Transportation Mode Detection with GPS Trajectory Data using Classification

Bobby Brady, Deepasha Jenamani, Gayathri Sundareshwar, & Keerthana Gopikrishnan

Department of Applied Data Science, San Jose State University

DATA 270 – Data Analytics Process

Dr. Eduardo Chan

December 11, 2022

Abstract

Advancements in technology and rapid transit systems have led to immense research which could aid in improving social and ecological systems. Existing research in the trajectory analysis lane deals mainly without grouping to distinguish latent patterns that could be due to the transport nature utilizing either supervised machine learning or deep learning but not combined in a single study. Also, the trajectories were concentrated on a single medium in existing studies. This study classifies vehicle modes based on GPS trajectories with high accuracy while detecting the changes in modes by allowing multiple classifications for a single sensor. It also generalizes the latent patterns in GPS trajectories to accurately classify vehicle modes for various localities. The significant difference relies on condensation and exhaustive training. Extensive feature extraction combined with dimensionality analysis was performed, unlike existing studies that concentrated on either. The study involvesZ a dataset collected over five years. Extensive feature identification was used to extract quantifiable information from the research data, which various ML Models can utilize. The models chosen were Random Forest (RF), SVM, Decision Tree (DT), and LSTM. Adding extensive features and opting for cross-validation helped narrow the characteristics of each travel mode and enhance precision. RF achieved the highest performance (82%), followed by SVM(74%), whereas SVM was faster in the performance test. As the future scope of GPS Data can vastly vary, it is hopeful that the study can utilize real-time data and extend to other research areas, such as city planning and target marketing.

Introduction

Project Background and Executive Summary

The transportation industry has constantly been evolving. Every year, innovations have led to the development of new technologies that could reshape the sector entirely. These new technologies have increased the efficiency of transportation systems over a period. Some of the facts that influenced in pursuit of this topic are listed here. First, the preliminary data from the USDOT's FHWA showed an upward trend in the miles traveled compared to the same time of the previous year, the first half of 2022. In other words, a 2.8% increase of approximately 43.2 billion miles traveled by vehicles (NHTSA Media, 2022). Second, the projected annual growth rate of the public transportation market is 5.9% from 2022 to 2028 (Global Public Transportation Market Size Report, 2028, n.d.).

Advancements in the rapid transit systems have also made it possible for people to have various choices in deciding how to travel. Though several factors account for influencing the decision-making and behavior of a person while traveling, there is one predominant factor that always comes atop, which is travel mode. The information gathered by the sensors in mobile data and GPS (Global Positioning System) Devices regarding the mobility pattern can be used in various sectors. Due to this large influx of data collected by GPS Devices, a significant amount of attention is focused on determining how this data will be utilized efficiently. One such way to use this data will be to identify and classify the modes of transportation.

The proposed project aims to classify vehicle modes based on GPS trajectory data with high accuracy. Multiple models are added as a part of the proposed approach so that the potentially best-suited one can be identified. In addition, it can also help detect the changes in vehicle mode by allowing multiple classifications for a single sensor. It also aims to generalize

the latent patterns in GPS trajectories to accurately classify vehicle modes for various localities. The research can be helpful in realistic scenarios like assisting transportation systems in identifying travel patterns and congestion.

Detecting these modes will help determine the traveling patterns and also aids in attaining additional insights through the data. Identifying the transportation modes can be used in several applications. Some of such applications are listed here. Identifying preferred travel modes can help improve urban township planning, transportation, and other affiliated businesses that can thrive with it. The data recorded by the GPS Sensors in mobile phones can also help track the carbon footprint, and the calories burnt depending on the transportation mode. Health, hazard, and environmental impacts can be tracked through environmental applications. Currently, multiple approaches exist to detect the travel mode using the data collected by GPS Devices, smartphone sensors, and other devices that could track the motion patterns. It can also aid in identifying potential users and their current vehicle modes, which can be helpful in targeted marketing.

Project Approaches and Methods

An increase in the population and vehicles is directly proportional to the rise in data collection using GPS Sensors. This increase makes it mandatory to have better roadway planning and better utilization. Detecting transportation modes will also be helpful in multiple scenarios, such as urban planning, targeted marketing, traffic regulations, and planning for unexpected scenarios. Hence, it is necessary to devise methodologies for transportation mode detection and discuss the usage of the insights attained. The following section highlights the in-depth data collection methods, pre-processing techniques employed to cleanse the data, model selection, and pictorial representation of the workflow. This discusses the methodologies in detail.

Data Collection Approaches

The dataset used in this research is a GPS trajectory dataset collected over five years, from April 2007 to August 2012. The data collected by Microsoft Research Asia involved 182 users. The 17,621 trajectories collected consist of sequential time-stamped points containing information regarding the latitude, longitude, and altitude.

Various GPS loggers and Phones were involved in the data collection process and sampled at various rates. The records logged in dense representation account for 91.5% of the dataset. Dense representation implies that the records collected are in an interval of five to ten meters or one to five seconds. The dataset contains the logs of usual movements like going to work and home. In addition, it also includes the records of other activities like entertainment and sports, such as shopping, dining, sightseeing, jogging, hiking, and cycling. The location of data collection is mainly in the cities of China. The data collection process required a segment of users to carry GPS loggers for years and the other set to have trajectory recorded only over a few weeks. This dataset can have a wide variety of usage involving research fields such as pattern mining, activity recognition, and recommendation systems.

Data Pre-Processing Techniques

The dataset chosen must be passed through various pre-processing techniques and condensed to a format that allows better utilization and insight gathering. Since the data set chosen is in a vector-based plotter (.plt) format, it requires a specific set of pre-processing directions. The following section discusses the pre-processing techniques used.

Labeling. The first pre-processing step will be checking if each user's dataset file directories have a label file attached. Non-labeled trajectories in the dataset will not be used in this research due to the label requirement, thus deducting them from the input fed to the model.

Format Conversion and Merging Data Sets. The second step will be converting the file format into a better-suited one. The process involved in this step would be extracting the .plt format and bringing it down to the comma-separated values (CSV) format. The conversion to CSV format also happens with the label files originally in .txt format. It is also worth noting that the .plt files present in the dataset are a cluster of individual files containing records of every user data based on time stamps. Hence, when converting the files to CSV format, the datasets are merged into a single CSV file containing all the user's data with the labels added. CSV formatted file also ensures better usability and easy access.

Feature Extraction and Normalization. The third step would be feature extraction. The features extracted from the CSV file are sequentially time-stamped by start and end time. The other features extracted include latitude, longitude, altitude, and transportation mode, among which the transportation mode will serve as the target for prediction. The original set of features from the file can be used to calculate and extract additional features. Some of the additional features extracted are the time gap being the difference between end and start time, distance gap being the difference between the coordinates, and speed calculated using the distance divided by time formula, acceleration, total time, and total distance. The addition of calculated features can help extract more distinct features such as maximum, mean, standard deviation, and different quartiles of speed and acceleration. It is then necessary to normalize the extracted features to the standard order of magnitude so that the higher-order values do not influence the classification results by inducing training bias.

Data Wrangling. The insights attained from wrangled data tend to be more useful in analysis, and the surety of it being more reliable than the raw data is indisputable. Hence after the feature extraction, the CSV file needs further scrubbing so that the models can learn better

and predict better. Thus, the fourth step would be purging the data set using different scrubbing methodologies, such as eliminating unnecessary records and cleansing null entries.

The first phase of scrubbing would be eliminating the records where the start and end times are the same since it implies that no significant movement could have occurred, and hence eliminating such records would be an ideal choice. Likewise, the second would be eliminating records with latitude and longitudinal exceptions—for instance, removing the records with longitude greater than 180 and lesser than -180. The next scrubbing phase involves the detection of rows with complete null values and eliminating them. Eliminating null rows can help increase processing speed. The data wrangling process will bring the data closer to the cleaner validated form.

Sampling. The dataset tends to be slightly on the heavier side; hence a random, unbiased sampling has been performed to reduce the load on the system. The sampling should not negatively impact the model's prediction results and must be mindful of the variety factor. The sampling rate should sufficiently induce a resultant set with adequate data. Once all the pre-processing steps mentioned above are performed, the resultant data set can be passed through different ML models and analyzed further.

Models

This section of the study discusses different algorithms and models adapted for the travel modes classification along with the corresponding advantages and disadvantages of each. Some of the algorithms analyzed in this section are Random Forest (RF), Support Vector Machine (SVM), Decision Tree, and Long-Short Term Memory (LSTM).

Random Forest. One of the chosen models for this research is the Random Forest Classifier. The algorithm boasts unique advantages, such as the ability to generate plenty of

classification trees while resisting noise. The model randomly selects data and variables from the source set to proceed with the prediction. A random forest classifier is handy when dealing with diverse data sets with high dimensionality.

Random forest algorithms also seem to have a higher training speed which in turn tends to help reduce the computation cost. While performing background research related to the topic, it was interesting to notice that multiple studies with the Random Forest Classifier as a chosen model were prevalent. Wang et al. (2018) and Gao et al. (2020) proposed the usage of random forest classifiers to detect travel modes, and the results of the study attained sustainable results. Figure 1 below shows the accuracy attained as the result of the study by Wang et al. (2018) (p. 10), and Figure 2 shows the accuracy attained during the study by Gao et al. (2020) (p. 6).

The result shown in figure 1 shows the confusion matrix of GPS Data and Socioeconomic Attributes. The experiment proved the efficiency of the Random Forest. The results of figure 2 also shows the results of the smartphone based survey that was performed with various travel modes identified. The accuracy attained during the study proposed by Wang et al. (2018) and Gao et al. (2020) for Travel Mode detection played a dominant role while deciding on the model selection.

Figure 1

Confusion Matrix of GPS Data and Socioeconomic Attributes

Actual	Predicted					Precision	Recall	TPR	FPR	F-measure
	Walking	Bicycle	e-bike	Bus	Car	(%)	(%)			
Walking	463	1	0	0	0	99.4	99.8	0.998	0.007	0.996
Bicycle	3	75	2	2	1	92.6	90.4	0.904	0.007	0.915
e-bike	0	2	17	2	0	85.0	81.0	0.810	0.003	0.829
Bus	0	2	1	99	24	79.2	78.6	0.786	0.033	0.789
Car	0	1	0	22	197	88.7	89.5	0.895	0.036	0.891

Figure 2

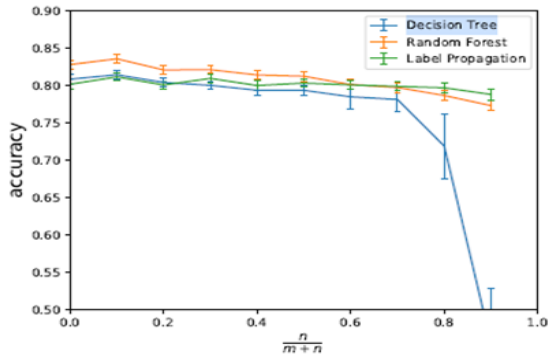
Mode Detection Accuracy Result for RF Models

Attribute		Speed		Acceleration	
		Average (m/s)	Std. Dev. (m/s)	Average (m/s2)	Std. Dev. (m/s2)
Peak hours	Walk	1.194	0.542	0.017	0.106
	Bike	2.853	0.753	0.005	0.185
	Bus	5.105	3.032	0.113	0.696
	Car	6.718	3.217	0.081	0.642
Off-peak hours	Walk	0.986	0.634	0.012	0.058
	Bike	2.565	0.839	0.010	0.218
	Bus	5.946	3.052	0.063	0.594
	Car	7.883	3.154	0.031	0.599

Decision Tree. Decision trees can be considered one of the most prominent and baseline models for classification problems. It follows a hierarchical tree structure and contains various components such as a root and leaf node, branches, and internal nodes. Starting from a root node, it flows onto the internal nodes and finally reaches the leaf nodes, which represent the potential outcomes. Decision trees tend to be more flexible in exploring due to the characteristics involving non-linearity. Decision trees are especially effective when dealing with multiple potential outcomes from a range. Furthermore, since the study at hand is a classification problem with a set of potential outcomes, a decision tree is chosen as one of the models to be explored and compared with the other potential models. Previous research works related to the topic have explored the possibility of using a decision tree for transportation mode detection, its success, and its pain points. Some of such works are discussed briefly in the following part. Rezaie et al. (2017) and F. de S. Soares et al. (2021) proposed an approach to classify travel modes using smartphone data entangling a Decision Tree Classifier in their respective studies (p. 1). Figure 3 below shows the results of the study by Rezaie et al. (2017) (p.7).

Figure 3

Average Accuracy Comparison of DT, RF, and LP



As we can see in the above picture, the accuracy of the decision tree tends to be comparatively low. However, the data source used in this study is semi-supervised, which means unlabeled. Decision trees work better with labeled data sources. Since the dataset chosen for this study involves a labeled data set, it is expected to perform better than the study results. Hence it is also chosen as one of the performing algorithms. Figure 4 below shows the accuracies attained by the results of the study by F. de S. Soares et al. (2021).

Figure 4

Performance Metrics of Various Classifiers

Classifier	$Average\ Accuracy$	$Precision_M$	$Recall_M$	$F1score_M$
MLP	45.6%	37.8%	31.8%	31.0%
SVM	62.2%	60.3%	60.2%	60.3%
DT	67.1%	52.0%	46.9%	49.3%
BN	67.0%	45.0%	36.2%	40.1%

The result of the study shows that the decision tree had the highest accuracy among the model, but it is not adequate to classify as an efficient classification process. The data source chosen for this study is real-time; hence, it could be unclean due to its raw nature. The algorithm could perform better with an extracted data source. Hence this model is chosen as one of the algorithms to perform the analysis.

Support Vector Machine. The next model chosen for travel mode detection is Support Vector Machines (SVM), which uses linear and non-linear enactments and is especially helpful with data containing highly complex datasets. SVM tends to be less prone to overfitting and performs well when involving margin separations. SVM is one of the most preferred algorithms for classification problems, and many existing works related to the topic at hand attest to that. SVM generalizes the difference between the classes with the help of a hyperplane.

A hyperplane is a line that distinguishes and establishes the margin between the classes. Suppose the data in hand is linearly non-separable, then the SVM uses kernel tricks to make it potentially separable. One advantage of SVM is that it does not get influenced by the presence of outliers and skew the results. Numerous related kinds of research aided in attaining insights into the utilization of SVM better. Some of the related works are discussed here briefly. Zhu et al. (2021) proposed a model to detect the light-duty vehicle's (LDV) travel mode, concentrating mainly on highly imbalanced datasets. The training and validation involve real-world, large-scale datasets. The proposed model uses OCSVM in comparison with SVM. The study also explored the addition of an extensive list of features extracted (pp. 1, 7). Figure 5 shows the performance of OCSVM for all the travel modes in the study by Zhu et al. (2021) (p. 7).

Figure 5

Performance of OCSVM for all Travel Modes

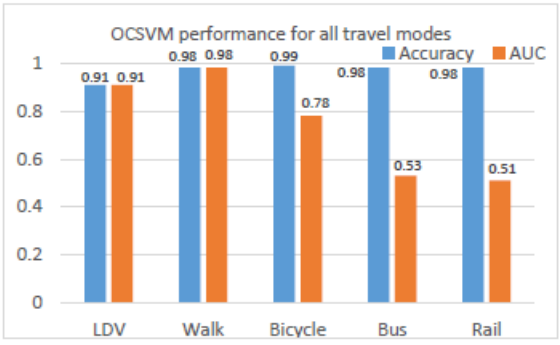


Fig. 5. OCSVM method performance for all travel modes within data set #5

The study results proved that a regular SVM works exceptionally well with higher accuracy. When observing the results closely, we can see that the results of the regular SVM algorithm with full features perform slightly better than the SVM with three features. Since the proposed project involves extracting additional features during the pre-processing stage, this reference played an essential role in selecting this model.

Asad et al. (2021) proposed a model to predict mobility during the pandemic. Six machine learning algorithms were employed to carry out this prediction to identify the potentially better-performing one. This study helps to understand the impact of the pandemic on transportation modes and will, in turn, be beneficial in optimizing the economic benefits. The study also aids in understanding how each of the algorithms works; SVM is one of the six chosen algorithms in this study. Figure 6 shows the accuracy achieved by each model tested in Asad et al. (2021) study (p. 5). The study result shows that SVM had a higher accuracy when compared with all the models and proved that it could come in handy to detect the travel modes with the dataset in hand.

Figure 6

Transportation Mode Classification Accuracy

Machine Learning Classifier	Accuracy	Precision	Recall	F-Measure
Support Vector Machine (SVM)	91.25	0.91	0.90	0.90
Naive Bayes (NB)	86.25	0.86	0.85	0.86
Multi-Layer Perceptron (MLP)	83.0	0.83	0.82	0.83
Logistic Regression (LR)	81.9	0.81	0.80	0.81
K-Nearest Neighbour (KNN)	81.9	0.81	0.80	0.81
Random Forest (RT)	86.0	0.86	0.85	0.86

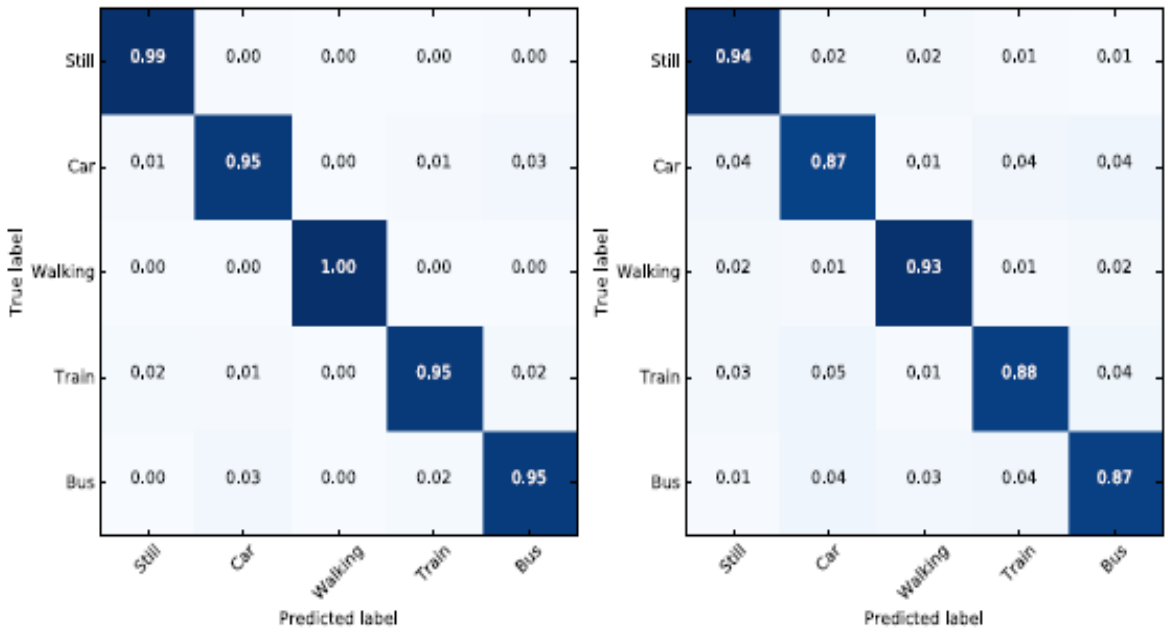
Long-Short Term Memory. Long-Short Term Memory is a Recurrent Neural Network (RNN) model type that has recently been a hot topic in research related to classifications. LSTM

frameworks involve the basic concept of withholding data that appear necessary and discarding the ones not useful for predictions in the future. LSTM is also cost-efficient and consumes less time due to the discarding characteristic. Since this is relatively new, the number of existing materials related to the research topic is sparse, but not none. A few works used LSTM and explored the possibility of devising a better classification model for transport mode detection. Some of the works are discussed here briefly.

