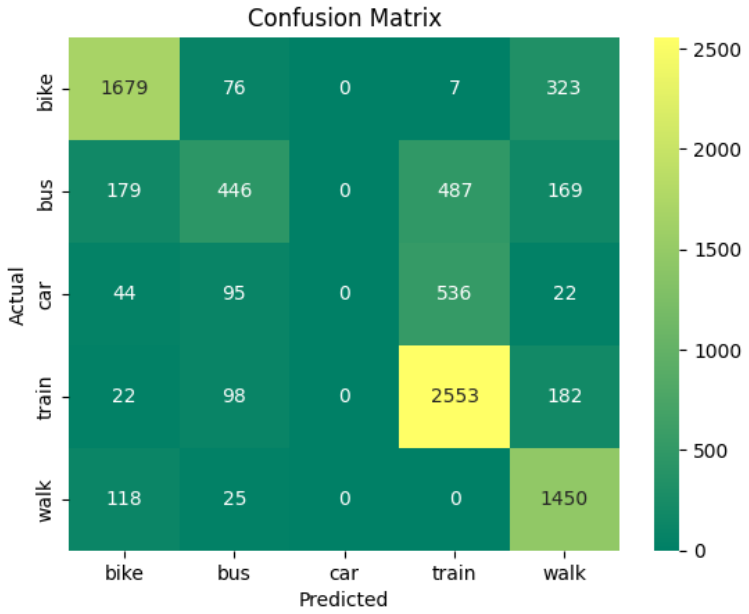
Figure 14: SVM with RBF Kernel on 2D Data



This data shows that the use of an RBF kernel is likely to be the best method to achieve a high accuracy SVM over the full data frame.

When using an RBF kernel, the SVM was able to achieve an accuracy of 72.2% on the full data frame. The following confusion matrix was also given.

Figure 15: Confusion Matrix for the SVM



3.4 Implementation of the Convoluted Neural Network

For Convoluted Neural Networks (CNNs), This project has used a simple solution that makes use of TensorFlow's library [20].

CNNs are able to extract small features and those up to make classifications. Because of this, they are usually used for image classification, however, that is not what is being done in this project. So, the first step in the implementation of this CNN is to format the data so that filters can be applied to it effectively. The way this CNN will search for filters in the data is by applying a filter to the attributes (speed, acceleration, etc) of 3 consecutive trajectories. This filter could then slide across the trajectories to create a feature map. Below is a visualisation of the structure of this data.

Table 6: Format of Data for CNN

Trajectories (T)	T1	T2	T3	T4	…	T(n)
Speed (S)	S1	S2	S3	S4	…	S(n)
Acceleration (A)	A1	A2	A3	A4	…	A(n)
Jerk (J)	J1	J2	J3	J4	…	J(n)
Alti-Velocity (AV)	AV1	AV2	AV3	AV4	…	AV(n)
Bearing Rate (BR)	BR1	BR2	BR3	BR4	…	BR(n)

In the convolutional layers, filters of dimension 5x3 (height x width) are used to create 64 feature maps, the ReLu activation function is then applied to add a bias.

This CNN also uses a global average pooling layer, a dense layer (with an ReLu activation function), and a final dense layer (with a soft-max activation function)

The full architecture and shape of the CNN is as follows:

- Convolutional layer with shape (13, 64)
- Convolutional layer with shape (13, 64)
- Convolutional layer with shape (13, 64)
- Pooling layer with shape (64)
- Dense layer with shape (64)
- Dense layer with shape (5)

This model also uses an Adam optimiser and a cross entropy loss function. During the training, the model will make predictions based on the current weights. The loss function will then calculate the error by comparing the predictions to their targets. This error is then used by the optimiser so that is adjusts the weights in order to minimise the loss over iterations and therefor improving the model's accuracy over time.

When the CNN was trained/tested on the dataset, which only included trajectories that were exactly 1 second apart, it achieved an accuracy of 77.1%.