F. de S. Soares, Salehinejad et al. (2019) proposed an approach to identify travel modes using a recurrent neural network algorithm combined with domain feature extraction. The methodology used is a deep RNN architecture with LSTM cells using a wide range of sensor data (pp. 1, 5-6). Figure 7 shows the confusion matrices of average test and train sets examined in the study by F. de S. Soares, Salehinejad et al. (2019) (p. 5). LSTM appears to have the highest accuracy while still being on the lighter side of the scale regarding weight.

Figure 7

Confusion Matrices of Average Test and Train of Best Performing LSTM



Wang et al. (2021) proposed a machine learning algorithm that uses the sensors incorporated in commercial smartphones to identify patterns and also help deal with the limitations regarding the accuracy levels of previously existing works. The method suggested in this work is a residual and LSTM recurrent networks-based transportation mode detection algorithm (p. 1). Figures 8 and 9 show the accuracy attained using LSTM on different datasets in the study by Wang et al. (2021) (p.11). The study results are one of the main reasons behind choosing this model, and the other one is the recency of trend and scarcity of related materials.

Figure 8

Precision, Recall and F1-Score using LSTM

PRECISIONS, RECALLS AND F1-SCORES OF DIFFERENT MODES USING THE MSRLSTM MODEL ON THE HTC DATASET AND THE ICT DATASET

Dataset	Mode	Precision	Recall	F1-Score
HTC Dataset	Still	96.87%	97.72%	97.29%
	Walk	95.70%	96.32%	96.01%
	Run	98.21%	98.83%	98.52%
	Bike	96.75%	97.21%	96.98%
	Car	96.31%	96.60%	96.46%
	Bus	92.08%	89.70%	90.87%
	Train	94.60%	93.51%	94.05%
	Subway	93.27%	92.37%	92.82%
ICT Dataset	Still	99.79%	99.96%	99.87%
	Walk	99.83%	99.96%	99.89%
	Car	96.38%	97.26%	96.82%
	Bus	98.27%	94.72%	96.46%
	Train	99.50%	99.68%	99.59%
	Subway	99.72%	99.37%	99.55%

.

Figure 9

Precision, Recall and F1-Score of Sussex-Huawei Locomotion Dataset using LSTM Model

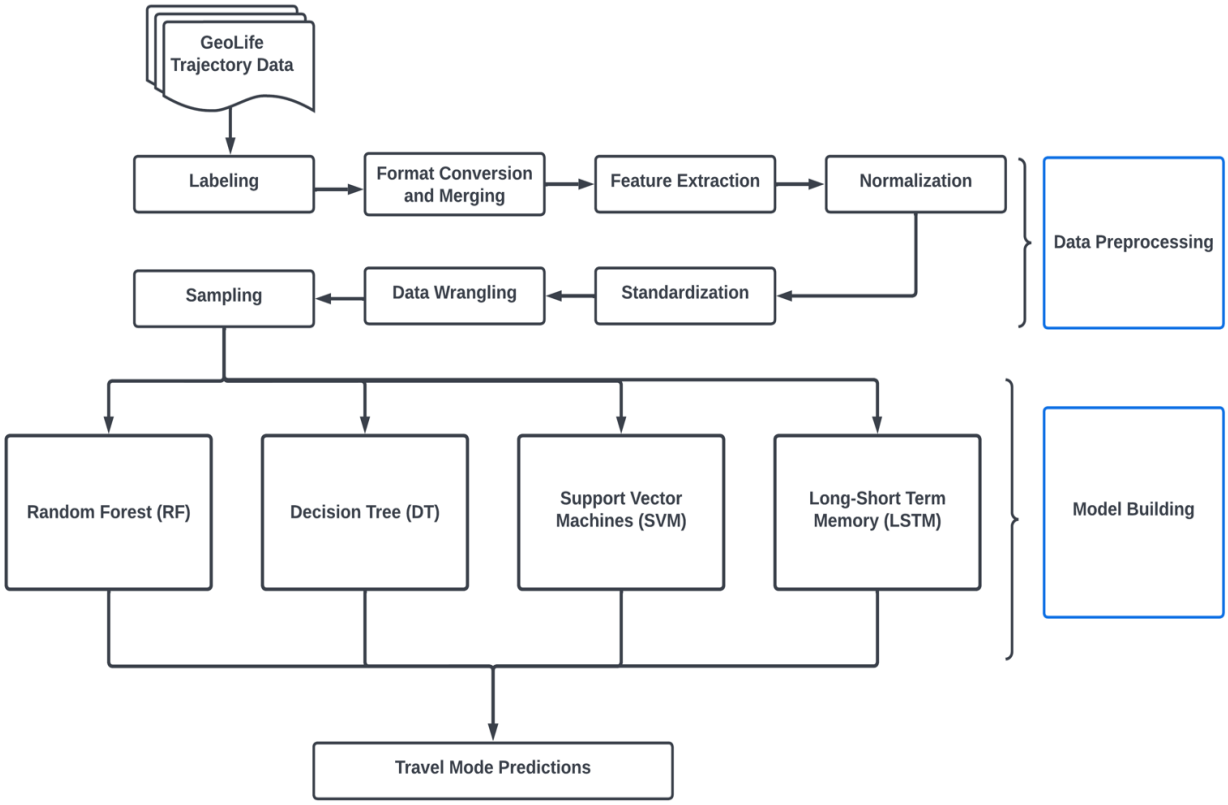
	Precision	Recall	F1-Score
Still	97.47%	98.29%	97.87%
Walk	98.25%	98.11%	98.18%
Run	99.66%	99.60%	99.63%
Bike	98.95%	98.05%	98.50%
Car	99.44%	99.27%	99.35%
Bus	97.15%	98.78%	97.96%
Train	96.77%	96.14%	96.45%
Subway	95.51%	95.00%	95.26%

Workflow Diagram

The workflow diagram helps represent the flow of the steps associated with the process involved in the proposed study. It provides an overview of the tasks involved in the research completion and the flow of resources through those various tasks and helps understand the action items. The workflow diagram of the proposed study is illustrated in this section. Figure 10 shows the workflow of the proposed work.

Figure 10

Workflow Diagram



The proposed study is expected to be helpful in various areas, such as urban planning, targeted marketing, and improving the current transit systems. Identifying and classifying the different travel modes using the data recorded through GPS sensors will help understand the concentration of the distribution of each travel mode in different demographics. Hence, detecting

the travel modes can play a vital role in multiple scenarios, such as deciding on the requirement of new roadways in a particular location. It can also help concentrate more on marketing the products related to a specific travel mode in the most preferred areas.

Project Requirements

Functional Requirements. The dataset used in this project, Microsoft Geolife project GPS trajectory data, must be acquired through proper procedures. According to Microsoft Research (2021), accessing the dataset for download requires a Microsoft Research Open Data account. Prior to downloading, the user must agree to the terms of the CDLA Permissive 2.0 license and submit a dataset usage reason. A zip or tar file containing the entire raw dataset can then be downloaded directly with a browser. The model requires extracting features for each raw input trajectory object during data pre-processing. Along with the extracted features, it is also advisable to add additional calculated features, which could play a vital part in ensuring the model performance concerning the accuracy of classifications. Being cautious of the mentioned factors leads to a project with a successful outcome.

The model will require labels for all extracted features, which is integral in training, verification, and test datasets. Training for all algorithms will be conducted by supervised learning, requiring labels to be present in training and verification datasets. Labels will be excluded during testing but will be required for checking classification accuracy when evaluating model performance. This ensures that the models can generalize and accurately measure and summarize performance. Non-labeled trajectories in the dataset will not be used in this research due to the label requirement, thus deducting them from the input fed to the model.

Computational needs for the model require both sufficient resources for implementing machine learning and deep learning algorithms. System storage size must be adequate for

acquiring, storing, and processing the raw dataset, a minimum of 3GB. System memory size must be adequate for loading the entire raw input dataset for pre-processing and algorithms performing classification tasks, a minimum of 8GB. A GPU with a minimum of 8GB of memory is needed to load the entire input dataset for leveraging classification algorithm libraries that use GPU parallelization capabilities. These minimum requirements will ensure successful data loading for model tasks and timely performance for computationally intensive tasks, particularly model training. The above-listed requirements ensure the timely completion of tasks in the project plan and aid in meeting deadlines.

AI Requirements. Cortes and Vapnik (1995) developed a Support Vector Machine (SVM) as a supervised learning model that implements a margin to maximize separation between two classes with the ability to find non-linear separation through kernel trick. The margin is defined by a controllable number of feature vectors that maximize separation between classes. Only a tiny subset of the input data is evaluated, minimizing the influence of outliers (pp. 273, 285). This model is well suited to our research because of its efficiency in dealing with non-linear classifications.

The decision tree (DT) is a supervised learning model that predicts classification based on a sequence of tests. All DT begins with a single test called the root, and each test results in two or more branches that connect to a subsequent test or termination called a leaf. DT is purely deterministic, and each leaf represents a class or probability distribution. Controlling, interpreting, and explaining DT model classifications is simple, making it well-suited for our research.

Ho (1995) developed random forest (RF) as a supervised learning model where multiple DT outputs determine the classification. This approach reduces variance in individual DTs

through random sampling and aggregation during training (pp. 278–279). RF is well suited for our research due to its straightforward implementation and interpretation of classifications.

Hochreiter and Schmidhuber (1997) developed long short-term memory (LSTM) as an artificial neural network (ANN) that can be used for supervised learning of time-series data. LSTM can learn short-term patterns and retain this information for many time lags. The model allows information to propagate through the network unchanged. This model can learn and optimize parameters automatically (pp. 1750, 1755-1756). LSTM is well suited for our research because it does not need procedures outside the model implementation to determine critical parameters for optimization.

Data Requirements. Microsoft Research Open Data (n.d.) published the GPS trajectory dataset. The data source utilizes multiple GPS loggers and phones, recording data every 1~5 seconds or 5~10 meter points. As part of the data collection process, one set of users carried GPS loggers for years, whereas the other set carried it only for a couple of weeks. The Geolife project tracked 182 users over five years, from April 2007 to August 2012. The recorded data contains a user's log file in PLT format. The dataset contains the logs of usual movements like going to work and home. In addition, it also includes records of other activities like entertainment and sports, such as shopping, dining, sightseeing, jogging, hiking, and cycling.

The records logged in dense representation account for 91.5% of the dataset. The 17,621 trajectories collected consist of sequential time-stamped points containing information regarding the latitude, longitude, altitude, the number of days since December 30, 1899, the current date, and the time. The third field is blank. Table 1 below depicts the User's log file. The file consists of features representing geographical information such as latitude, longitude, and altitude and also contains timestamp information monitoring the travel made.

Table 1

User's Log File Fields

Field name	Description	Type
Field 1	Latitude	Decimal degrees
Field 2	Longitude	Decimal degrees
Field 3	All set to 0	NA
Field 4	Altitude (-777 for invalid)	Feet
Field 5	Date – number of days passed since 12/30/1899	Decimal
Field 6	Date	String
Field 7	Time	String

Figure 11 below depicts an example of a user's log file where a comma is used as a delimiter to separate the fields. Line 1 to 6 does not hold any significance and hence can be ignored. A travel modes label is also available, with various modes such as light-duty vehicles, trains, airplanes, and boats. Table 2 represents the fields in a user's transportation mode label file, which is available for 73 users. Figure 12 represents an example of this data, which includes the start and end time and the mode of transportation. The dataset is 1.67 GB in size and contains 17,621 trajectories. Understanding the dataset will help expedite the process flow and also aid in deciding the right way to tackle the study.

Figure 11

Example of User's Log File

```
Geolife trajectory
WGS 84
Altitude is in Feet
Reserved 3
0,2,255,My Track,0,0,2,8421376
0
39.9733333333333,116.3294,0,472.44094488189,39415.4023958333,2007-11-29,09:39:27
39.97585,116.330466666667,0,416.666666666667,39415.4025347222,2007-11-29,09:39:39
39.976,116.33045,0,360.892388451444,39415.4025810185,2007-11-29,09:39:43
39.9761166666667,116.3303,0,291.994750656168,39415.4026967593,2007-11-29,09:39:53
```

Table 2

Transportation Label Fields

Field Name	Description	Type
Field 1	Start Time	GMT
Field 2	End Time	GMT
Field 3	Transportation Mode	String

Figure 12

Example of the Transportation Label File

```
Start Time          End Time Transportation Mode
2008/03/30 08:20:50      2008/03/30 08:37:01      car
2008/03/30 08:45:43      2008/03/30 10:09:13      car
2008/03/30 10:38:13      2008/03/30 11:02:45      car
2008/03/30 12:54:57      2008/03/30 13:08:53      car
2008/03/30 13:08:53      2008/03/30 13:29:50      walk
2008/04/02 05:52:27      2008/04/02 07:49:21      walk
2008/04/11 12:14:37      2008/04/11 13:32:05      bus
```

Project Deliverables

Project Proposal. As the name suggests, the project proposal contains the proposed research study with relevant and sufficient background information, which includes a summary of the existing related works and usable dataset examples. The proposal document should also contain the reason behind the topic selection and its relevance to the current trends.

Project Introduction. The project introduction is the initial phase of this research. It involves creating a document containing the information related to the study, such as the facts relevant to the topic, justification for choosing the proposed research topic, proposed approaches, background information of related existing works, and an overview of project requirements and deliverables. Background information includes a technology/solution survey and a literature survey of existing research.

Project Management Tool. A project management tool helps keep track of the tasks involved in different phases, relationships between the tasks, resource allocation, and progress tracking. It also involves the implementation of CRISP-DM to understand and implement project phases. Project management tools play a significant role in ensuring the project is right on track.

Work Breakdown Structure. Work breakdown structure involves breaking down the work associated with project completion into smaller tasks to make it manageable and easier to track. It will break the work into five successive phases that must be completed.

Effort Estimation. Effort estimation contains the forecast of the efforts required to complete the project. Effort estimation plays a significant role in the successful completion of the project. The forecast will contain the estimates regarding the overall cost and expenditure plan, hours required for the completion of tasks, and allocation of resources.

PERT Chart. The PERT chart is a tool used to allocate tasks required for project completion. It also lists the individual tasks and their dependencies. The PERT chart provides an illustrated representation of the project's timeline. It assists the project manager in creating the breakdown of distinct tasks and follows up with the task's progress until completion.

Gantt Chart. Gantt chart is a tool for quick monitoring project schedules and allocation of resources. It will present the schedule of tasks, deliverables, and persons responsible.

Data Management Plan. The data Management Plan summarizes the type of data used and the steps involved in data collection, organization, and storage. It also contains the steps to be followed so that compliance regulations for data security are respected.

Data Collection Plan. The data collection plan specifies procedures for collecting data from apt sources as the research problem requires. The plan specifies collection procedures, schedules, and persons responsible.

Project Management Plan. The project Management Plan provides guidelines for executing and monitoring the project. It also includes the plans to handle the risks that could occur. It contains many features that could act as a baseline, such as information regarding the methodologies used, resource allocation, in-scope and out-of-scope segregations, and organization planning. It will also state the project schedule with PERT and Gantt chart.

Data Exploration Plan. The data exploration plan defines procedures for preparing data for model input. The raw data extracted from the source is often unclean and requires a particular set of procedures to transform it in a way that could be helpful to attain insights. The plan specifies data cleaning, validation, and transformations required for training, validation, and test datasets.

Data Engineering. Data engineering involves the steps such as data collection, pre-processing, transformations, and preparations. It helps extract the relevant features, and the resultant data set often tends to be a cleaner and standardized version that could serve as a better input to the models. It will also aid in summarizing project input data statistics and analytics, contributing to a better understanding of the features.

Paper Abstract. The paper abstract provides a short summary of the entire project and its results. It also includes the research problem, methodologies used, and implications. Furthermore, it also helps comprehend the subject matter in a short span. A good abstract will help the study gain immense attention.

Project Presentation. The project presentation is an informative deck of PowerPoint slides that help present the research problem, information regarding the dataset, methodologies used, results, limitations, and future enhancements. It acts as a guide for the researcher while explaining.

Individual Research Paper on Model Development. The individual research paper will be the documentation that digs deep into the distinct model developed by each research team member. The paper will cover the implementation and justification of distinct algorithms chosen by the individual, the evaluation methods used, and the results.

Final Group Report. The final group report will contain the team and individual efforts altogether. It is a document that encapsulates the entire process from scratch to the end. Table 3 shows the detailed list of deliverables.

Table 3

Project Deliverables List

Deliverable	Description	Due Date
Project Proposal	A document describing the proposed project	21-Sep-22
Project Introduction	Documentation of the study's initial phase	28-Sep-22
Project Management Tool	Schedule tasks and implement CRISP-DM	3-Oct-22
Work Breakdown Structure	Breakdown of project tasks	3-Oct-22
Effort Estimates	Forecast of effort required	10-Oct-22
PERT Chart	Pictorial representation of project timeline	10-Oct-22
Gantt Chart	Visual representation of project timeline	10-Oct-22
Data Management Plan	Plan to manage data, maintain its security, and respect compliance regulations	17-Oct-22
Project Management Plan	Task allocation and guidelines for successful project completion	17-Oct-22
Data Collection Plan	Procedures for collecting data	24-Oct-22
Data Exploration Plan	Plan for steps involved in data preparation	24-Oct-22
Data Engineering	Carrying out the data preparation steps	31-Oct-22
Paper Abstract	A short summary of the entire project	21-Nov-22
Project Presentation	Presentation of the research project	28-Nov-22
Individual Research Paper on Model Development	A research paper on the distinct model developed by each team member	2-Dec-22
Final Group Report	A report encapsulating the team and individual efforts covering all the tasks mentioned above	9-Dec-22

Technology and Solution Survey

GPS Trajectory Analysis has been a hot topic recently, especially with the exponential boost in the data collected using GPS-supported devices. Multiple works are getting published on attaining insights from the collected data. Hence, to understand the existing approaches and their limitations, performing an in-depth analysis of such related works is necessary. Some of such works are discussed in this section.

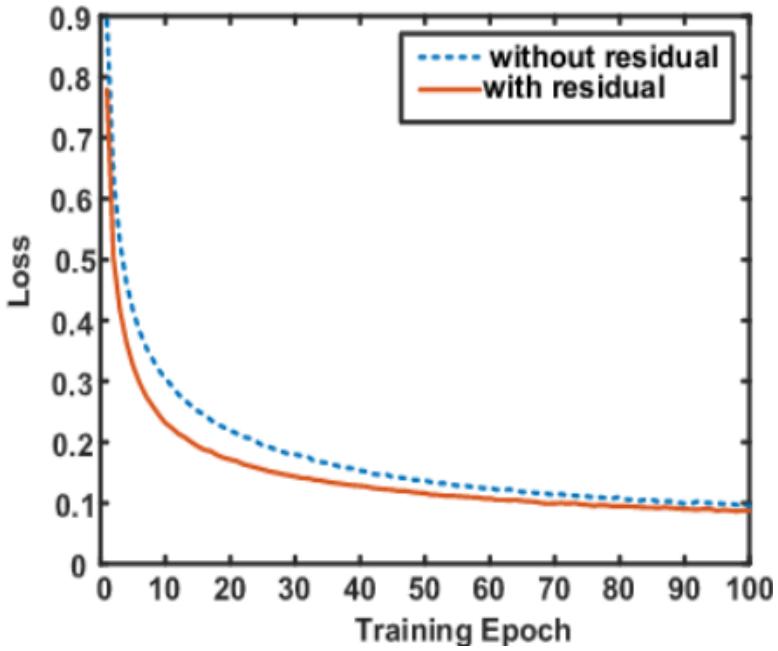
Wang et al. (2021) proposed a machine learning algorithm that uses the sensors incorporated in commercial smartphones to identify patterns and also help deal with the limitations regarding the accuracy levels of previously existing works. The preprocessing technique, such as removing unclean data and transforming the data into uniform matrices, followed by using a sliding window to perform batch normalization and segmentation to lessen the impact of outside noises. The model for transportation mode detection proposed a convolution operation in residual units applied after preprocessing of every individual sensor to accelerate the feature learning process and lessen the degrading influence of several hidden layers. The first of these model's layers is a multimodal input layer where all the preprocessed sensor data is sent to the model, called tensors. Secondly, a residual layer is the first feature learning of the data from the sensors, influencing the rest of the network.