Figure 16: Graph of CNN's Accuracy on the Smaller Dataset

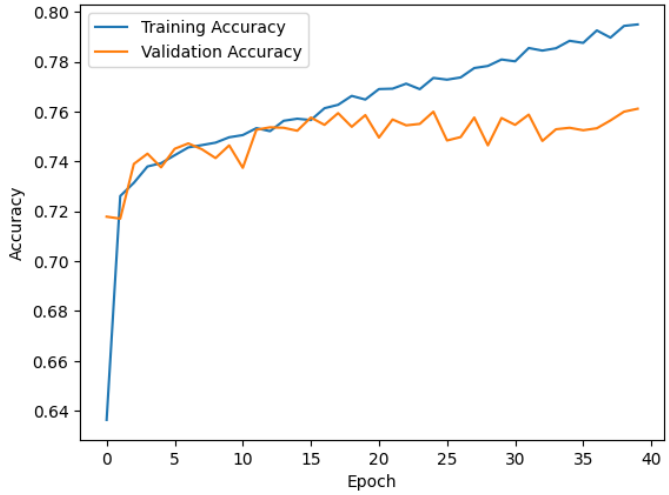
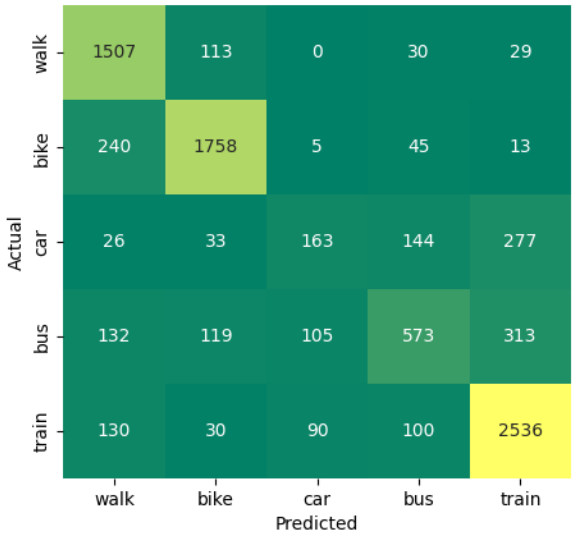


Figure 17: Confusion Matrix of CNN on the smaller Dataset



The larger dataset (which contains trajectories with inconsistent time periods between them) did not perform as well as it obtained an accuracy of 73.6%. The 3.5% drop off in accuracy from the smaller dataset is significant when you consider that the larger dataset trains on roughly 8 times more data. This shows that for CNNs training on motion data, it is important to use a low and consistent time difference between trajectories in order for the CNN to find distinct features for each travel mode.

However, the highest accuracy achieved in this whole study was 82.6% which was found when the group size was set to 64 and the larger dataset, that includes all time ranges, was used. This is somewhat unexpected because it has a higher accuracy than the

version of this dataset which filtered out inconsistent time ranges (for the same group size of 64), which obtained an accuracy of 81.5%.

Figure 18: Confusion Matrix for CNN with the Highest Accuracy

4 Comparisons of The Models

It is important to compare these models because some models make work better than others under certain forms of the data that is being passed into them (e.g. a larger dataset but with high number of inconsistencies compared to a small but consistent dataset).

Table 7: Accuracies of all the Models

Model	Group size	Range of 'Δ time'	Accuracy
DT	16	1s ≤ Δ time ≤ 6s	70.5%
DT	16	Δ time = 1s	74.6%
DT	64	1s ≤ Δ time ≤ 6s	79.0%
DT	64	Δ time = 1s	81.1%
SVM	16	1s ≤ Δ time ≤ 6s	66.7%
SVM	16	Δ time = 1s	72.24%
SVM	64	1s ≤ Δ time ≤ 6s	74.1%
SVM	64	Δ time = 1s	77.6%
CNN	16	1s ≤ Δ time ≤ 6s	73.6%
CNN	16	Δ time = 1s	77.1%
CNN	64	1s ≤ Δ time ≤ 6s	82.6%
CNN	64	Δ time = 1s	81.5%

It is clear from this table that all of the models perform better when training with a dataset that has more trajectory data in each group (despite the length of the dataset being proportionally smaller to the group size).

The highest accuracy was recorded by the CNN when 'Δ time' was not consistent, this was unexpected and shows that it is important for these models to be trained on a lot of data.

5 Conclusion & Future Work

5.1 Conclusion

After comparing all of the models it was found that a CNN which was trained on the dataset with 64 trajectories per group (which is the highest amount this project tested with) and allowed for any time between 1 and 6 seconds for consecutive trajectories was able to obtain the highest accuracy, 82.6% in this project.