Thirdly, a convolutional layer adopts a CNN (Convolutional Neural Network) to enhance the model fitting, fasten training and decrease the amount of data. Fourthly, an LSTM layer analyzes the relationship between data flow and the data in a given time frame to explore its long-term features. Fifthly, an attention layer learns from the previous layer to increase the weights of more significant features and time steps for better recognition performance. Lastly, an MLP layer uses the previous layer output to perform the last feature learning process. Figure 13

below shows the convergence rate of the model with residual and without residual networks, which indicates that the efficiency of the convergence rate is improved when using residual networks (pp. 1, 3-5, 7, 12).

Figure 13

Convergence Rate of Model



The experiment results helped confirm the proposed algorithm's scalability and prove that it outperforms the other baseline algorithms. The algorithm's limitation is that it is yet to use data from individual sensors separately and enhance the accuracy. Future enhancements can involve the implementation of a client/server architecture (p. 12).

F. de S. Soares, Salehinejad et al. (2019) proposed a recurrent neural network algorithm to identify the travel modes since most previous works rely on machine learning but not RNN combined with domain feature extraction. The study is conducted by using several samples for each mode and user. A sampling technique called synthetic Minority Over-Sampling (SMOTE) is applied to the training set for handling any bias in the samples. In total, 16 sensors (both

physical and virtual) were used for the experiment. After extraction, the feature selection and classification were conducted using Random Forest with recursive Feature Elimighion (RFE) and Mutual Information for classification. The extracted features are used to perform the comparison using 10-fold cross-validation with other baseline ML approaches such as Naive Bayes (NB), Support Vector Machine (SVM), Random Forest (RF), and Decision Tree(DT). The evaluation of the suggested model is done with various configurational settings, such as with and without dropout. Each configuration layer differs based on the number of hidden layers and units per layer. Figure 14 retrieved from F. de S. Soares, Salehinejad et al. (2019),shows that the AutoML, RF, and TMD-LSTM show a high accuracy, precision, recall, and f1-score, nearly 90% (pp. 1, 3-6).

Figure 14

Performance of the ML Model

AVERAGE ACCURACY (AC), PRECISION (PC), RECALL (RC), F1-SCORE (F1), KAPPA COEFFICIENT (KP) AND MODEL SIZE (MS) OF PER-FOLD OBTAINED FOR EACH ML ALGORITHM (ALG) TESTED.

ML ALG	AC	PC	RC	F1	KP	MS
AutoML [29]	**96%**	**95%**	**95%**	**95%**	**95%**	**117.9 MB**
NB	54%	50%	52%	47%	41%	10.5 KB
AB	76%	70%	75%	71%	69%	69.5 KB
SVM	74%	75%	69%	74%	65%	13.8 MB
DT [30]	**91%**	**87%**	**88%**	**87%**	**88%**	**159.9 KB**
RF [30]	**95%**	**93%**	**93%**	**93%**	**94%**	**3.6 MB**
KNN	87%	81%	86%	83%	83%	41.2 MB
SFNN	84%	84%	84%	84%	80%	100.2 KB
DFNN	83%	83%	82%	83%	78%	114.0 KB
LR	26%	26%	28%	23%	09%	5.8 KB
TMD-LSTM	**90%**	**90%**	**90%**	**90%**	**87%**	**314.5 KB**

The methodology used is a deep RNN architecture with LSTM cells using a wide range of sensor data. The experiment proved that the model proposed reaches a 90% accuracy while

simultaneously being lightweight (less than an MB). Though the prediction accuracy is high, this work has a limitation: the car, train, and bus modes are not generalized better than the still and walking patterns. The project can be enhanced further by detecting offline/online travel modes (p. 6).

Gao et al. (2020) devised a method to detect the travel mode using network data and geographic information system (GIS) information on, for instance, the public transport network, which is rarely explored, unlike GPS Data. The noise in the trajectory points was dealt with by removing cell towers and GPS data with an accuracy of not less than 80 meters and transforming that data to Mercator projection from WGS-84 coordinates to extract features. The methodology used is a criteria-based random forest algorithm; using this impact of public transport on automobile travel is also explored. Two variations were used to compare the experiment result; the first model has two components: a normal RF model for training and predicting. The second model uses a criteria-based method added to components of the first model, which aims to enhance the output of the prediction (p. 1).

Figure 15 below depicts the prediction accuracy for the two models retrieved from Gao et al. (2020). The prediction involves classification into four different travel modes. The overall accuracy is better for transportation modes, and the Random Forest model has an accuracy of around 94% in both rush and non-rush hours. In addition, it is interesting to see that while comparing the results, the accuracies of the BUS mode and CAR mode were enhanced during both hours. The study results show an improved efficiency in differentiating the modes, especially in a heavy traffic environment. Though helpful, the cost of transport network modeling and analysis is a limitation. The proposed work can be enhanced by breaking down the modes based on the vehicle type (pp. 1-3, 6).

Figure 15

Model's Prediction Accuracy

Testing Dataset		Model 1 / Model 2				
		Walk	Bike	Bus	Car	Accuracy (%)
Rush hours	Walk	17/--	1/--	0/--	0/--	94.44/--
	Bike	2/--	39/--	1/--	1/--	90.70/--
	Bus	2/2	6/6	115/129	23/9	78.77/88.36
	Car	0/0	4/4	21/5	111/127	81.62/93.38
Non-rush hours	Walk	18/--	1/--	0/--	0/--	94.74/--
	Bike	2/--	41/--	2/--	0/--	91.11/--
	Bus	1/1	8/8	119/126	17/10	82.07/86.90
	Car	0/0	4/4	19/9	114/124	83.21/90.51

Wang et al. (2018) suggested that the accuracy of travel mode detection using various methods ranging from criteria-based to machine learning could be improved by addressing the limitations of some existing works. Some of those limitations are limited sample size and inadequate feature selection. A random forest classifier algorithm was proposed with extended feature selection from 22 variables and six travel modes. A model was proposed with set rules to detect subway trips due to very little GPS data that can be obtained compared to the other modes of transportation and using different thresholds to calculate the critical distance. The second model is used for the other modes considered in the paper, which are classified to obtain the optimal feature set. Techniques such as Filter and Wrapper as attribute evaluators combined with three searching strategies, Bestfirst, Greedystepwise, and Ranker, are used. Figure 16 shows the accuracy comparison of the different algorithms (pp. 1, 4-5, 7, 10).

Figure 16

Comparison of Accuracy for SVM, ANN, and RF Algorithms

Classifier	Testing Set(914 instances)	
	# of correctly classified	Accuracy (%)
ANNs	779	85.23
SVM	728	79.65
RF	851	93.11

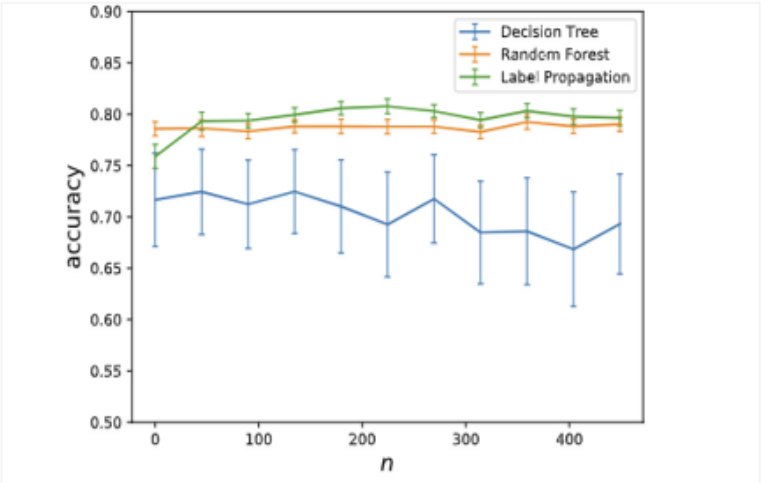
The experiment results showed that the random forest classifier had the highest accuracy among the testing set of classifiers. The additional attributes contributed to the influx of overall detection accuracy by 3.83%. The limitation of this approach is that the mode BUS is detected less accurately than other modes due to the lack of public transportation data. The proposed method can be enhanced further by adding GIS Data (p. 11).

Rezaie et al. (2017) observed that most of the existing travel mode detection methods depended on fully-validated data. Hence, a new approach was proposed using the semi-supervised method to use both the validated and un-validated data. There are three pieces of information used in this approach for data. The first is GPS data for user trips, the second is General Transit Feed Specification (GTFS) data, and the last is a city zone map dataset. The collected GPS data are preprocessed to derive the trip information. The feature vector proposed here has five components: total trip travel time, average trip speed, trip length, the minimum distance between the trip origin and the closest transit stop, and the minimum distance between the trip destination and the nearest transit stop. Two methods are proposed: supervised methods, DT and RF, and one semi-supervised method, Label Propagation (LP). These models were estimated with different meta-parameters to identify the best one. DT mode detection uses a

recursive partitioning in which the partition of the training set was done based on the thresholds of the features. A reduced pruning approach is also considered to avoid over-splitting. In RF mode detection, each tree contains the most suitable split, derived from a subset of candidate features, which minimizes the effect on variance. In LP mode detection, label propagation along with KNN was used. KNN approach uses the Euclidean distance to measure the similarity. The experiment was performed with a different ratio of unlabeled data to ensure the stability of the result. Each suggested model is compared with the other with a meta-parameter setting. The meta-parameters for DT CP are 0.01, and the minimum number of observations for pruning is 20. The meta-parameter for RF is ten classifiers. For LP, the clamping factor is assigned to one, and the nearest neighbor K is set to be below 5 for labeled data and above 5 for most unlabeled data. Figure 17 shows that the number of unlabeled observations does not impact the average accuracy of the supervised models. In contrast, the accuracy is improved for the semi-supervised algorithm as n increases (pp. 1-4, 6-7).

Figure 17

Accuracy of Supervised Model



Comparison of average accuracy of the models for $m = 110$ and different n: DT with $CP = 0.01$, RF with 10 trees, fully clamped LP with $K = 15$

The proposed methodology uses a decision tree, random forest, and KNN kernel. The main focus of the study was comparison instead of improvement. The results showed that when the dataset contained more than 70% of unlabeled data, the semi-supervised algorithm performed better and vice versa. Since this study is among the few initial studies involving semi-supervised data, there are a few errors in the estimations, which can be improved with a more detailed dataset. Future enhancements could be using a better dataset and feature selection (pp. 7-8).

F. de S. Soares et al. (2021) devised an approach to detect the travel mode using real-time data since most previous works rely on offline data. Since processing is done within smartphone devices, real-time detection could lower the cost and latency for ITS applications. Information will also be current, allowing for quicker action. The proposed approach involves developing an API called ActivityRecognitionAPI for travel mode detection, which uses supervised machine learning on real-time data from smartphone sensors. Rather than relying on noise removal techniques and expensive segmentation, travel mode was inferred from the features extracted from chunks of location traces every 90 seconds.

The hierarchical classification was used based on their previous approaches, which showed they led to better performance, differentiating motorized and non-motorized using Sequential Minimal Optimization (SMO) with SVM pre-trained. The SVM model is used to identify motorized vehicles; similarly, a Decision tree or Bayesian Network identifies the mode, such as walking or biking. Implementing the classification as mentioned earlier, SVM achieved an average accuracy of approximately 62.2%. Figure 18, retrieved from F. de S. Soares et al. (2021) shows a summary of each classifier's performance metrics (pp. 1, 3-4, 9).

The algorithm's performance is tested using a confusion matrix. The study results show that the ActivityRecognitionAPI has higher precision in recognizing vehicles. However, it has a

low recall, which is a limitation when the requirement is related to the quantity of the inference. The study can be enhanced further by including a vast range of users, modes including aquatic and airborne, and usage of different settings such as urban and rural (pp. 9-10).

Figure 18

Performance Metrics for each Activity Class

PERFORMANCE METRICS FOR EACH ACTIVITY CLASS RECOGNIZED BY THE ACTIVITYRECOGNITION API. *TP - TRUE POSITIVES, FP - FALSE POSITIVES, A - ACCURACY, P - PRECISION, R - RECALL, F1 - F1-SCORE

Class	TP	FP	A (%)	P (%)	R (%)	F1 (%)
Other	0	212	89.3	0.0	0.0	0.0
Tilting	0	243	87.7	0.0	0.0	0.0
Still	86	377	80.0	18.6	59.3	28.3
On Foot	471	91	80.5	83.8	61.6	71.0
Biking	0.0	41	97.9	0.0	0.0	0.0
Vehicle	442	16	67.5	96.5	41.3	57.9
Average/Macro	*166.5*	*163*	*83.5*	*33.1*	*27.0*	*26.2*

Su et al. (2016) proposed an algorithm to detect travel modes. The approach involves an android application with multimodality sensors and uses a GPS-and-network-free method, unlike the current practices, which often use GPS or fixed sensors for detection. The internal noise picked during data collection was addressed by setting a standard coordinate system to measure the X, Y, and Z axes from the phone to ensure the data was calibrated (p. 1).

The data collected was diverse in range and value; Normalization was performed to deal with it. The other techniques adopted to clean the data were winsorization to reduce the outliers present in the data and Gaussian smoothing to remove the fluctuation in data value. An SVM model between two hyperplanes was used to classify the multi-dimensional feature space. The hierarchical classification model, along with a rule-based classifier incorporated to categorize the data into wheeled and unwheeled in the first classification level, followed by the second for the two categories (pp. 2-4).

The unwheeled travel mode to identify patterns using acceleration features of the data, a simplified model of Decision Tree and Hidden Markov model was adopted. The wheeled travel mode uses a three-phase approach to classify the modes; namely, the data was split into three parts on a specific ratio training, add-up training, and testing. Training data for multiple users generate an initial model, followed by an add-up training set of a single user added to the model. It is tested at the end of each iteration. Figure 19, retrieved from Su et al. (2016), below depicts the classification results for the wheeled and un-wheeled modes, along with a confusion matrix for the further drilled classifications (pp. 1, 4-8, 12).

Figure 19

Classification Result based on Modes

<table>
<tr><th>First Layer</th><th>Accu -racy</th><th>Second Layer</th><th colspan="4">Confusion Matrix</th></tr>
<tr><td rowspan="4">Unwheeled Modes</td><td rowspan="9">100%</td><td>Classified as</td><td>Walk</td><td>Jog</td><td colspan="2" rowspan="3"></td></tr>
<tr><td>Walk</td><td>47</td><td>0</td></tr>
<tr><td>Jog</td><td>0</td><td>17</td></tr>
<tr><td rowspan="5">Wheeled Modes</td><td>Classified as</td><td>Bike</td><td>Bus</td><td>Subway</td><td>Car</td></tr>
<tr><td>Bike</td><td>23</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>Bus</td><td>0</td><td>14</td><td>0</td><td>0</td></tr>
<tr><td>Subway</td><td>0</td><td>0</td><td>27</td><td>0</td></tr>
<tr><td>Car</td><td>1</td><td>2</td><td>0</td><td>63</td></tr>
</table>

The methodology uses a hierarchical binary classification model to detect six travel modes. The study results show that the prediction of wheeled/un-wheeled modes, which is the first level of the hierarchy, has an accuracy of 100%. Further drilling down to the second level, the un-wheeled (walk and jog) and wheeled (bike, bus, subway, and car) modes have 100% and 97.1% accuracy, respectively. The model's limitation is that it is practical only in realistic

applications since offline data will increase the energy concerns and server response speed. Future enhancements can include detecting complicated travel modes, trip-based prediction, handling sensor failures, road safety, and usage of other environmental sensors (pp. 12-13).

Zhu et al. (2021) proposed a model to detect the light-duty vehicle's (LDV) travel mode, which, unlike most previous works, concentrates on training sets highly imbalanced to the point that it is impossible to deploy machine learning algorithms. The training and validation involve real-world, large-scale datasets, and the data was filtered to remove the outliers for nonrepresentative trips. Downsampling was performed to ensure the frequencies were consistent, and feature extraction took significantly less time. The data were reduced to a set of attributes and features as time series data are varying lengths and in a sequence. A set of detailed feature extraction processes was carried out. First, a python package tfresh extracts the trip feature, which uses a continuous function to calculate the distinct characteristics of the speed traces. Secondly, the 216 distinct, descriptive features obtained from the previous step are reduced dimensionally to 132 features for classification. Lastly, the features obtained from the above steps using zero mean and unit variance scaling are normalized to increase the performance. The proposed model employs one-class support vector machines (OCSVMs) for novelty detection and brand-new exhaustive feature extraction (EFE) methods on continuous time series data. Figure 20, retrieved from Zhu et al. (2021), shows the comparison accuracy achieved by different models used in this paper (pp. 1, 4, 7).

The result of the study showed that, with the extensive list of features added, the regular SVM (accuracy -99%) outperformed the proposed OCSVM (accuracy-92%) and the SVM (accuracy-97%) with just three features. The limitation is that the proposed method works better only with large-scale homogeneous data. It does not hold its fort against a conventional SVM

when dealing with well-sampled, balanced data. The studies can be improved by incorporating multimodal travel detection scenarios when a sufficient dataset is available (p. 7).

Figure 20

Performance Comparison of OCSVM and Regular SVM Model

PERFORMANCE COMPARISON OF OCSVM AND REGULAR SVMS FOR LDV MODE CLASSIFICATION

	Accuracy	AUC
EFE-OCSVM	0.92	0.92
Regular SVM (full features)	0.99	0.99
Regular SVM (three features)	0.97	0.97

Xaio et al. (2019) proposed a joint approach to explore machine learning classification and rule-based methods, unlike most works that solely concentrate on either. The proposed methodology involves a two-stage approach, the first using rule-based methods to detect subway modes and the second being a Gaussian process classifier (GPC) to sort out the remaining travel modes. The paper uses segments to detect subway patterns using a grid search algorithm to establish data mining rules. Parameter optimization exhaustively searches for the best combination through a subset of parameter space. The feature related to speed is used to differentiate the car and bus modes, and two consecutive points are calculated to find the average heading change. The data ratios were considered to ensure walking and e-bicycle to avoid low detection prediction. Two rebalancing methods, cost-sensitive learning and ensemble learning, were used to obtain segments for each mode. The resampling technique addresses the undersampling of modes by eliminating the majority of mode segments. Figure 21, retrieved from Xaio et al. (2019), below shows the transportation mode detection accuracy for the chosen models (pp. 1, 4-5, 9).

Figure 21

Mode Detection Accuracy Comparison

Comparison of detection accuracy (without subway).

Methods	Walk-based training subset			Walk-based test subset		
	Sample size	# of segments correctly detected	Detection accuracy (%)	Sample size	# of segments correctly detected	Detection accuracy (%)
SVM	8,635	7,699	89.16	2,875	2,500	86.96
MNL	8,635	7,089	82.10	2,875	2,282	79.37
BN	8,635	7,806	90.40	2,875	2,533	88.10
ANN	8,635	8,065	93.40	2,875	2,608	90.71
GPC	8,635	8,101	93.82	2,875	2,675	93.04

The test result shows that the two-stage method had the highest classification accuracy (93%) compared to the other tested models and presents the most preferred generalization. The limitation of the proposed method is that it is suitable only for places where the subway segment accounts for a more significant part of the GPS travel survey. Future enhancements can be focused on improving the low precision rate in detecting e-bicycle segments (pp. 1, 9-10).

F. de S. Soares, Revoredo et al. (2019) proposed an algorithm that considers both travel mode and trip purpose detection as a combined unit since most of the existing work tackle either of the two but not jointly, which leads to redundancy in the data preparation steps. The data preprocessing implemented in this research paper transformed the JSON file into a flat table, and filtering techniques were deployed to process the location traces to avoid noise in the data. The trip segments were extracted for each device segmented with a constrained time segment. The following steps were to extract features, analyze the correlation using Pearson Correlation Test, and the data was normalized. Following the previous step, the data was partitioned and resampled to remove any bias from the unbalanced dataset.