However, the decision tree with a max depth of 5 (which is 31 decisions) was able to achieve 81.1% accuracy, this shows the effectiveness of decision trees despite their simple form.

The accuracy of SVMs were weaker than the other 2 models when comparing them by every form of the dataset. This suggests that they are not well suited to this classification task.

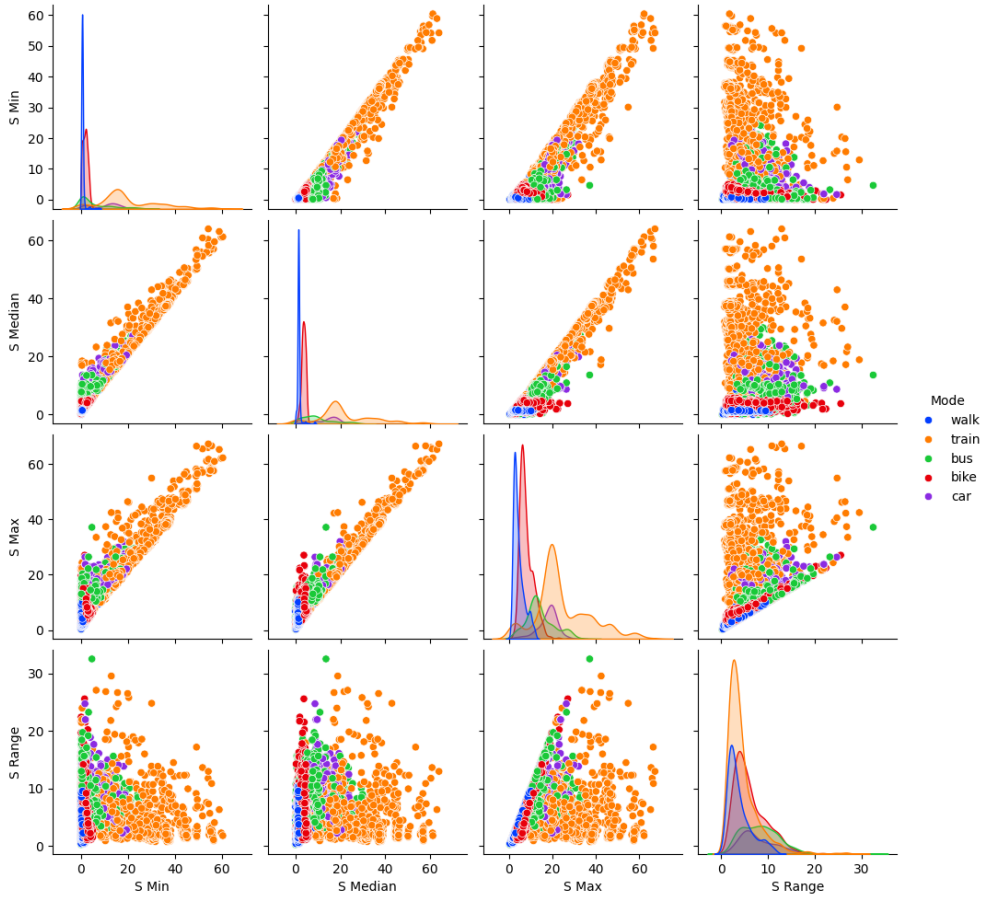
5.2 Further Work

The data used to train, test, and compare models was not consistent because there was not enough data when it got filtered down, and also because the data was not spread out evenly between the 5 classes, this caused biases in the data which means these models may perform very poorly if they were fed new grouped motion data were these biases towards some classes are not present. This means that in future, in order to train better models for travel mode identification, more data is needed.

Additionally, a higher accuracy could have likely been achieved with the data available. The model for the CNN that achieved the highest accuracy is very simple version of what is possible with CNNs.

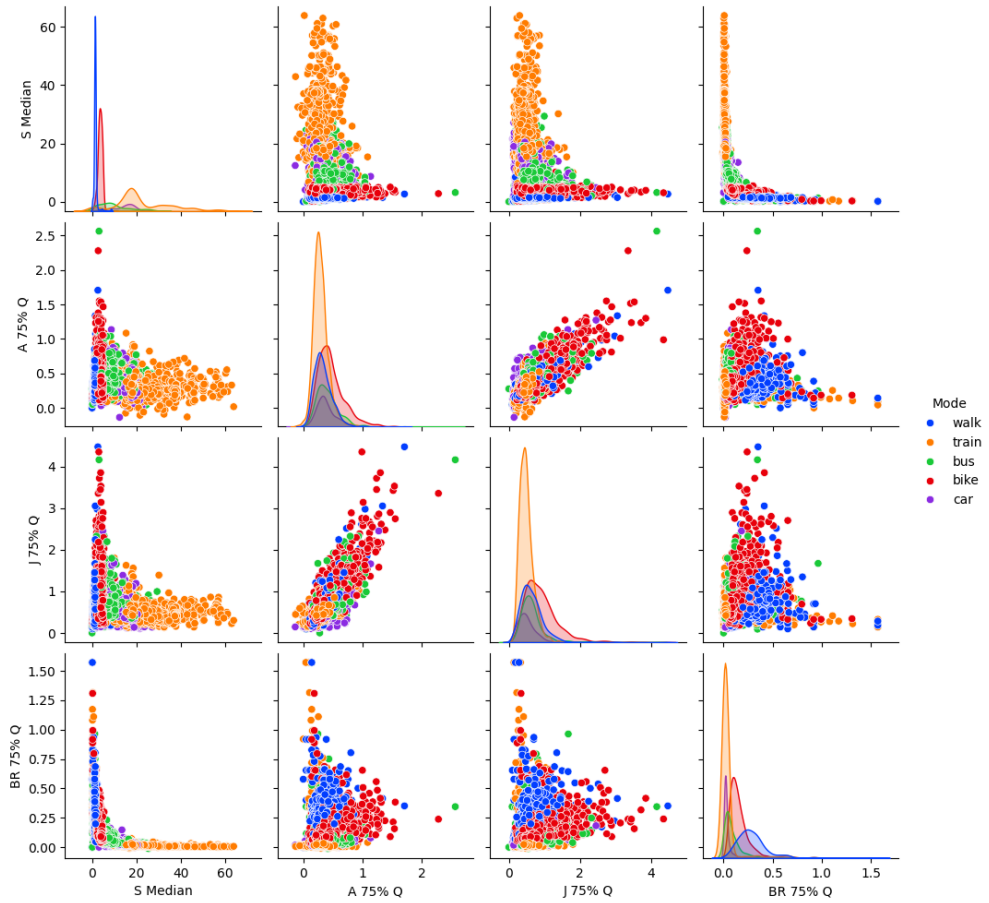
Appendix A

Pairplot that compares minimum speed, median speed, maximum speed, and the range of speeds for different travel modes



Appendix B

Pairplot that compares median speed, 75th quartile of acceleration, 75th quartile of jerk, and 75th quartile of bearing rate of different travel modes.



6 Bibliography

[1] Statista, “Smartphone adoption rate worldwide in selected years from 2021 to 2030, by region,” statista, February 2024. [Online]. Available: https://www.statista.com/statistics/1258906/worldwide-smartphone-adoption-rate-telecommunication-by-region/. [Accessed May 2024].

[2] J. Krumm, “A survey of computational location privacy,” 2008. [Online]. Available: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/12/computational-location-privacy-preprint.pdf. [Accessed May 2024].

[3] Z. Yu, F. Hao, X. Xing, M. Wei-Ying and L. Quannan, “Geolife GPS trajectory dataset - User Guide,” Microsoft Research Asia, July 2011. [Online]. Available: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/User20Guide-1.2.pdf. [Accessed November 2023].

[4] S. Dabiri and K. Heaslip, “Inferring transportation modes from GPS trajectories using a convolutional neural network,” January 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X17303509. [Accessed February 2024].

[5] T. Vincenty, “Direct and Inverse Solutions of Geodesics on the Ellipsoid with Application of Nested Equations,” pp. 88-93, 2013.

[6] Google, “Google Colaboratory,” [Online]. Available: https://colab.research.google.com/. [Accessed May 2024].

[7] pandas, “pandas - Python Data Ananlysis Library,” [Online]. Available: https://pandas.pydata.org/. [Accessed May 2024].

[8] J. Morgan and J. Sonquist, “Problems in the Analysis of Survey Data, and a Proposal,” June 1963. [Online]. Available: https://cs.nyu.edu/~roweis/csc2515-2006/readings/morgan_sonquist63.pdf. [Accessed May 2024].