The research paper applied two AutoML techniques to determine the best ML configuration. The methodology used in this approach incorporates multiple preprocessing steps; the first is to extract the features for classification, followed by automated machine learning

methods, which will help identify the most suited classifier and its hyperparameter configurations. Figure 22, retrieved from F. de S. Soares, Revoredo et al. (2019), below shows the performance metrics of travel mode and trip purpose classification. The study result showed that the most suited machine learning algorithm is Random Forest in most cases, and Up-sampling is the best resampling technique (pp. 1, 4-6, 7).

Figure 22

Performance Metrics of Trip Purpose and Travel Mode Detection

AVERAGE PER FOLD PERFORMANCE METRICS OF TRAVEL MODE DETECTION ON THE 10-FOLD CROSS-VALIDATION

Sampling Technique	Classifier	Accuracy (%)	Precision (%)	Recall (%)	F-Measure (%)
None	Support Vector Machine	76	76	76	76
Up-sampling	Random Forest	88	88	88	88
Down-sampling	Decision Tree	70	70	70	70
SMOTE	Ada Boost	83	82	83	82

AVERAGE PER FOLD PERFORMANCE METRICS OF TRIP PURPOSE PREDICTION ON THE 10-FOLD CROSS-VALIDATION

Sampling Technique	Classifier	Accuracy (%)	Precision (%)	Recall (%)	F-Measure (%)
None	Random Forest	69	69	69	69
Up-sampling	Random Forest	81	80	81	80
Down-sampling	Random Committee	67	66	67	66
SMOTE	Random Forest	77	77	77	77

10-FOLD CROSS-VALIDATION PERFORMANCE METRICS OF THE CLASSIFIERS USED IN THE CITYTRACKS-RT APPLICATION

Classes #	Classifier	Accuracy (%)	Precision (%)	Recall (%)	F-Measure (%)
5	Multilayer Perceptron	51	48	51	48
2	Support Vector Machine	68	47	68	55
4	Decision Table	77	76	77	76
2	Bayesian Networks	91	88	91	87

PERFORMANCE METRICS OF THE CLASSIFIERS USED IN THE FIELD TESTS OF CITYTRACKS-RT APPLICATION

Classes #	Classifier	Accuracy (%)	Precision (%)	Recall (%)	F-Measure (%)
4	Multilayer Perceptron	34	31	28	29
2	Support Vector Machine	58	61	54	57
4	Decision Table	59	40	42	41
2	Bayesian Networks	88	44	50	47

The study result showed that the most suited machine learning algorithm is Random Forest in most cases, and Up-sampling is the best resampling technique. It also proved that the proposed solution could detect travel mode and predict trip purpose using just 60-second location data collected from a smartphone. The limitation is that during trip purpose prediction, the samples of two categories with similar features often get mixed up. Future enhancements can be done by adding other sensor data to improve performance. The enhancements can also be made to the prototype by running it through different intensive field tests (p. 9).

During the technology and solution survey and looking into the related reference papers in-depth, certain similarities and contrasts were observed between the works. Some of the observations will be discussed in the following sections.

Similarities between the Related Works

The research papers provided insights into the data sources, algorithms, methodologies, architectures, and the ideal model relevant to the scenario. In addition to the insights, resemblances were observed between the various works. These similarities have helped identify the persistent hurdles that might occur and also aided in cultivating the optimal workaround to avoid such issues. Some of the parallels observed are discussed in this section. Firstly, the dataset data source used in the existing works. From the works of Wang et al. (2018) and Rezaie et al. (2017) it was observed that the data source was a GPS Sensor. The likeness in the data source and the various methodologies used in each of those papers helped derive the optimal way to detect travel modes. The second is the parallels between the papers related to the types of algorithms used. Random Forest, Decision Tree, and SVM were found to be the most commonly used algorithms in the works of Asad et al. (2019), Rezaie et al. (2017), and Gao et al. (2020). F. de S. Soares,E., Salehinejad et al.(2019) used RNN in conjunction with LSTM. And the work of

Wang et al. (2021) showed the use of LSTM approach. Thirdly, the studies by Su et al. (2016) and F. de S. Soares et al. (2021) scoured further into the concept of offline and online data as data sources. Such papers helped with the handling of real-time data as well as understanding the limitations associated with it. The next similarity between these works is related to the limitations; the number of travel detection modes is limited in most cases and can be enhanced by adding extra travel modes involving aquatic or airborne data sources. The limitations also extend to the individualistic architecture of works. Enhancements can be done by invoking client-server architecture.

Contrasts between the Related Works

The background research also showed the contrast apart from the similarities mentioned in the previous section. These discrepancies showcased how each paper had an alternative approach to solving the problem. Some of those contrasting points are associated with the data sources, algorithms, and methodologies preferred by each work. Interestingly, in some cases, a research paper's limitations were addressed in another paper. Some such observed contrasting opinions are discussed here. Firstly, the discrepancy occurs in the choice of the data source preferred. Wang et al. (2018) preferred the usage of data collected using GPS Sensors, whereas Gao et al. (2020) relied on the GIS data and the data from GPS Sensors. Gao et al. (2020) explored the usage of public transport data recorded using various means.

The second contrast between the papers focuses on the algorithms chosen to achieve the optimal result. Some of the algorithms preferred are Random Forest, Decision Tree, SVM, and LSTM. Wang et al. (2018) , Gao et al. (2020) preferred random forest, Zhu et al. (2021), Asad et al. (2021) used SVM. On the other hand, F. de S. Soares, Salehinejad et al. (2019) ,Wang et al. (2021) utilized LSTM. It is also worth noting that, though this may seem similar between a few

works, it can also be considered a contrast when compared with other works out of the cluster. The third contrasting opinion is related to the total number of features extracted. Some papers preferred limiting the number of features, whereas others opted to enhance the selection by increasing the total features chosen. F. de S. Soares, Revoredo et al. (2019) and Xaio et al. (2019) used different set of features. had 22 features extracted, whereas the other had only 6. The contrast also arises from the choice between the offline and online modes of the data source used by the various papers. Su et al. (2016) preferred offline data whereas F. de S. Soares et al. (2021) relied on online real time data.

Literature Review

Li et al. (2021) suggested detecting transportation modes using GPS trajectory data and GIS information. Unlike other previous works, the recommended method uses only limited GIS information and does not use any user's personal or other sensors' data. The result reveals that the RF algorithm delivered the highest accuracy of 91.1%. Furthermore, four GIS features have improved accuracy, particularly for bus and subway travel modes. This study shows a way to use limited information to detect travel modes with high accuracy (pp. 1-2, 4, 7, 11-12). Liu et al. (2019) suggested an RNN-based approach, Spatio-Temporal GRU(ST-GRU), that considered spatial and temporal trajectory information simultaneously, unlike other works using this information individually. The result compares the proposed method with other baseline ML algorithms using three types of datasets. It proves that ST-GRU performed better than the others in all three cases (91.25% for Geolife, 93.89% for Shanghai, and 93.21% for the synthetic dataset), and considering both spatial and temporal information improved the accuracy. This study shows that the effective use of additional information, like spatial and temporal, helps achieve superior model accuracy (pp. 1-2, 5-6).

Tišljarić et al. (2021) suggested using origin-destination (OD) data retrieved from cellular network data for travel mode classification, unlike the previous works where the data is collected by survey (either offline or online). The focus was on comparing the performance of existing most used ML algorithms and finding the best-fit one for classification. The result shows that RF outperforms other ML algorithms with the highest accuracy of 99.35%, followed by KNN and DT with an accuracy of nearly 98%. This paper helps show an efficient way to use cellular data and mode classification using various baseline algorithms (pp. 1, 3-4). Wang, Yang et al. (2020) suggested an approach for identifying and classifying anomalous trajectory patterns. The evaluation of the proposed method is done using cab trajectory data, and the methodology suggests metrics to calculate the distance between two trajectories. This paper is helpful as it helps in identifying the classification of real-world trajectory data. The study's results showcased that car is the primary mode of preferred travel at 40%, followed by public transport at 25%. The survey consisted of over 600 plus valid observations. (pp. 1, 4, 12).

Brunauer et al. (2013) proposed an approach to using a comparatively large set of GPS-trajectory datasets describing the motion characteristics features with relative frequency distribution, leading to a better recognition rate without using any GIS information. The result shows that multilayer perceptrons (MLPs) and logistic model trees (LMTs) performed better than C4.5 trees. This study is beneficial as it involves mode detection using GPS-only trajectory data related to rural and urban areas. It was possible to attain an overall accuracy of 90% by selecting 10 out of the 54 available features, and accuracy spiked to 93% when 30 features were selected. The in-depth evaluation helped conclude that it is impossible to identify the optimal feature set since it constantly evolves. (pp. 1, 7). Lu and Xia (2020) proposed an approach to fully utilize the Spatio-temporal information and the features(high-level) extracted by neural networks, which

were not explored well by previous works. The approach involves utilizing state information (stop and turn) for spatial data, and time information is derived using Recurrent Plot (RP). In addition, a dual supervised autoencoder is suggested, which uses CNN to construct neural networks and to avoid a low classification accuracy, a PCC layer was recommended. The results show that the accuracy of 89.475% for the Geolife dataset and 89.602% for the SHL datasets. This study guides how to incorporate and utilize Spatio-temporal information for optimal trajectory classification (pp. 1, 6-7, 10, 14).

Patil et al. (2019) suggested an approach (GeoSecure-O) to calculate the distance and detect travel modes securely with better accuracy. The method proposed here ensures security by not revealing users' location data to cloud-based or location-based service providers. The result shows that this approach accurately calculates the distance and derives the necessary features for determining the travel modes. This study helps to guide how to detect travel modes while preserving user location privacy. After classifying, an accuracy of 0.815 was achieved. The features chosen from the GeoSecure-O method emit a classification accuracy of 0.806. The difference between both was recorded as 0.85 (pp. 1-2, 7-8). Asad et al. (2019) proposed a model to detect the efficiency and variation of transportation modes during the pandemic using six ML algorithms. The results showed that the travel dynamics were shaped with an accuracy of 91.21% for SVM and 84.5% for Random Forest. The study provided insights into the data processing and selection techniques that can play a vital role in classifying travel modes (pp. 1, 5).

Wang, Yuan et al. (2020) explored to study the travel mode choices made by tourism using cell phone signaling data. The model was designed to find the starting point of travel using a time-space threshold algorithm, and a fuzzy identification theory was used to classify the transportation mode. The study provided insights into the different approaches to how time-space

from the data can be used to identify the starting location and different methods that can be adopted to identify the transportation mode. The study results proved that the proposed approach achieved a high macro F1-score and F1-score on a global detour and global shortcut datasets at greater than 0.886 and 0.929, respectively, when compared to the local detour and local shortcut datasets (pp.1, 5-6). Zhu et al. (2010) suggested a neural network model identify the travel mode choice of residents during the spring festival. Non-linear and linear indicators will be identified by the neural network, which predicts the mode based on the Elman network. The result shows that the visitors chose other modes instead of railways, and the model could classify the modes used, which had a prediction accuracy of 91%. The methodologies used in this study aided in understanding how linear and non-linear indicators can be used to identify the multi-mode utilized (pp. 1, 4). Su et al. (2016) suggested using a GPS-and-network-free method for prediction. The study aided in realizing how hierarchical binary classification can be used to predict. The study proposed the utilization of hierarchies, and the accuracy attained by the first layer was 100%, which was a binary classification of wheeled and unwheeled. The second level of the hierarchy reached an exceptionally high accuracy of 97% (pp. 1, 13).

Wang et al. (2021) suggested that the issues regarding the accuracy levels of previous works can be improved by using a long short-term memory network(LSTM). This work helped in learning how LSTM works while detecting travel modes. The accuracy attained by the model proposed in the study was 98.29%, 96.22%, and 98.44% for each of the datasets used. The model recognized the modes with a precision of over 95% accuracy, and the algorithm's accuracy was over 90%(pp. 1, 12). F. de S. Soares, Salehinejad et al. (2019) suggested detecting travel mode using a wide range of sensor data. The method incorporates deep RNN with LSTM cells, which helped understand feature extraction and deep RNN with LSTM. The proposed model could

detect the various travel modes with an accuracy of up to 90%. The model also performed surprisingly well with an exponentially lower member comparison and lesser computational cost. The average accuracy, precision, and f1-score all were examined to be over 90%. The kappa coefficient was also higher or equal to 90%. (pp. 1, 5-6).

Gao et al. (2020) explored the possibility of utilizing network data and geographic information system(GIS) information on a criteria-based random forest algorithm to predict travel modes. The study explained how different data sources impact the resultant prediction. The findings of the proposed approach illustrated that the information regarding public transport could help determine the differentiation between the motorized models. By utilizing the public transport network data, the accuracy of predicting buses and cars improved by more than 10%. When the transportation modes involve unstable environments, the survey using GPS-based information helped. This study proved that the public transportation network information was indeed helpful in enhancing prediction accuracies.(pp. 1, 6). Wang et al. (2018) addressed the limitations related to accuracy in some existing works by introducing an extended feature selection method involving 22 variables. The study provided insights into how the concept of feature selection plays a vital role in influencing the final prediction. The study's results portrayed that more than 98% of the subway travel mode was identified precisely. In contrast, the overall accuracy accumulated by the result of the five modes reached a peak of 93.11%. The precision of the prediction of modes was high except for the mode bus. The ROC curve proved that the model possessed an excellent classification capacity, and the best-suited model was identified as Random Forest. (pp. 1, 11).

Rezaie et al. (2017) proposed using both validated and un-validated data while predicting travel modes. The approach involves a decision tree, random forest, and KNN kernel. The result

provided insight into how the proposed approach is practical when dealing with a higher concentration of unlabeled data in the dataset. For comparison, a portion of the training set was void of labels, ranging from 0% to 85%. Best-suited model was identified as Random Forest while including the label observations. The LF model could outperform the Random Forest only in scenarios including more than 80% of unlabeled data. (pp. 1, 7). The study of the approach proposed by F. de S. Soares et al. (2021) helped attain insights into the pros and cons of using real-time data while predicting travel modes using supervised machine learning algorithms. The modes involving the categorization 'On Foot' had a decent precision, recall, and f1-score of 71%. In contrast, the Activity Recognition API could have showcased a satisfactory performance. The result also indicated that the model could predict the travel modes recorded through smartphones in city scenarios. (pp. 1, 10). Zhu et al. (2021) study provided insights into how real-world, large-scale datasets can be dealt with using different models, aided by extracting an extensive set of features. The result of the proposed method has been promising, lying within the range of 92% and 94%. It was concluded in this study that approximately 5000+ training samples were satisfactory for developing a decent working model. The theoretical baseline of the SVM model was established only with three features. The trained model was tested on a uniform test set to maintain consistency (pp. 1, 7).

Xaio et al. (2019) study explored how joining both rule-based methods and machine learning classification can be used to predict. The study proved helpful in realizing how rule-based and Gaussian process classifier models work. The proposed model opted for utilizing a feature set consisting of the feature's average acceleration, heading change, average speed, and low-speed point rate. The percentage of subway mode records identified correctly was 97%, and similarly, the walk-based models were identified with an accuracy of 93.04%. Thus the study

concluded that a good feature selection could result in an optimal outcome. (pp. 1, 10-11). F. de S. Soares, Revoredo et al. (2019) study demonstrated how to combine travel mode and trip detection into a single unit. The study methodologies also enabled attaining knowledge of the required preprocessing steps. The study results with smartphone data attained an accuracy of 88% at the maximum and 81% at the average. The CityTracks-RT solution achieved better accuracy with non-motorized modes for the Bayesian Network model at 91%. The multilayer perceptron recorded the worst performance at 50%.(pp. 1, 9).

Similarities between the Related Works

The works explored as part of the literature review provide insights into using different algorithms, data features, and the inclusion of additional information. The similarities found in the papers show which approaches and data sources are adequate and primarily used while considering travel mode detection and classification. Some of the similarities found are explained in this section. The first one is related to the data source being utilized. The works of Li et al. (2021) and Brunauer et al. (2013) showed the work relies on GPS data, GIS information, and cellular-network data. The second one is related to the algorithms used. Rezaie et al. (2017), Soares et al. (2019) and Zhu et al. (2010) used baseline ML algorithms like SVM, RF and neural networks are primarily used for model evaluation purposes. The third similarity includes spatial and temporal information for effective mode detection and classification. The work of Lu and Xia (2020) , Liu et al. (2019) utilized spatial and temporal information.

Contrasts between the Related Works

Although the works referred to here show some similarities, some distinctions are also identified. These differences mainly exist in the approach to achieving the goal, the number of features considered, and evaluation methods. These contrasting observations are discussed in

detail in this section. The first difference is the number of datasets considered for evaluation. Liu et al. (2019) used three datasets whereas Wang et al. (2020) relied only on the cell phone signaling data. The second one is related to how the features are extracted and used. While retrieving the data, the studies like Zhu et al. (2010) and Patil et al. (2019) differs from the way the users' privacy is maintained.

Furthermore, there is a difference in data utilization for various purposes such as works of Xaio et al. (2019) and Wang et al. (2018) . For example, one study focuses on detecting the anomalies present in the trajectory patterns, unlike others, where the main goal is identifying and classifying the travel modes. Moreover, the type of algorithms used differs from one work to another. In some, only baseline algorithms are explored, whereas, in others, some novel approaches using neural network is suggested. Wang et al. (2018) and Gao et al. (2020) works use the Random Forest model, Rezaie et al. (2017) and F. de S. Soares et al. (2021) research used the Decision Tree model, Zhu et al. (2021) and Asad et al. (2021) proposed the model implementation using Support Vector Machine and F. de S. Soares, Salehinejad et al. (2019) and Wang et al. (2021) approached the use of deep learning algorithm of LSTM.

Additionally, the features used vary from one to another. Some calculate the distance, acceleration, and speed from the GPS data, and some consider spatial and temporal information to improve the model's accuracy. The works that extracted additional features were from the works of Su et al. (2016) and Lu and Xia (2020). Lastly, some studies, such as Brunauer et al. (2013) and Wang et al. (2018) used to classify only single travel modes like subway, car, and bike. Still, the work of Tišljarić et al. (2021) were more focused on detecting the modes included in a trip chain, like a user can change the travel mode from bus to bicycle, and some works also consider those modes in the classification approach.

Data and Project Management Plan

Data Management Plan

A data management plan assists in organizing and planning the key activities associated with the data lifecycle. It aids in understanding various aspects of data, such as data collection methods, storage and management methods, and how the data can be used to gain valuable insights. An effective data management plan was created and explained in greater detail in the following sections keeping this in mind.

Data Collection Approaches

The dataset used in this research was a GPS trajectory dataset collected over five years, from April 2007 to August 2012. There are 182 users of the collected data by Microsoft. (n.d.). The number of trajectories collected in this process was 17,621, and it consisted of sequential timestamped points which contained information related to the latitude, longitude, and altitude. The data was collected using various GPS loggers and phones and sampled at multiple rates. This collection process involved two segments of users: one carried GPS loggers for years, and the other carried them for only a few weeks. The records logged in dense representation account for 91.5% of the dataset. Dense representation implied that the records collected were in an interval of five to ten meters or one to five seconds. The data was collected mainly from the cities of China. The dataset contains the logs of usual movements like going to work and home. Furthermore, the records also included other activities like entertainment and sports, such as shopping, dining, sightseeing, jogging, hiking, and cycling. The volume of the data was 1.67 GB. The format of the data collected here was in a vector-based plotter file for each GPS log based on the user's trajectory timestamp. The Microsoft Research License Agreement was accepted while collecting the data from the Microsoft site. As per the agreement by Microsoft. (n.d.), the data

could be used for non-commercial purposes, such as academic research. Some compliance guidelines include: not removing copyrighted content, distribution in any form not allowed, and prohibiting modifications. Furthermore, the patent rights were not applied to the derivatives of work, and the agreement is controlled by the laws of the State of Washington, USA (p. 4). These conditions were respected throughout the project duration.

Data Storage Methods

The GPS Trajectory data collected by Microsoft. (n.d.) was then stored for usage following the collection process. Two kinds of storage were employed; one was the local memory of the respective team member's systems, while the second was the google cloud storage. The dataset's compliance policies and ethics were respected throughout the model development process, including while storing the data. Access to google drive was provided only to the team members, and no outside interference was involved.

The original raw data consisted of multiple files in a vector-based plotter files (.plt) format, which were grouped into various folders based on the user id. The user id ranges from 000 to 181, implying that there were 182 folders available, each containing a Trajectory folder inside. All the trajectory folders inside each user id contained a .plt file, whereas a handful contained an additional format file (.txt). For instance, the folder path of the .plt and .txt file inside the user id 84 would be something like data/84/Trajectory. Inside the folder, the .txt and .plt files were present.

The next thing to observe was the nomenclature of the .txt and .plt files, which had a particular format. 'yyyymmddhhmmss' was that format which was nothing but the timestamp where 'yyyymmdd' represented the year, month, and date, followed by 'hhmmss' representing the time. A single comma-separated (.csv) format file was generated following the conversion of .plt

and .txt files and merging them into one—the nomenclature of the resultant .csv file was ProcessedGPSTrajectoryData.csv. The file was then placed into the same good drive inside a folder named 'processed_data.' Similarly, the training and testing datasets were placed inside the folder named 'Modeling_Dataset,' while the files were named 'TrainingDataset.csv' and 'TestingDataset.csv.' In case of any updations following data archival, versioning was enabled. The versions were named 'Backup_' followed by the nomenclature of the filename, followed by the archival date. For instance, a backup of ProcessedGPSTrajectoryData.csv was named Backup_ProcessedGPSTrajectoryData_yymmdd.csv and placed inside the 'Data_Archival' folder.

Data Management Methods

The data placed in the local client machines and google drive were protected using a strong password. Access to the downloaded dataset was provided only to team members, which was maintained throughout the project duration. Since the team was not the original owner of the raw data, it was not shared as a file anywhere. Only the link to the source, which is the site of Microsoft. (n.d.), was provided. The processed data was protected by not sharing with outsiders, and only the metadata and the transformation steps were published. The team paid utmost attention to adhering to the ethics and compliance policies enforced by Microsoft. (n.d.), which also included strong laws enforced to it. Some of the compliance policies laid out by Microsoft. (n.d.) were not removing copyrighted content, distribution in any form not allowed, and prohibiting modifications.

Data Usage Mechanisms

Project Report Documentation, PowerPoint Presentation, System Specific Documentation, and a defect tracker were prepared throughout the project duration.

Documenting every step of the way was essential to keep track of the progress, which will also come in handy when approaching similar circumstances in the future. These documents can also be helpful for future users to understand the usage of the models. Project report documentation contained the detailed steps taken, including some tasks such as understanding the business, data collection, preparation, modeling, and evaluation.

The user guide attached to the downloaded dataset provided metadata related to the raw data. Microsoft. (n.d.) provided detailed documentation regarding the fields present, the distribution of various kinds of data, the characteristics of the data along with compliance and ethics policies. The source's privacy policies were respected while downloading the dataset. Table 1 depicts the metadata of the user's log file, whereas table 2 represents the fields present in the transportation mode label files and the metadata.

Every team member was involved in every step of project completion, and everyone on the team was granted full authority over the data and the models. Everyone from the team had a hand in preparing the project report documentation. The ethics and compliance policies set up by Microsoft. (n.d.), discussed in the Data Collection Approaches, were respected throughout the project duration. Some compliance guidelines included not removing copyrighted content, distribution in any form not allowed, and prohibiting modifications.

An effective data storage plan was essential because of the sheer amount of raw data, which was 1.67 GB. The data storage plan should be capable of handling larger volumes as well as being efficient while doing so. Hence, it was stored in google drive, a part of the free student-licensed version, with unlimited capacity. While handling such vital data, it was mandatory to devise a robust backup strategy that could be used in unfortunate data loss situations. The backup strategies utilized were carefully considered.

Thus, google drive was utilized to store the data, which had a data storage backup plan, including the option to choose the desired file and folders manually for automatic backup. In addition, the data was also stored in individual client systems of the involved team members to ensure that a proper backup strategy was in place to avoid any data loss. In the event of accidental removals, the in-built restoration option provided by google drive came handy. The other way could be utilizing the backup available in the respective team member's systems. The storage and backup options ensured security.

All the team members were authorized to perform the recovery operations after consulting with other team members and attaining the team's approval. The various backup and restoration methods discussed ensured the preservation of data. The processed data was placed in google drive, and only the team members were authorized to access this. When publicizing the work, the link to the original data source was provided to meet the compliance policies laid out by Microsoft. (n.d.).

Project Development Methodology

A set of guiding principles is required for project planning, management, and execution. These guiding principles are known as a project management methodology. This methodology aids in the process of prioritizing the identified works and determining how each can be completed efficiently. The following sections go over the project management methodologies used in this project.

System Development Life Cycle

The Software Development Life Cycle (SDLC) defines a series of steps involved in the development and deployment of software. Singh (2009) defined SDLC as a proper framework for software engineering that is vital for building reliable, cost-effective software or an

application that is easy to maintain (Software Development Life Cycle Section, Para.1). Typically, it consists of seven phases: requirement, analysis, design, development, testing, deployment, and maintenance. There are different SDLC models, such as waterfall and agile, and each model follows a series of unique steps to ensure development success. Data science demands a solution-oriented approach to solving problems. In this field, data plays a vital role, and thus a data mining process assists in extracting patterns, finding relationships, and yielding actionable insights. This project used a methodology widely used in industry and research communities known as Cross-Industry Standard Process for Data Mining (CRISP-DM). The main benefits of this methodology are domain independence, flexibility, iterative approach.

Planned Project Development Processes and Activities

This section explains the planning of the project development process and the steps incorporated based on the CRISP-DM methodology.

Business Understanding. The business understanding was the first stage of the process, where an idea about the project's requirement was understood from a business perspective. Business understanding helps to attain insights into the goals and focus on determining the key factors that could influence the project outcome. The business understanding phase involved identifying project goals, background research, scope definition, requirement gathering, identifying data sources, and document preparation. Through each process, it was helpful to gain a more in-depth understanding of the project. The first step of business understanding was identifying project goals, which could be achieved through regular meetings with the stakeholders and brainstorming sessions among the members. The next step was intensive research on various existing studies and research related to the topic and methodologies. Furthermore, the background research helped realize the existing shortcomings and how this can

be improved and avoided in the current study. Scope definition helped define what items will be part of the project result and what not. The requirement-gathering step helped with accumulating software, hardware, and storage requirements that were vital for the execution. The questions for the stakeholders and associated teams helped further narrow the execution steps, and the data sources that suited the topic were identified. Finally, a project plan document with updates regarding progress and achievements after each phase was prepared. Documenting the progress made so far could serve as future reference material. Once comprehensive insights were attained from the business understanding phase, the next one began.

Data Understanding. Project-related data was collected in this phase based on the resources identified in the previous stage, which was achieved by referring to different sources. While referring to various sources, it was crucial to stay compliant with the policies, rules, and guidelines the data owner recommends. The first step was to review the ethics and compliance policies laid out by the original authors of the shortlisted data sources. After review, the dataset that would be used was chosen. The next step was to collect the data from the source without violating the ethics and compliance guidelines. It is necessary to understand the data to progress further; hence a meticulous exploratory analysis was performed. The properties of the acquired data, such as the size, format, fields, and corresponding type, were also identified.

Additionally, to address the data mining questions, it was necessary to explore the data to comprehend the distribution of essential attributes, the relationship between them, also the statistical analysis, and aggregation outcomes. Developing the data quality assurance plan helped maintain the data's integrity and handle any mishaps—verifying the authenticity and quality of the data, which cannot be ignored. The verification included identifying any incomplete, error, or missing data. The issues identified were rectified by performing an intensive data-wrangling

process which was performed in the next step. Apart from the mentioned issues, if any additional quality issues were specified, then a possible solution was suggested, which depended on data and business knowledge.

Data Preparation. Once a conclusive understanding of the data was achieved, the data preparation phase followed. This phase helped decide the data which ought to be used for further analysis. The decision criteria included quality, relevance, technical constraints, and inclusion or exclusion of any attributes. The data quality was raised by performing an intensive set of data-wrangling operations. Data wrangling aided in cleansing the data by performing tasks that eliminate corrupt data, redundancy and converting the data to be standardized. The existing features were identified, which was very helpful during model development.

Furthermore, apart from the features present in the processed dataset, additional calculated features that might be useful were identified, which was achieved either by aggregation or merging operations. Merging happened when multiple sources were presented with a common object, whereas aggregation occurred when a new value was calculated by summarized information. The processed dataset attained from this step was split into train and test datasets of the required proportion, which helped ensure the precision of the models developed in the following phase.

Modeling. After the data preparation phase, the split train and test data were passed through various ML techniques employed in the modeling phase. Several models were ideal for the project's goal based on the requirement, and the four most suitable ones were identified from those. A plan was devised to determine the initial architecture of how the model is supposed to be, and the next step was to build the models based on the devised architectural plan. In addition, the models were built with various parameter and hyperparameter settings such as kernel,

number of iterations, and metrics. There were possibilities of issues arising when developing the code for the model, and those issues were identified, and proper fixes were implemented. Those issues and fixes were tracked using a defect tracker. The models created were assessed from both project and business standpoints. The outcomes and quality of each model's outputs were measured, and the parameters were tuned if required. The assessment ran iteratively until the best version was identified.

Evaluation. The evaluation phase helped identify the degree to which the model meets the requirements. The accuracy and predictions resulting from the previous phase were validated in the evaluation phase. The evaluation approaches included various comparison techniques such as confusion matrix, ROC curve analysis, and accuracy comparison. The model's performance was also examined with the help of datasets of various sizes. Performance testing also included comparing memory usage, CPU utilization, and model size. The best-suited model was determined after comparing and evaluating the chosen models concerning the business success criteria. Additionally, this phase helped determine if any results generated in the modeling phase were not necessarily related to the original business objectives. Such identified results might help to derive insights that could help guide future works. Based on the assessment result, a decision was made on whether to proceed with the deployment or initiate further iterations.

Deployment. Based on the evaluation of results, a deployment strategy was devised in this stage. This plan included what the necessary steps were and how to perform those. Some factors were considered while deployment, such as the model size and packaging, retaining and versioning, and traffic routing.

Testing the model's performance in a production environment helped determine its efficiency under stressful circumstances since the production environment typically deals with a

higher volume of data due to its real-life implementation. Additionally, the change in data's volatility and volume may have a ripple effect influencing the precision of the model. Thus, a monitoring and maintenance plan needed to be identified and employed. These phases of SDLC were explored further in the following sections discussing the Work Breakdown Structure, PERT, and Gantt Charts.

Project Resource Requirements and Plan

A resource is a valuable asset in carrying out the tasks associated with a project in order to achieve its goal. Poor resource planning impedes a project's smooth progress, delaying delivery deadlines. As a result, an appropriate resource requirement plan was developed, and the necessary resources for this project, such as hardware, software were identified.

Hardware Requirements

The project required computing platforms with hardware configurations capable of performing tasks required for data and modeling tasks, as well as facilitating documentation and communication. The computing platforms utilized for the project development are consumer macOS/Windows local clients and Google Research Colaboratory (Colab). Data understanding and preparation tasks were performed on Apple and Windows Clients, which required an X86 64-bit CPU with a minimum of 8GB available system storage capacity for raw data, processed data, and software package for processing. Modeling tasks were performed on Colab and Jupyter, connected via MacOS/Windows clients.

In addition, TPU and GPU notebook configurations required 1GB of Google Drive storage for processed model input data and notebooks. Documentation and communications were conducted on the macOS/Windows clients, and storage of all project documentation required an additional 1GB of available Google Drive storage. Listing the hardware contents makes it easier for future

references as well. Tracking and identifying the required resources are made more accessible through this process. Table 4 shows the hardware requirements for the project.

Table 4

Hardware Requirements Table

Hardware	Configuration	Purpose
macOS Local Client (2x)	X86 64-bit CPU, minimum 8GB available system storage, internet connection	Data Collection, EDA, ETL, Connecting to Colab and Jupyter documentation, communication
Google Drive	2GB available storage	Model Input Data, Colab Notebook, Documentation Storage
Google Research Colaboratory	TPU, GPU	Model Development, Training, and Evaluation
Windows Local Client (2x)	X86 64-bit CPU, minimum 8GB available system storage, Internet connection	Data Collection, EDA, ETL, Connecting to Colab and Jupyter documentation, communication

Software Requirements

All macOS/Windows clients ran on the latest version of the respective OS with a browser capable of hosting Jupyter notebooks and connecting to Colab notebooks. The browser used were Chrome, Edge, and Safari. Models were written in Python 3.9 and used libraries from Scikit-learn and TensorFlow Keras for model development. Jupyter notebook, python, and the libraries are primarily used for coding. This also involves the various pre-processing operations required to remove the dataset from any inconsistencies. Data transformation, wrangling, and ETL Testing were performed using NumPy and Pandas. Visualizations for the exploratory analysis and model accuracy validation were created with Matplotlib, Scikit-learn, and Keras libraries. Table 5 lists the minimum software requirements and the versioning information for easier accessibility.

Table 5

Software Requirements Table

Software	Version	Purpose
macOS	macOS 10.15 (Catalina) or higher	Client OS
Chrome Browser	106 or higher	Colab/Jupyter Notebooks, Zoom
Matplotlib	3.6	Visualization
NumPy	1.23	EDA, ETL
Pandas	1.5	EDA, ETL
Python	3.9	EDA, Model Development
Safari Browser	15.6 or higher	Colab/Jupyter Notebooks, Zoom
Microsoft Edge Browser	106	Colab/Jupyter Notebooks, Zoom
Scikit-learn	1.1	ML Model Implementations, Visualization
TensorFlow Keras	2.1	DL Model Implementations, Visualization
Windows OS	Windows 11	Client OS

Tools and Licenses

Specific tools with their respective license will be used in conducting the project. Anaconda will be used for installing Python and Jupyter Notebook, all being open-source licenses. Jupyter Notebook will be used on local clients for data understanding and preparation. Google Colab, Jupyter Notebook, and Drive are free services used for model development. Google Drive was also used for storing project documentation. Project task management and visualizations were facilitated with free services from Jira, Diagrams.net, Lucid Charts, and DPMTool. Communications were conducted directly through the free service via video conference with Zoom, and Project documentation was captured with Office365 applications, which are student licensed. Table 6 lists these requirements.

Table 6

Tools and Licenses Table

Tool	License	Purpose
Anaconda	Open Source	Python 3.9, Jupyter Notebook Installation
DMPTool	Free	Data Management Plan Creation
Draw.io	Free	Pert Chart Creation
Google Drive	Free	Data, Notebook, Documentation Storage
Google Research Colaboratory	Free	Model Development Environment
Jira	Free	Project Management, Gantt Chart Creation
Jupyter Notebook	Open Source	EDA, ETL Environment
Lucid Charts	Free	WBS Creation
Office365	Student	Documentation, Presentation Creation
WhatsApp	Free	Direct Communications
Zoom	Student	Meetings

Project Costs and Justifications

The project had direct costs based on minimum requirements. Most of the resources were free of charge as free services/open-source software or had already been paid for prior to project commission. Student licenses cover some paid software, such as Office365 and Zoom. Table 7 depicts the cost breakdown of the hardware and software requirements. Moreover, the purpose of using this hardware or software is listed under the justification column. Free* in table 7 denotes a paid-for cost. The project cost section also includes the amount spent on the various system requirements. This can be attributed to the fact that it will be helpful when referred to in the future. The total cost of the project summed up to $907.

Table 7

Project Costs and Justifications Table

Resource	Duration	Cost
8GB RAM Cost	4 Months	$27
64-Bit Intel i7 processor	4 Months	$470
500 GB Hard Disk	4 Months	$50
Adobe	4 Months	Free*
Anaconda	3 Months	Free
Broadband Connection	4 Months	$80
Browsers (Chrome, Safari, Edge)	3 Months	Free
DMPTool	2 Days	Free
Diagrams.net	3 Days	Free
Google Drive	3 Months	Free
Google Research Colaboratory	1 Month	Free
Jira	3 Months	Free
Jupyter Notebook	3 Months	Free
Lucid Charts	5 Days	Free
Modem	4 Months	$150
Python, Libraries (Matplotlib, Numpy,	3 Months	Free
Office365	4 Months	Free*
TensorFlow Keras	1 Month	Free
Windows OS (10)	4 Months	$130
Zoom	4 Months	Free*

Project Organization Plan

An effective project organization plan is unquestionably necessary for carrying out a project from beginning to end, and it aids in identifying the tasks associated with the project's lifecycle. The understanding of the associated tasks aids significantly during the project schedule phase. The section that follows describes the project organization plan that was developed and implemented for this project based on the CRISP-DM methodology.

Work Breakdown Structure (WBS)

A Work Breakdown Structure (WBS) represents the complex project plan in a more straightforward format by breaking project steps into various components and work packages to get things done effectively. It also ensures that project plans are harmonious by integrating scope, cost, and schedule baselines. The WBS comes in handy for drafting the overall path that the project must take to ensure timely completion and will also help recognize the hurdles that could occur along the way and create a contingency plan beforehand. The WBS was created based on the six phases of CRISP-DM in Jira. The WBS also contained the higher-level breakdown of the six phases, which are discussed in the following sections. Servello and Evans (2002) summarized WBS as the set of tasks essential to accomplish and the milestones that must be met on a project.

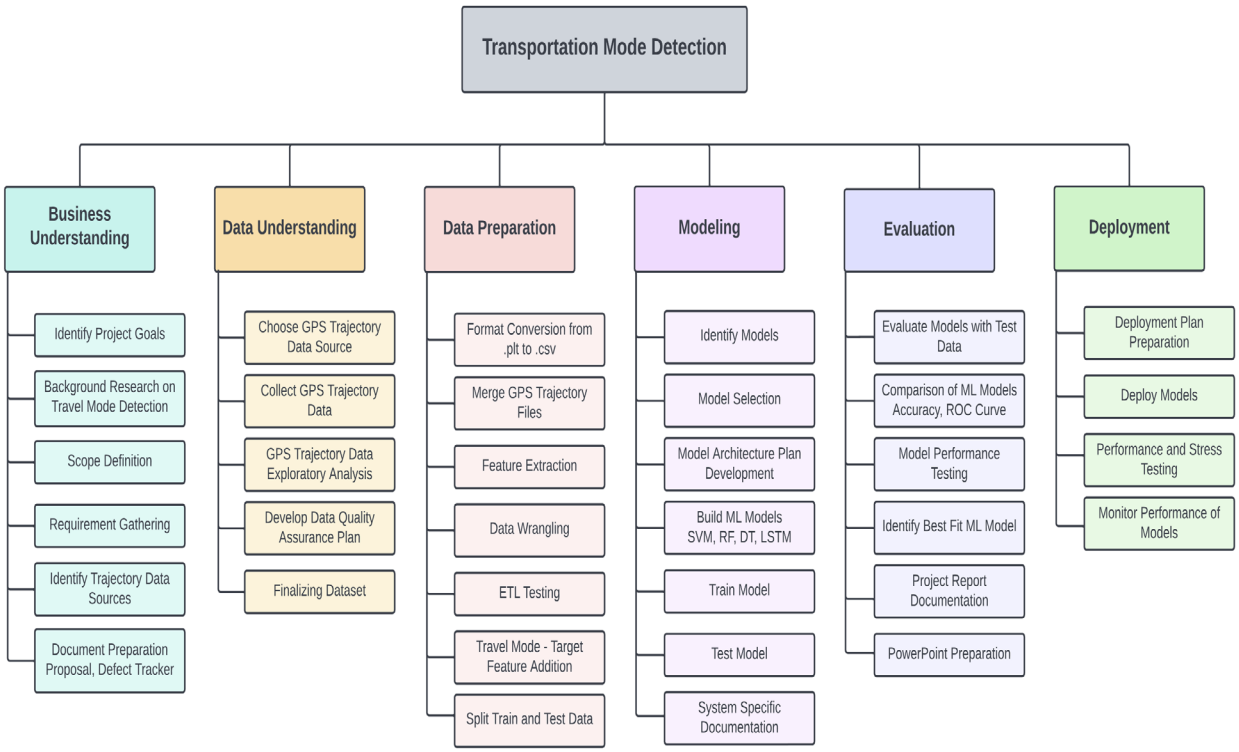
CRISP-DM is a popular data mining and analysis methodology. Some of the significant characteristics of this process are encouraging the focus on business goals, flexibility, and an iterative approach, not dependent on any specific technology. It has six primary phases: business understanding, data understanding and preparation, modeling, evaluation, and deployment. The project was set up in a project management tool called Jira, which helped in planning, tracking, and managing the project workflow.