[J. Quinlan, “Induction of Decision Trees,” 1986. [Online]. Available:
9 https://hunch.net/~coms-4771/quinlan.pdf. [Accessed May 2024].
]

[J. Quinlan, C4.5 Programs for Machine Learning, Elsevier, 1992.
1
0
]

[A. Dwarakanath, “Know your SVMs,” Medium, March 2018.
1 [Online]. Available: https://medium.com/@anuragbms/know-your-
1 svms-58977c4f09aa. [Accessed May 2024].
]

[C. Cortes and v. Vapik, “Support-Vector Networks,” 1995. [Online].
1 Available:
2 https://link.springer.com/content/pdf/10.1023/A:1022627411411.p
] df. [Accessed May 2024].

[Impvis, “Different Kernel Functions,” [Online]. Available:
1 https://impvis.co.uk/launch/support-vector-machines/rbf.html?.
3 [Accessed May 2024].
]

[C. Burges, “A Tutorial on Support Vector Machines for Pattern,”
1 Kluwer Academic Publishers, Boston, 1998.
4
]

[K. Fukushima, “Neocognitron: A self-organizing neural network
1 model for a mechanism of pattern recognition unaffected by shift in
5 position,” *Biological Cybernetics,* pp. 193-202, 1980.
]

[Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, “Gradient-based
1 learning applied to document recognition,” *Proceedings of the*
6 *IEEE,* pp. 2278-2324.
]

[Scikit learn, “sklearn.model_selection.train_test_split,” [Online].
1 Available: https://scikit-
7 learn.org/stable/modules/generated/sklearn.model_selection.train
] _test_split.html. [Accessed May 2024].

[Scikit Learn, “sklearn.tree.DecisionTreeClassifier,” [Online].
1 Available: https://scikit-

[18] learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html. [Accessed May 2024].

[19] Scikit Learn, “sklearn.svm.SVC,” [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html. [Accessed May 2024].

[20] TensorFlow, “An end-to-end platform for machine learning,” [Online]. Available: https://www.tensorflow.org/. [Accessed May 2024].

[21] F. Calabrese, M. D. L. G. Diao, J. Ferreira and C. Ratti, “Understanding individual mobility patterns from urban sensing data:,” 2013. [Online]. Available: https://senseable.mit.edu/papers/pdf/20130115_Calabrese_etal_UnderstandingIndividual_TransportationResearch.pdf. [Accessed May 2024].

[22] S. Kettle, “Distance on a sphere: The Haversine Formula,” 2017. [Online]. Available: https://community.esri.com/t5/coordinate-reference-systems-blog/distance-on-a-sphere-the-haversine-formula/ba-p/902128. [Accessed May 2024].

[23] S. Cheng and Y. liu, “Research on Transportation Mode Recognition Based on Multi-Head Attention Temporal Convolutional Network,” March 2023. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10098534/#:~:text=The%20experimental%20results%20demonstrate%20that,recognition%20effect%20on%20transportation%20modes.. [Accessed February 2024].

[24] S. Giri, R. Brondeel, T. El Aarbaoui and B. Chaix, “Application of machine learning to predict transport modes from GPS, accelerometer, and heart rate data,” November 2022. [Online]. Available: https://ij-healthgeographics.biomedcentral.com/articles/10.1186/s12942-022-00319-y. [Accessed February 2024].

[25] Z. Yu, F. Hao, X. Xing, M. Wei-Ying and L. Quannan, “Geolife GPS Trajectories,” Microsoft Research Asia, May 2016. [Online]. Available: https://www.microsoft.com/en-us/download/details.aspx?id=52367. [Accessed November 2023].

[26] D. Bhatnagar, “History of CNN & its impact in the field of Artificial Intelligence,” Medium, January 2023. [Online]. Available: https://medium.com/international-school-of-ai-data-science/history-of-cnn-its-impact-in-the-field-of-artificial-intelligence-2b1efb7d99e5. [Accessed May 2024].

[27] Prathammodi, “Convolutional Neural Networks for Dummies,” Medium, October 2023. [Online]. Available: https://medium.com/@prathammodi001/convolutional-neural-networks-for-dummies-a-step-by-step-cnn-tutorial-e68f464d608f. [Accessed May 2024].