Figure 23 represents the WBS created using a Lucid Chart. The top node depicts the project's name, 'Transportation Mode Detection.' It has six phases as per its association with CRISP-DM. The phases are Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, and Deployment. Each phase discussed above has its own stories and tasks tagged under it with various dependencies. The following sections will explain each of the phases mentioned above. Jira software does not provide any WBS functionality, so a marketplace plugin

called 'Aneto WBS' was installed.

Figure 23

Work Breakdown Structure



WBS Phase 1 – Business Understanding. Business understanding is the first Phase of the CRISP-DM methodology. In this phase, the focus was on understanding the objective and requirements of the project from the business perspective. The work breakdown of this Phase was created using Jira with the epic title "Project Initiation - Business Understanding." This Phase involved various tasks that helped attain insights regarding Travel Mode Detection. Some of the activities associated with this Phase are discussed here briefly. It started with identifying the project goals and then steered towards the background research of existing related topics. Knowledge attained from the Background research helped define the items in and out of scope

for this study. Then the Phase further comprised the tasks such as requirement gathering and identification of suitable data sources. The insights attained were useful in the upcoming phases.

WBS Phase 2 – Data Understanding. Data Understanding is the Phase that comes after the initial Phase of Business Understanding. In Jira, the WBS Phase was named "Data Understanding." The tasks associated with it were as the name suggests. This Phase consisted of tasks that helped decide the data source and familiarize with it, which implied that the focal point was on the data collection and analysis. The Phase began with choosing the data source after validating the compliance policies and relevance to the topic. Following the selection of data sources, the next step was to collect the data. Data collection is nothing but downloading the dataset in the necessary format.

After the collection, it is very much necessary to understand the data. Exploratory data analysis was the next activity that helped gain insights into the data more granularly. Data understanding also aided in deciding on various factors, such as the quality assurance plan to be developed, which ensures that the data passed through the models in the future is clean. The last task in this Phase was to finalize the dataset, which involved choosing the data source that passed through exploratory analysis and a quality assurance plan. The dataset chosen here was determined to be credible enough to be passed through the models.

WBS Phase 3 – Data Preparation. Following the "Data Understanding," the next phase of WBS was preparing the data. The phase was named in Jira "Data Preparation" and consisted of the tasks associated with bringing the raw data obtained in the previous step into a cleaner, condensed form. The raw data obtained in the previous phase were multiple files in vector-based plotter file (.plt) format. The first step of this phase was to convert the .plt files into the comma-separated file (.csv) format, then merge the multiple files into one. Next came the data wrangling

steps, which helped eliminate the corrupted data and redundancy. The original features present in the .csv file were extracted. This step was followed by extracting additional features such as distance traveled, the time taken for travel, speed, and other features that will aid in mode identification which are nothing but calculated fields from the original features. ETL Testing was also performed in this phase to validate the condensed data, followed by the steps to ensure that the condensed file has the target feature added. The processed dataset was split into training and testing sets and passed to the next phase.

WBS Phase 4 – Modeling. The training and testing datasets generated from the previous phase were passed on to this next phase, named in JIRA as "Modeling. "The Modeling phase was where the intensive course of action began, and modeling was where various machine learning models were designed and built.

Firstly, a considerable amount of time was spent identifying the potential models, followed by Model selection. Once the models were selected, the next step was to develop an architectural plan on how the selected models will be brought to working conditions. Following the architectural design step, the actual model building began. The models chosen were built in this step, followed by training the model with the training data from the previous phase.

The four models developed here used a Decision Tree (DT), Random Forest (RF), Support Vector Machine (SVM), and Long-Short Term Memory (LSTM). The models were then tested using the testing data, and the results in this step were passed on to the next phase for evaluation. This phase also included the process of documenting the changes made and tracking the issues and fixes as well.

WBS Phase 5 – Evaluation. The current phase was named "Evaluation" in JIRA, and the results from the modeling phase were validated in this phase. The evaluation phase focused on

the technical model assessment, and the models were evaluated and compared to determine if the selected model covers all essential business issues. The results produced after validating the models with the test data in the previous phase were evaluated in this step. Some validation tasks included cross-validation of accuracies attained by the results of multiple models followed by the confusion matrix analysis and the ROC Curve analysis. Stress testing was also performed, and each model's results were evaluated under specific conditions during the stress test. The outcome of the tests and analysis helped identify the best fit model for Travel Mode Detection. Proper documentation of the design and evaluation done so far were maintained, which will help for future reference. Any issues in the stress testing phase were fixed and tracked to be easier to reference when a similar issue occurs in future cases. This phase also included a PowerPoint presentation to showcase the model and results. The best fit model was then passed on to the next phase, which was deployment.

WBS Phase 6 – Deployment. The best fit model chosen from the previous evaluation phase was passed on to the current phase, which involved deployment in the production environment. The WBS phase was named "Deployment" in JIRA and contained the tasks associated with the deployment, testing, and monitoring in the real-time environment. The first step was to prepare a plan for deployment, and the planning helped devise the procedure to be followed while carrying out the deployment process. The next step was to carry out the devised plan and implement the model in a real-time environment. Performance testing was also performed in this phase to detect if any mishaps occur due to the intense data load in the production environment. The model's performance over a set period was monitored, and enhancements and fixes were carried out during the maintenance phase. Maintenance phase helps the model keep its competency over a long tenure.

WBS was a big help during the presentation of a high-level overview of the project plan to the stakeholders and team. As WBS represented the initial planning stage, it might not be perfect due to its volatility. The phases discussed here will be drilled further at a granular level during the project schedule phase using PERT and Gantt charts. WBS acted as the foundation, listing the associated components and work packages. This WBS was enhanced further using PERT Chart highlighting the critical path, which is discussed next in the section. Moreover, an even more in-depth breakdown of the Gantt chart is discussed in the upcoming sections. This breakdown by PERT and Gantt charts assisted in comprehending the tasks in more detail.

Project Schedule

WBS provided a high-level overview of the tasks associated with the six CRISP-DM phases. On the other hand, a project schedule goes into greater detail about the tasks, timelines, dependencies, and assigned resources associated with each task. It aids in tracking project progress and ensuring that it is on track for success.

PERT Chart

A Project (or Program) Evaluation and Review Technique (PERT) chart is a visual project management tool. The chart is mainly utilized for creating an initial schedule to estimate the number of days required for project completion, which can then be shared with the stakeholders for further discussion. A PERT chart can aid in achieving various planning activities, such as getting the timeline approved, estimating the time required for individual tasks, identifying project objectives, and showcasing the overall flow of the project before even beginning.

This chart illustrates all the tasks and the time required to complete the project. Additionally, it is used to organize, schedule, and coordinate the tasks involved. A PERT chart

uses boxes that depict the tasks and arrows representing the dependencies between the tasks and the criticality based on the color. The tasks linked with a red arrow represent the critical path which can be explained as the shortest path to completing the project and ensuring timely completion. Any delay in the tasks associated with the critical path might have a ripple effect and influence the entire project timeline.

Figure 24 below depicts a PERT chart for the project 'Transportation Mode detection.' Each color-coded lane in figure 24 represents the six phases of CRISP-DM: Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, and Deployment. The boxes inside each lane represent a task and consist of information regarding the task, such as task name, estimated start date, estimated end date, duration, and the assignee of each task. The phases and the enclosed tasks will be explained in detail in the following section.

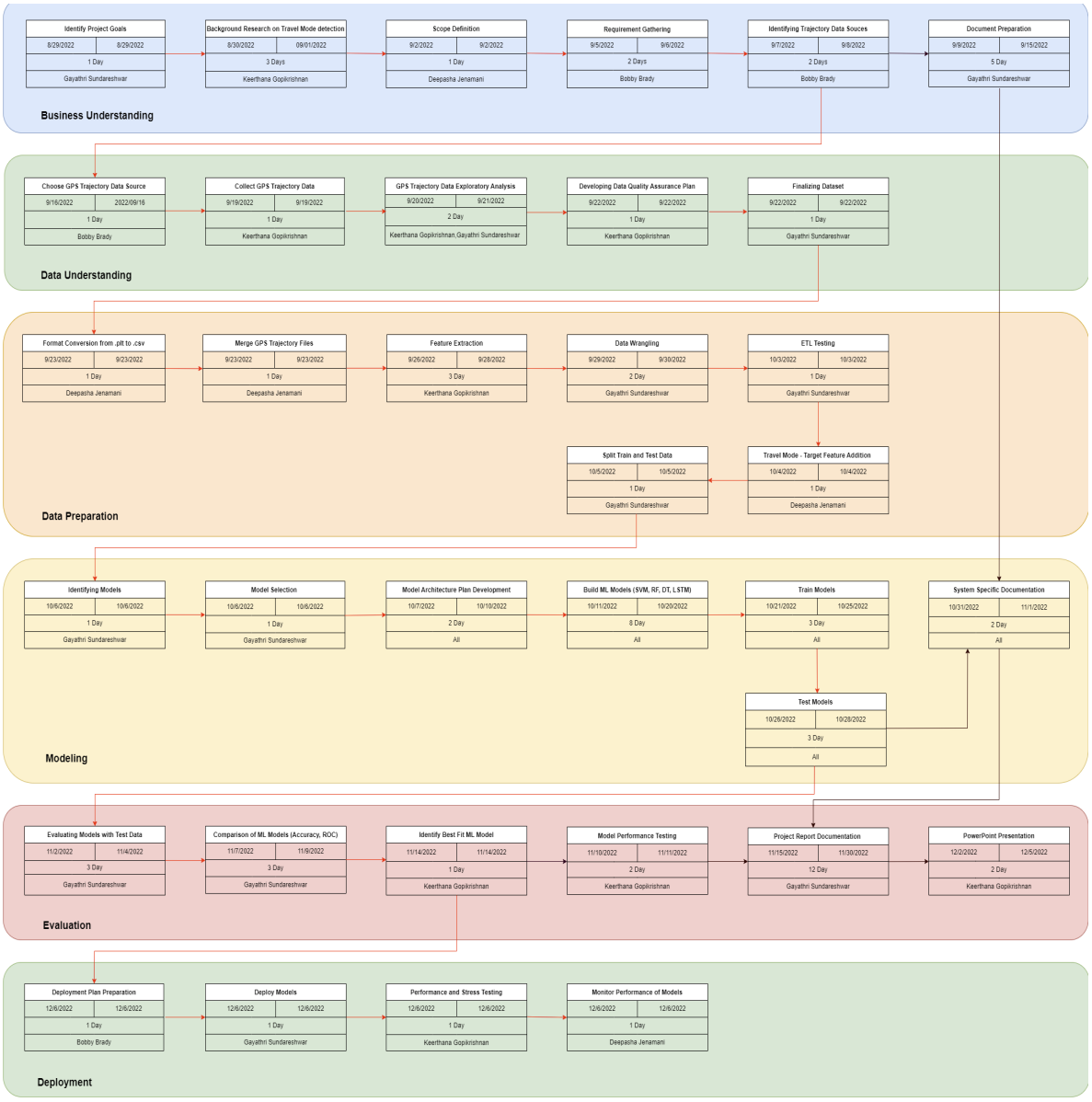
Various tools assist in the creation of the PERT Chart. The tool used for this project was diagrams.net which had specific predefined templates that aided in speeding up the process of creation.

PERT Phase 1 – Business Understanding. Business understanding is the first phase of the CRISP-DM methodology. The PERT chart represents a visual depiction of each phase marked as a lane color-coded distinctly. The tasks associated with this lane consist of identifying the project goals, background research on existing works, scope definition, requirement gathering, identifying data sources, and project proposal documentation. The epics created were similar to the plan created using WBS. However, the distinctive feature was that the tasks were more detailed with the estimated start and finish dates and duration and contain the name of the person to whom the task was tagged. The phase began with understanding the project's goals, after which an intensive search for the existing related works is done. Insights attained during the

phase helped decide on the requirements to be gathered. The task following this was identifying potential data sources suitable to the topic. Documentation was done to keep track of the progress made so far. The document was expected to be prepared to be a helpful reference for related future studies.

Figure 24

PERT Chart



When observed closely, it was easy to notice that most of the tasks listed above depended on the previous one, and delay in any of the tasks influenced all the following tasks. Project documentation was the only task that did not induce a delay in this phase. Hence all the tasks except that were marked as a part of the critical path. The estimated days for completion of this phase was 14 days which was found by cumulating the duration of each task.

PERT Phase 2 – Data Understanding. Similar to WBS, the phase following business understanding was Data Collection. The phase consisted of tasks that help get familiarized with the data. The main focus of this step was on data collection and analysis. It began with choosing the best fit data source from the potential ones identified in the business understanding phase. The task involved identifying a credible data source after verifying the compliance and privacy policy. The chosen dataset was collected in the required format and stored at a specific place similar to the WBS explanation of the same phase.

Understanding the data also helped decide on the quality assurance plan to be developed, which ensures that the data passed through the models in the future is clean. Exploratory analysis was the next task performed to understand data after extracting the data, which aided in checking the distribution and size, the correlation among different features, and other insights to help identify any patterns or trends in the data. The next task developing a data quality assurance plan, was vital, and it consisted of the validations to check if the chosen dataset was suitable for the model's development. The last task in this phase was to finalize the dataset, which involved choosing the data source that passed through exploratory analysis and a quality assurance plan so the dataset chosen here was credible enough to be passed through the models. The 'Data Understanding' phase lane in the PERT Chart showed that the overall duration assigned for this phase was five days. Each task was linked using a red arrow representing the dependency.

PERT Phase 3 – Data Preparation. The Third phase in the PERT chart was called 'Data Preparation.' The phase consisted of seven tasks which are format conversion, merging, original and target feature extraction, cleansing, ETL testing, and concluded with the creation of the test and train datasets. Format Conversion from .plt to .csv was the first task in this phase, which involved converting the GPS trajectory details in the .plt file and the labels in .txt files into a .csv file, which was present for the users with their timestamps. The second was merging the datasets from multiple .csv files obtained for each user with labels and GPS Trajectory into a single .csv file. Feature extraction, which involved extracting additional features such as distance traveled, the time taken for travel, speed, and other features that will aid in mode identification, was also extracted in this task.

The following task was data wrangling/cleaning, which involved eliminating redundant records that could skew the predictions and interfere with the visualizations. Handling nulls and redundancies were some activities that could be carried out as part of this process.

The tasks in the 'Data Preparation' phase resulted in the generation of processed data from the raw data extracted in the previous phase. ETL Testing was performed to ensure the credibility of the dataset. The dataset generated from the wrangling process was passed through a fixed list of tests such as nulls check, duplicate check, special character analysis, and other required custom tests. The target feature addition task included adding the target feature to the finalized dataset, which was used to evaluate the models. The final task in this phase was to separate the training, and test datasets, where the data prepared for processing by the models is divided into training and testing sets, which were used to train and test the model. The final dataset was obtained from the raw data in this phase. The PERT chart also showed that all the tasks in this phase were dependent on the previous task and hence were linked using red lines representing that the linked

tasks belong to the critical path. The chart also depicted that the total number of days allocated for this task was nine days, which can be confirmed by cumulating the duration allotted for each of the tasks.

PERT Phase 4 – Modeling. Various appropriate models were built and evaluated using the extracted features in the modeling phase. The selected models were developed with different settings, parameters, and features to detect and predict travel modes. The first task was identifying potential models; this involved identifying potential algorithms, models that appear suitable for the topic, and additional parameters for solving the problem based on the collected features. The best-fit models that will be built were identified during the model selection.

Developing an architecture plan for chosen models involved identifying the steps that should be conducted for the successful execution of the model and keeping track of the roadblocks that could occur along the way. The following task was to build selected ML models, which consisted of generating the required codes for the selected models. The chosen models were trained, which involved running the model using the training data. The model was then tested in the next task, which involved testing the execution of the models with a sample data set after training, which helped identify and resolve the issues. The last task in this phase was system-specific documentation that included documents regarding the defects in the model, the bugs, and its solution. This task also had records about the changes implemented in each model.

In the PERT chart, all the tasks except the system Specific documentation were linked using a red line representing the critical path. System Specific Documentation task did not affect the start date of any of the following tasks, and any delay with it did not have much ripple effect; hence it was not part of the critical path. The overall time allocated for this phase was 19 days which was cumulative of all the tasks that were part of this phase.

PERT Phase 5 – Evaluation. The evaluation phase focused on the technical model assessment, which involved evaluating the individual models, comparing the accuracy attained by the models chosen, and evaluating the performance and stress testing results. As a result, the best fit model was identified, which was then passed on to deployment. The phase also included the preparation of project report documentation and PowerPoint preparation.

The task named comparison of the ML Models involved comparing the accuracy attained by the selected models and analyzing the confusion matrix and ROC curve. The following task involved model performance testing, which helped identify the models' performance when put through an intense workload and thus confirms whether the chosen model will be efficient enough in case of more extensive data loads.

Identifying the best fit ML Model was the next task which identified the best-suited model. Additionally, it involved identifying future enhancements that could be performed on the selected model. The next task was developing project report documentation which included documenting the problem definition, literature review, technological survey, dataset selection and preparation, architecture review, model evaluation, result comparison, limitations, and future enhancements. The report worked as reference material when used in other future studies. The last task in this phase was PowerPoint preparation, where a PowerPoint was created to act as a reference while presenting the project and its solutions to the stakeholders/users. In the PERT chart, dependencies were established between all the tasks.

The project document preparation and PowerPoint presentation were deemed not critical since neither influenced the initiation of the following tasks; hence all the tasks except the mentioned two were marked to be on the critical path. The overall time allotted for this phase was 23 days, which was the cumulation of the duration of individual tasks.

PERT Phase 6 – Deployment. Deployment was the last phase of this project, where the model was tested in real-life working conditions. Additionally, it checked whether all the business requirements were met and worked as expected. The first task was deployment plan preparation, which involved devising a plan to evaluate the performance of the models in real-life scenarios where roadblocks were inevitable, and the pressure on the model was immense due to the large dataset volume. This phase's second task was deploying the model in the production environment based on the deployment plan, and successful execution means the model was credible enough to be launched.

Performance and stress testing this task included executing the model in a production environment and putting it through stress tests to evaluate how it held its place when dealing with a massive amount of data. A stress test also helped identify memory usage and efficiency under such conditions. The last task of this phase was to monitor the performance of models, which involved timely tracking of the models to check for potential failures or issues during the initial phases of deployment. In the PERT chart, all tasks in this phase were linked with red lines, representing the critical path due to its dependency factor. The overall time allocated for this phase was 14 days which was the cumulation of all the individual task's durations.

Pert chart aided in understanding the overall flow of the steps that need to be taken to ensure the project's timely completion. The critical path helps comprehend which tasks in phases needed to be completed without delay so that the following phases and task deadlines were not affected.

Gantt Chart

A Gantt chart visualizes project tasks planned out over a period using a bar chart. The graph displays the beginning and ending dates of epics, stories, and subtasks. It helps manage,

monitor, and schedule the resources and tasks in a project. The charts' bars have different lengths, representing the timeline of how long each step lasts. Visual representation helps us identify tasks that cannot begin until the previous tasks are not finished and the ones that are carried out concurrently. The stories created will have dependencies to help establish the critical and the noncritical paths.

Bednjanec and Tretinjak (2013) said that the Gantt chart's advantage is a graphical outline, and each block of time has a title that identifies the task and its duration. Each distinct action is represented as a block of time in Gantt charts. The graphical overview is a crucial benefit. Using Gantt charts, one can see how long each task should take and the sequence in which it should be completed. Each plan must be specific, with significant objectives divided into smaller, more manageable subtasks to make it simpler to track its progress (p. 1).

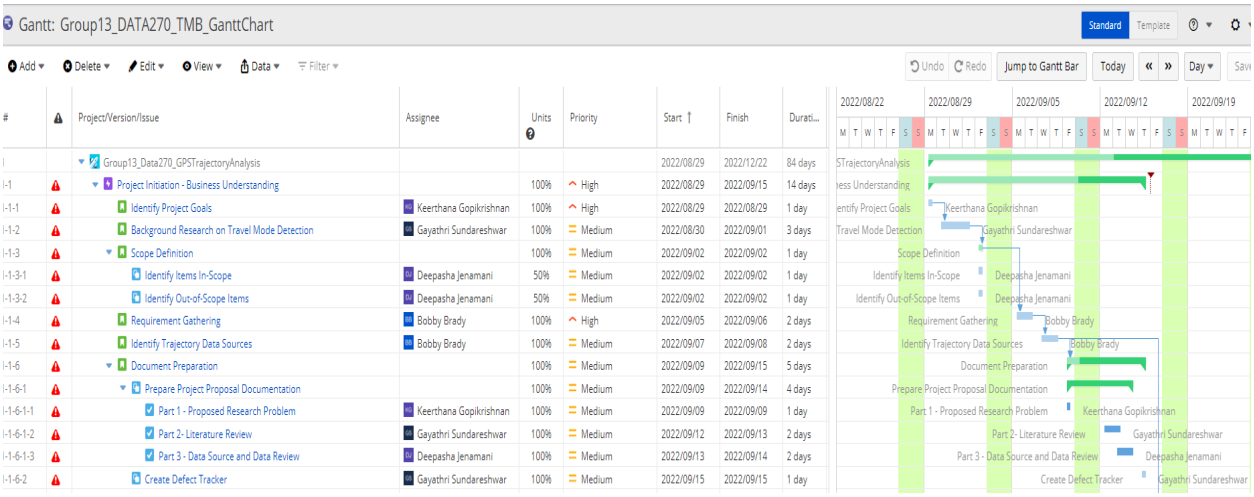
The previous section, which covered the WBS and PERT chart, gave an overview of what comes in each phase of CRISP-DM. This section pays attention to the phases at a granular level which included assigning various measures related to the task, such as the person in charge, the priority of each task, the estimated start and finish date, and allocated units. The Gantt chart also aided in tracking the progress of the task over time, starting from the initial status 'to do' and ending with the status 'done.'

The units allocated to the task were directly proportional to the estimated effort required for its completion. For instance, the units allocated were 100% if it required an entire day. Each task unit was assigned based on the team member's workload on a given day. The units were allocated respectful of the key constraint, which ensures that no person was assigned more than 100% effort estimation for a single working day. The granular level breakdown of each of the CRISP-DM phases will be discussed briefly.

Business Understanding – Granular Level Breakdown. The tasks associated with this phase were identifying project goals, background research on existing works, scope definition, requirement gathering, identifying data sources, and initial document preparation. Some of the tasks were broken down further from how they were in the PERT chart, and additional subtasks were added. Some subtasks added were scope definition, broken down into identifying items in scope and out-of-scope for this project. Figure 25 represents the Gantt chart breakdown of the business understanding phase.

Figure 25

Gantt Chart - Business Understanding Phase



Most of the tasks in business understanding had the priority set to the medium level, and the tasks were split among the team members evenly; the first phase was part of sprint one; this sprint lasted for 14 days and encapsulated the overall completion of phase one. The tasks in this phase must be completed before the initiation of the next phase due to dependency. As mentioned in PERT, the documentation task does not delay any other upcoming tasks; hence it was not part of the critical path while all others are. The individual tasks in the phase, such as the background research, were done in three days; the project goals and scope definitions were

carried out for one day; and finally, the project requirement identification was allotted in two days. All the weekends and holidays, such as Labor Day, Thanksgiving, Christmas, and Easter, were not considered working days, and tasks were assigned based on this. The units allocated for the tasks were mostly 100%. Some subtasks were allocated to the same person and started on the same day. In such cases, the allocation was split between the task; for instance, in the scope definition, the task has two subtasks that started on the same day and were assigned to the same person; hence the effort was split equally between the subtasks.

The stories created in the Gantt chart were linked to show the task dependencies. A proper flow was established so that the following data understanding phase cannot start until the business understanding phase's tasks were complete.

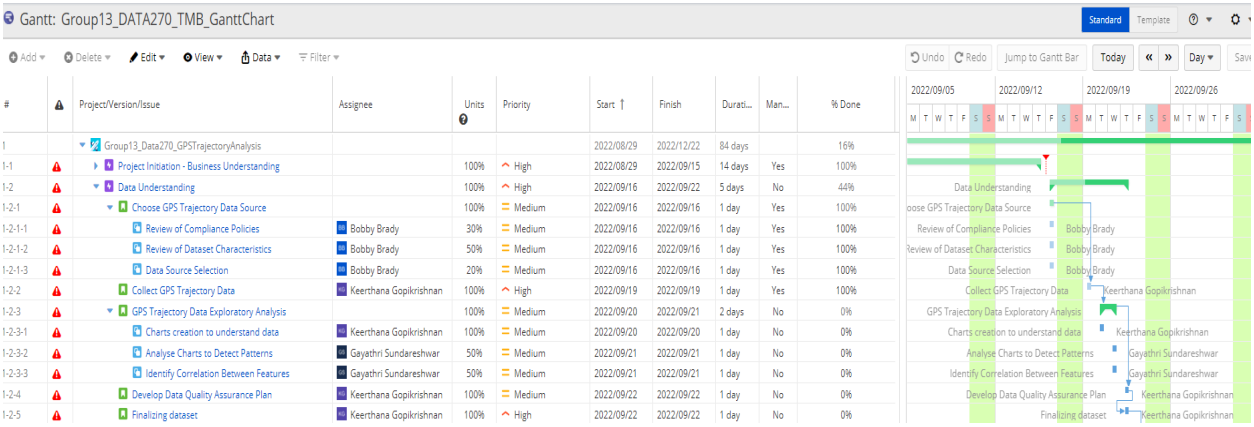
Data Understanding – Granular Level Breakdown. The data understanding phase began after the business understanding phase was completed, and this section goes over the tasks associated with this phase in greater detail. This phase consisted of five tasks: choosing the data source, data collection, exploratory analysis to understand data, developing a data quality assurance plan, and finalizing the dataset. Some of the tasks mentioned above were divided further into smaller chunks, and various related subtasks were created. Figure 26 represents the Gantt chart breakdown of the Data Understanding Phase.

The first task in this phase consisted of three subtasks: a review of compliance policies, dataset characteristics, and data source selection. Similarly, the exploratory analysis task was split into three: chart creation to understand data, analysis charts to detect the pattern, and identifying the correlation between features. The units representing the estimated efforts were influenced by the start date and assignee. Tasks beginning on the same date and assigned to the same person were allocated with units that would cumulatively add up to 100%. For example,

the efforts assigned for the task 'choose data source' were divided into 30%,50%, and 20%, since all three were allocated to the same person and began on the same day.

Figure 26

Gantt Chart - Data Understanding Phase



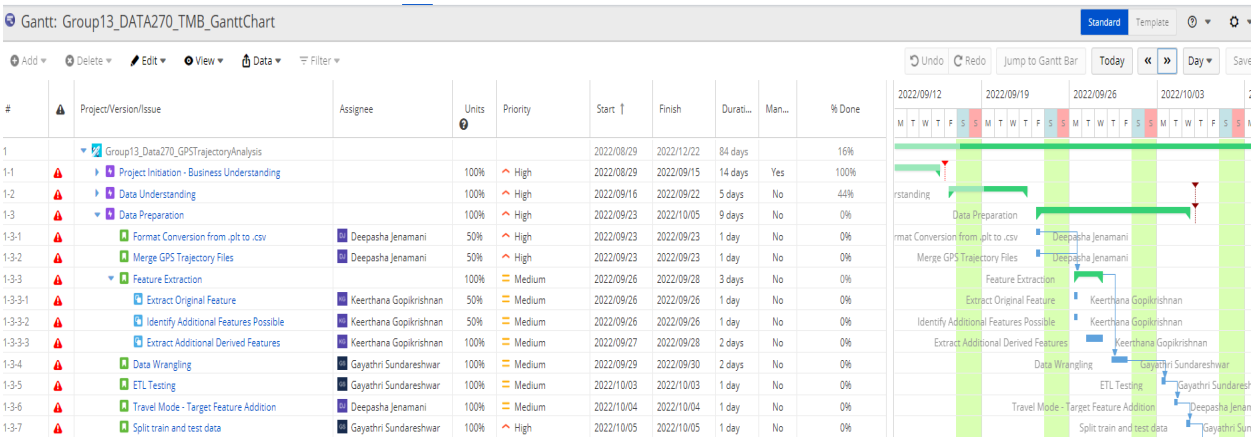
Most of the tasks in this section had a medium priority, except for data collection and finalizing the dataset, which was of high priority. This phase was the first part of the second sprint, estimated to last five days. The estimated duration for each task was not more than one day except for the exploratory analysis, which estimates two days. All the weekends and holidays, such as Labor Day, Thanksgiving, Christmas, and Easter, were not considered working days, and tasks were assigned based on this. A task cannot be marked done until all the associated subtasks were completed. The data preparation phase's initiation depended on the data understanding phase's completion; therefore, a dependency was established between these phases to ensure the requirement is met.

Data Preparation – Granular Level Breakdown. The tasks linked with the data preparation phase were format conversion from a .plt file to a .csv file, merging the multiple files containing the dataset into one, feature extraction, data wrangling/ cleansing, ETL testing, target feature addition, and splitting the dataset for training and testing.

Figure 27 depicts the Gantt chart breakdown of the Data Preparation Phase. In addition to the tasks already present in the PERT chart, some tasks were further decomposed. The feature extraction tasks were broken down into three subtasks: extracting the original features, identifying the additional features, and extracting the derived feature. In order to complete this task, the subtasks associated with it had to be completed. The tasks in this phase were performed in the same order as previously mentioned, and they cannot start until their previous task was completed.

Figure 27

Gantt Chart - Data Preparation Phase



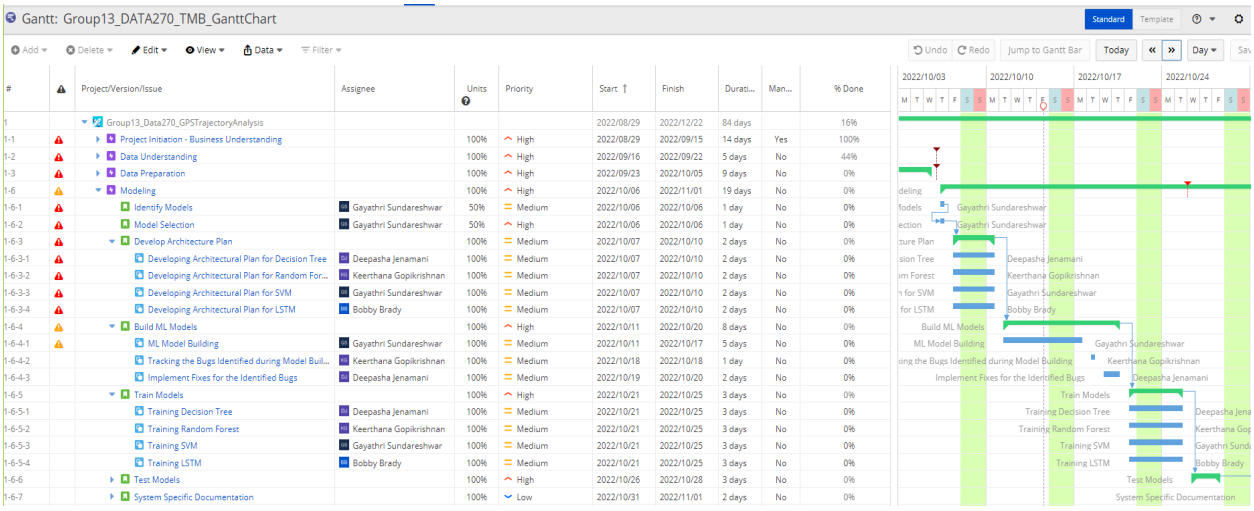
The tasks set to high priority in this phase were formatting and merging the data after acquiring ad splitting the train and test set. The epic fell in sprint two, and nine days were allotted for this phase. All the weekends and holidays, such as Labor Day, Thanksgiving, Christmas, and Easter, were not considered working days, and tasks were assigned based on this. Cleaning the dataset was done for two days, and feature extraction was performed for three days. The units were allocated to the tasks by the assignee's daily workload, which added up to 100%. For example, there were three subtasks in feature extraction; the first two started on the same day, the units to measure efforts were split equally with 50% for each, and the third one began

the next day; hence the units allotted for this subtask will be 100%. The overall units for this task were allotted 100%. This epic for data understanding needed to be completed to continue the project's next phase.

Modeling – Granular Level Breakdown. After the data preparation phase was completed, the modeling phase began. This section dived into greater detail about the tasks related to the modeling phase. Seven tasks were part of this phase: identify potential models, initial model selection, develop an architecture plan, build selected ML models, train chosen models, test chosen models, and system-specific documentation. Figure 28 depicts the Gantt chart breakdown of the Modeling Phase.

Figure 28

Gantt Chart - Modeling Phase



Like the data preparation phase, some of the tasks mentioned above in this phase were further divided into smaller segments. For instance, developing an architecture plan for each identified model was necessary; thus, a subtask for each under the main task was created. Tasks began on the same date and were assigned to the same person, and the units allocated were divided so that the total allocated units to the same assigned person would be 100%. For

instance, the efforts set for the task 'identify potential model' and the 'initial model selection' were equally split into 50% and 50% since both tasks were allocated to the same person and started on the same date.

Specific tasks in this phase were marked as higher priority based on their criticality, such as initial model selection, building, training, and testing the model. This phase was estimated to get completed in 19 days. All the weekends and holidays, such as Labor Day, Thanksgiving, Christmas, and easter, were not considered working days, and tasks were assigned based on this. The model fell under the third sprint from the first task until the training task, whereas testing the model belonged to the fourth sprint. The estimated duration for each task differed based on the task's importance and the subtasks involved. For example, the estimated time for building a model was eight days, whereas identifying potential models was one day. The completion of a task depended upon the completion of the associated subtasks. Furthermore, the evaluation phase cannot initiate until the modeling phase was over; therefore, a dependency was specified between these two phases.

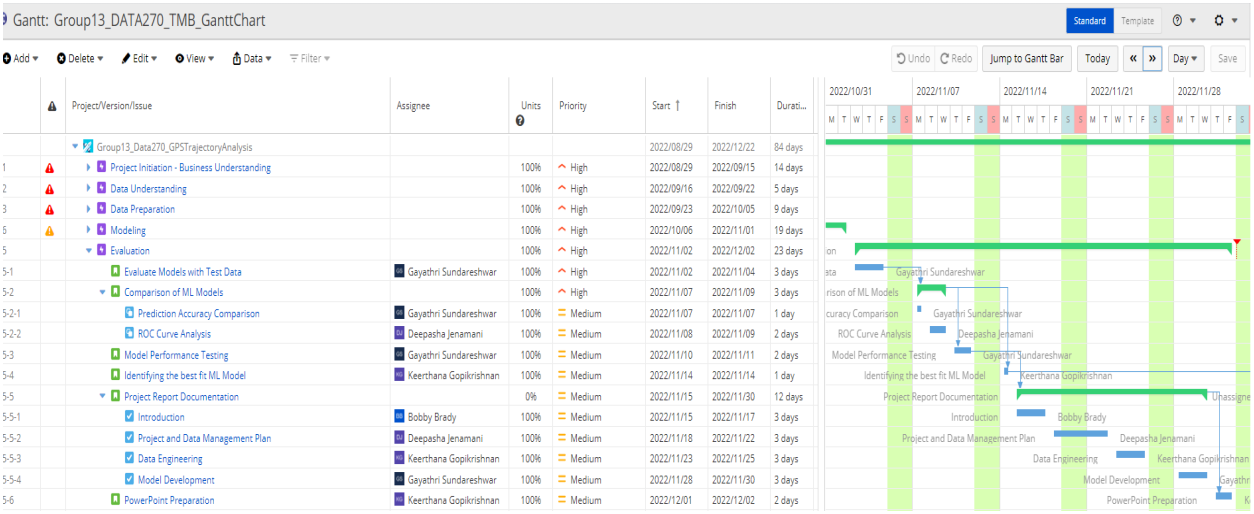
Evaluation – Granular Level Breakdown. The evaluation phase comprised various tasks: evaluating the models with test data, comparing ML models, model performance testing, identifying the best fit ML model, developing project report documentation, and PowerPoint presentation. The tasks from the previously mentioned section were divided into subtasks. The model comparison task was split into two subtasks where the ML models were compared using a confusion matrix and ROC curve analysis; doing so ensured the task will yield a good result. Figure 29 depicts the Gantt chart breakdown of the Evaluation Phase.

The tasks with high priority were the first two, which helped evaluate if the built model was functioning correctly. Any issues encountered during this process were noted and dealt with

before moving on to the next epic. Since each assignee had just one task to complete on any given day, the units used to measure efforts in this phase were 100% distributed among all tasks.

Figure 29

Gantt Chart - Evaluation Phase



This phase was split across sprints four and five; in sprint four, the first four tasks were spread across nine days, and in sprint five, the remaining two tasks were completed. Sprint four consisted of the evaluation and comparison of the ML model, which was done for three days respectively, and identifying the best fit ML model and performance testing was done for one and three days, respectively. All the weekends and holidays, such as Labor Day, Thanksgiving, Christmas, and Easter, were not considered working days, and tasks were assigned based on this.

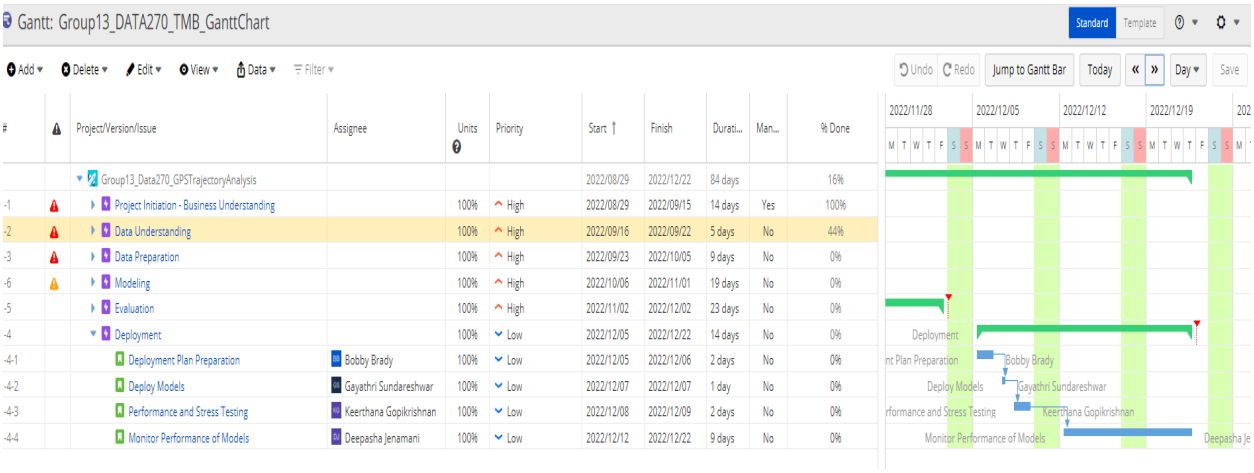
All the above-mentioned tasks depended on the completion of the previous task to begin the process and progress to completion. Sprint five contained two tasks that were not dependent on any other task and hence marked not critical; any delay with the documentation did not affect the next phase, which was the project deployment.

Deployment– Granular Level Breakdown. This phase encapsulated tasks such as deployment plan preparation, deploying the models, performance, and stress testing, and

monitoring the performance of the models. The task in this phase was marked as low priority as completing these tasks was not part of the course requirement, and delays in these tasks did not affect the project's overall completion**.** All the weekends and holidays, such as Labor Day, Thanksgiving, Christmas, and Easter, were not considered working days, and tasks were assigned based on this. Figure 30 depicts the Gantt chart breakdown of the Deployment Phase**.**

Figure 30

Gantt Chart - Deployment Phase



The units were assigned to the tasks to measure the efforts in this phase; all the tasks were allotted 100%. Since each assignee handled each task and did not begin on the same day. Deployment was covered in sprint six, which lasted for 14 days; monitoring the model's performance in a real-world setting was done for nine days. The final stage of the project was completed in this sprint, and any other future enhancements encountered during the model building were noted down for future works.

Data Engineering

Data Process

Data can be a significant factor that can be persuasive while making business decisions. An adequately sourced collection of information regarding the topic can come in handy when

identifying the patterns, leading to business decisions that can impact future performance. Hence, a proper process should be devised and maintained throughout the data collection and preparation. The volume of data collected is proportional to the efficiency of a solution. However, it was essential to know which kind of data to capture to gain insight, which can aid in sufficing the project requirements. Thus, accurate and appropriate data must be collected to maintain research integrity and assist in reaching valuable outcomes. Therefore, an effective collection plan must be devised and employed to achieve the goal. Hence, as the initial step, a coherent strategy was devised and executed for the data collection phase.

The problem statement in the previous section describes that this project's objective was to accurately detect the traveling modes from the recorded GPS trajectories. Thus, the dataset selected should have certain mandatory features, such as location details, time of travel, and modes used for traveling, to be considered qualifying. A data collection plan can aid in identifying these crucial features and other additional features. It can also help keep track of the characteristics of the data, such as the metrics, datatype, and instrumentation information. Thus, a data collection plan was vital. It involves the identification of viable data sources, the time frame for the data to be collected, and methods of collection such as instrumental or survey. If required, additional transformations were also performed to derive the desired features influencing the model's performance.

The collection process will result in a dataset containing all the mandatory and other optional features, but this data is often raw and unclean. The selected data may have missing values, incompatible data types, or nulls. These discrepancies could influence the model's prediction and lead to an incorrect conclusion. A cleaning process must be performed before it is used in subsequent steps to rectify such discrepancies. Some critical steps that help rectify are

handling missing or incomplete data, handling noisy data, which could be possible due to location factors and the same travel time, and handling inconsistent data that any redundant records might have caused.

Once the data collection was complete, it was crucial to visualize, analyze, and summarize the primary characteristics of the dataset. Initial Exploratory data analysis helps understand the data better, and also it helps discover the extremities in the dataset that were not identifiable through eyeball validation. Exploratory data analysis tasks involve creating visualizations and displaying statistical information, making it easier to discover the data distribution, patterns, correlation, and anomalies. The analytical tasks include a statistical display, univariate or multivariate analysis, and dimensionality reduction. All these analyses lead to a better understanding and gaining more valuable insights.

After the initial exploratory data analysis, the extremities and abnormal values in the dataset were identified and processed while cleaning. After the data cleaning step, the data was processed, and all the quality issues were handled appropriately. Although the data was processed and clean, it was necessary to identify if there could exist any quantifiable information that could be derived from the existing features, which will help enhance the model's prediction. Before proceeding with derived feature extraction, it was essential to look for ways to gain additional insight by performing some required transformations. This can help generate valuable derived features and help improve a model's prediction accuracy. Additionally, it was necessary to check if there was any mismatch in the data format, data type, and consistency between the features. In addition, it was also necessary to check if it needs any data standardization, normalization, or additional aggregation. Therefore, an apt transformation strategy was planned and implemented. As part of this, the validation steps mentioned here were conducted to resolve

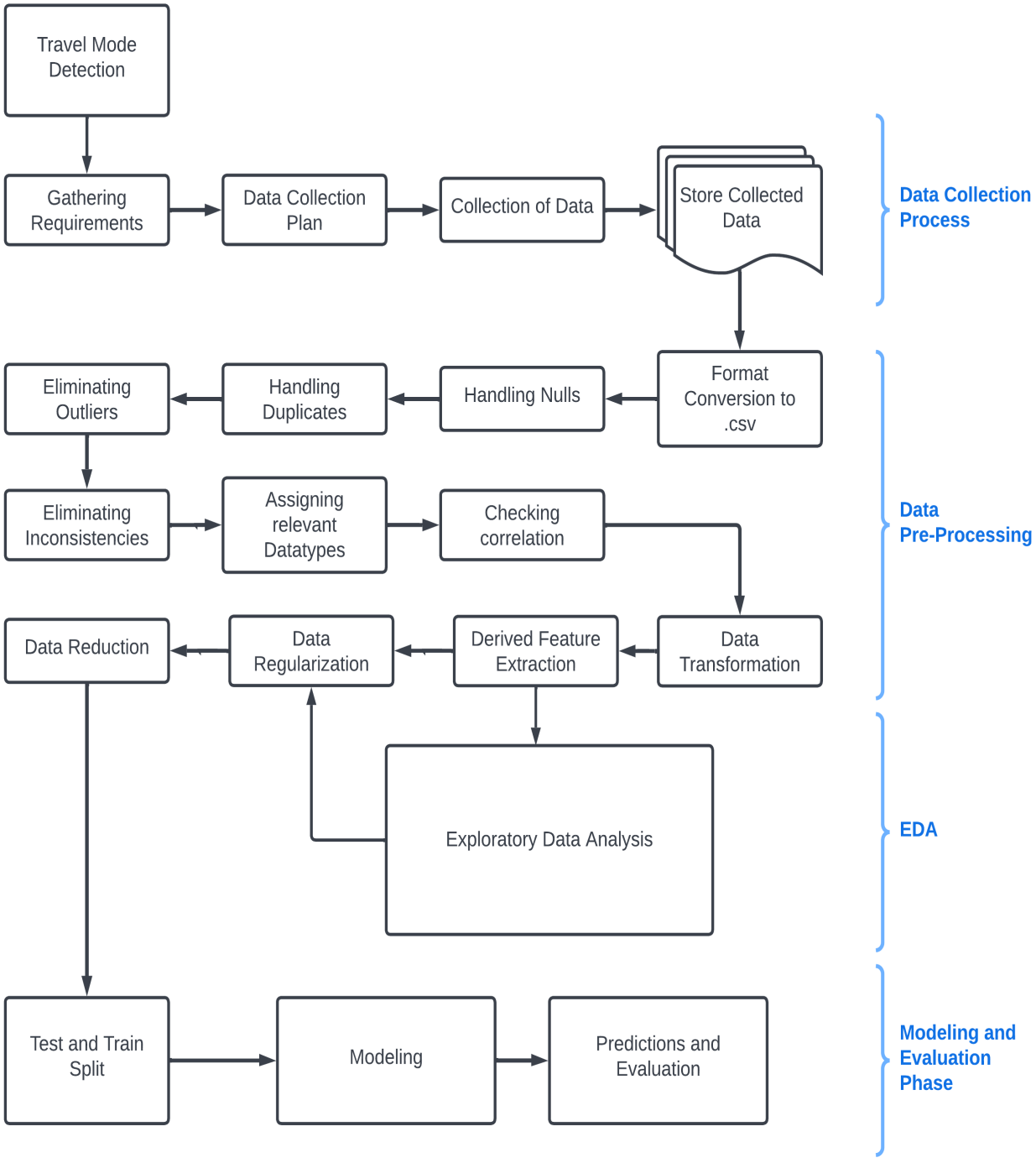
the issues in the processed data. Additional features were also derived, which could help in the modeling phase.

After completing the data collection, cleaning, transformation, and analysis, it was now qualified for further use in the subsequent modeling phase. Hence, an approach was devised to use the data effectively to evaluate the model's performance; thus, a technique known as the train-test split was used. The reason behind using this technique was that the dataset was sufficiently large, and each subset of the dataset will suitably represent the problem domain. As part of this process, the dataset was divided into two subtests. The first one was referred to as the training dataset, which was used to fit the model. The second one was used for testing, where the recorded predictions are compared to the expected values. It now raises the question of how much data should be considered for training and how much for testing. There is no optimal split percentage; instead, it depends on the project's goal with consideration of some factors such as computational cost and train and test data representativeness. Various libraries were present to aid in this process. In this project, scikit-learn, a Python machine-learning library, divides the dataset into training and testing sets. The split size was specified using a "test_size" argument; accordingly, 70% was allocated for training and the remaining 30% for testing. Another important consideration that needed to be made was getting a random set of records for the train and test sets. The test and train split were performed while ensuring that it represented the original dataset without any bias on either set. Figure 31 displays a workflow diagram. showcasing the various steps in data process phase such as data collection, pre-processing steps.

The sampling was done so that every model was fit and evaluated without bias and this was ensured using a "random_state.". Another important consideration for the classification task was ensuring that the sample is balanced based on the multi-level class labels present.

Figure 31

Workflow Diagram with Detailed Data Processing Steps



The sampling was done so that every model was fit and evaluated without bias and this was ensured using a "random_state." Another important consideration for the classification task

was ensuring that the sample is balanced, which means the balanced number of records for each target label. Thus, an argument known as "stratify" was used on the target label, which reserves the same proportion of records for each target label as in the original dataset.One of the chosen deep learning models uses dropout as a data regularization technique in the modeling phase. The reason for choosing this was because dropout works better with the Long-Short term memory model. It works by eliminating the unnecessary nodes, thus eliminating the complexity of the model without sacrificing the quality of the result. It also has efficiently reduced the system's overhead so that the model can run smoothly in the following phase.

Data Collection

Data Collection Requirements

Establishing the requirements to be observed while collecting data should be emphasized since it plays a critical role in identifying the relevant features without compromising the quality. The initial step would be to decide on the required type and volume of the data, which could be identified by understanding the needs from a business and project standpoint. Identifying the critical factors of data requirements could be achieved through multiple brainstorming sessions with the business and among the team members. It was an essential step, as the collected data forms a necessary foundation for future analysis and model development stages. The goal here was to detect the transportation modes. Therefore, the business understanding phase enhances the fact that the collected data must include the location features such as travel start and end time, latitude, longitude, and the date of travel information. A few more critical requirements for a successful implementation were a large dataset with unbiased records and features that make sense from the business and technical perspective. Emphasis must be put on the quality of data and the measures. Once the requirements were identified, data collection plan gets devised.

Data Collection Plan

As per the project's requirement, it was essential to understand the travel time, latitude, and longitude to calculate and analyze the travel mode. Based on this, accurate and complete data was required to achieve the outcome. There were several ways to gather the data, and it could be performed manually through a survey, interview, or telephone conversation. Other methods involve tracking (online or transactional), online media platforms such as marketing and social media, and instrument-based, which could involve non-humans or humans as subjects.

During the collection process, it was essential to adhere to the policies, compliance, and guidelines set by the data source owner. Furthermore, when considering human subjects for this process, it was critical to adhere to the policies, the privacy act, and regulations and ensure that a user's personal information was not put at risk and was adequately protected while using it. Many studies have used surveys, telephonic, or manual approaches to collect the user's transportation mode-related information. Still, the quality and the extent of the data collected were insufficient and sometimes inaccurate.

Thus, the approach suggested here to overcome this relies on a GPS, a predominantly used sensor for travel mode identification and classification. Figures 32 and 33 depict the data collection plan for the eight features involved in this study. The features which needed to be collected mainly focus on getting sufficient information keeping the project's requirements in mind. Such features were a user's id and location information such as latitude, longitude, travel date, and time information related to a trip's start and end and altitude. These data collaboratively help extract additional features such as velocity and speed. The data collection plan will provide an overview of the feature's characteristics, which were grouped into various criteria such as what kind of data, who collected the data, when it was collected, and the instruments involved.

Figure 32

Data Collection Plan

DESCRIPTION OF DATA COLLECTION	
Why are we collecting the data?	Data collection is very useful in business decision-making and analytical process. Though several factors account for influencing the decision-making and behavior of a person while traveling, there is one predominant factor that always comes atop, which is travel mode. Detecting these modes will help determine the traveling patterns and also aids in attaining additional insights through the data Hence in order to classify different travel modes using common characteristics, it is necessary to track all the trajectories.
How will the data help?	The data recorded can be helpful in various scenarios and also it can help attain insights on solving some existing problems such as devising methods to eliminate the concentration of traffic in a specific, predicting the travel modes based on trajectories recorded. It can also be helpful in designing township plans which are more efficient.
What should we do after collecting data?	The data collected often is raw and might contain some impurities. Hence an intensive wrangling process will be applied to extract the clean dataset. The redundancies and nulls also should be handled. In case any additional quantifiable features can be derived from this, then those should be extracted. The target variable is a multi-class classification with non-numerical data, the integrity of this should be maintained.

KEY VARIABLES - FEATURE DESCRIPTION					
Descriptive Criterias		**FEATURE 1**	**FEATURE 2**	**FEATURE 3**	**FEATURE 4**
What?	**Variable title**	People_Num	Time	Travel Start Time	Travel End Time
	Input (X) or output (Y) variable?	Input			
	Unit of measurement	Numbers	Date-Time	Date-Time	Date-Time
	Data type	Number	Date-Time	String	String
	Collection method	Log	Log timestamp	Log timestamp	Log timestamp
	If manual	Yes	No	No	No
MSA	**Gauge/instrument**	NA	Internal clock	Internal clock	Internal clock
	Location	Smartphone			
	Gauge calibrated?	NA	Yes	Yes	Yes
	Measurement system checked?	NA	Yes	Yes	Yes
	Precision (R&R) adequate?	NA	Yes	Yes	Yes
	Accuracy adequate?	NA	Yes	Yes	Yes
Sampling	**Minimum sample size (MSS)**	2000000			
	Sampling frequency	1/GPS location point	1/GPS location point	1/travel sequence	1/travel sequence
	Sub-grouping needed?	No			
	Stratification needed? (time, shift)	No			
Who?	**Data collector**	Microsoft Research Asia			
	Operational definition exist?	Yes			
	Data collector trained?	Yes			
	Resources available for data collector?	Yes			
When?	**Start date**	4/1/2007			
	Due date	8/1/2012			
	Duration (in days)	1949			

Figure 33

Data Collection Plan (Continued)

DESCRIPTION OF DATA COLLECTION					
KEY VARIABLES - FEATURE DESCRIPTION					
Descriptive Criterias		FEATURE 5	FEATURE 6	FEATURE 7	FEATURE 8
What?	Variable title	Lat	Lon	Alt	Transportation Mode
	Input (X) or output (Y) variable?	Input	Input	Input	Output
	Unit of measurement	Decimal Value	Decimal Value	Meter	Category
	Data type	Decimal	Decimal	Decimal	String
	Collection method	Sensor	Sensor	Sensor	Log
	If manual	No	No	No	Yes
MSA	Gauge/instrument	GPS Receiver	GPS Receiver	GPS Receiver	NA
	Location	Smartphone			
	Gauge calibrated?	Yes	Yes	Yes	NA
	Measurement system checked?	Yes	Yes	Yes	NA
	Precision (R&R) adequate?	Yes	Yes	Yes	NA
	Accuracy adequate?	Yes	Yes	Yes	NA
Sampling	Minimum sample size (MSS)	2000000			
	Sampling frequency	1/GPS location point			
	Sub-grouping needed?	No	No	No	No
	Stratification needed? (time, shift)	No	No	No	No
Who?	Data collector	Microsoft Research Asia			
	Operational definition exist?	Yes			
	Data collector trained?	Yes			
	Resources available for data collector?	Yes			
When?	Start date	4/1/2007			
	Due date	8/1/2012			
	Duration (in days)	1949			

Features one to seven are input variables, and the last feature is an output variable, transportation mode. The data type of these features was decimal, string, and date-time. Internal clocks from smartphones were utilized to collect GPS data from the users. Data collection was done at a frequency of one record per GPS location point. Data sampling will be done, where the minimum sample size was set to two million records. The duration for which this data was collected is from January 4th, 2007, to January 8th, 2012. As discussed, it helped gain more insights into the additional features and factors that could help determine the transportation modes.

Dataset Description

An appropriate dataset was chosen based on the requirements of the collection plan. The dataset used in this research is a GPS trajectory dataset collected over five years, from April 2007 to August 2012. There are 182 users of the collected data by Microsoft Research Asia. The number of trajectories collected in this process is 18,670, and it consists of sequential timestamped points which contain information related to the latitude, longitude, and altitude. The data was collected using various GPS loggers and phones and sampled at multiple rates. This collection process involved two segments of users: one carries GPS loggers for years, and the other carried them for only a few weeks. The records logged in dense representation account for 91.5% of the dataset. Dense representation implies that the records collected are in an interval of five to ten meters or one to five seconds. The data is collected mainly from the cities of China.

The dataset contains the logs of usual movements like going to work and home. Furthermore, the records also included other activities like entertainment and sports, such as shopping, dining, sightseeing, jogging, hiking, and cycling. The volume of the data is 1.67 GB, and the format of the data collected here is in a vector-based plotter(.plt) file and a text file showing the modes of travel for each GPS log based on the user's trajectory timestamp. The Microsoft Research License Agreement was accepted while collecting the data from the Microsoft site. As per the agreement by Microsoft. (n.d.), the data could be used for non-commercial purposes, such as academic research. Some other compliance guidelines include: not removing copyrighted content, distribution in any form not allowed, and prohibiting modifications. Furthermore, the patent rights are not applied to the derivatives of work, and the agreement is controlled by the laws of the State of Washington, USA (p. 4). These conditions are respected and followed throughout the project duration.

The figures below represent the .plt and .txt files. The .plt file contained the trajectory details recorded grouped based on users and various timestamps, whereas the .txt file included the travel modes associated with each recorded trajectory for multiple users. Figure 11 displays a sample of the recorded trajectory, and figure 12 showcases the travel modes.

The raw data shown above contains the trajectories recorded, which was then passed through various preprocessing stages, and additional insights can be attained. Some of the features of the original dataset can be unclean and contain corrupt information. So, it was essential to perform quality control procedures to prevent such misinformation from spreading. The processes involved will be explained in the next section.

Exploratory Data Analysis Plan

A data exploration plan needed to be devised so that it can be executed on the derived and raw features, which will help gain more insights about the data at hand. Before devising the exploratory analysis plan, specific observations made through eyeballing must be addressed. Some of those observations involve eliminating the records of wrong stature and transformation, which are addressed in the following section, succeeded by the exploratory analysis plan. Figure 34 shows the list of cleaning, transformation, and feature extraction operations planned to be carried out before the exploratory analysis begins. The additional features must be efficient to aid the successful implementation of the model. Some derived features that were planned to be extracted are described in table 8 which will help attain more insights.

In addition to the features collected from the raw dataset, some additional features must be generated to detect the modes effectively and efficiently. Extraction of derived features involves aggregation and calculations using the original feature so that additional insights can be attained. The derived feature extraction will be explained in the data transformation section

Table 1 portrays the additional features planned to be derived from the available raw data. The first one, "Travel Count," shows the split of travel segments. If the travel start time was not the same and the delay between two trips was more than 30 minutes, then a travel count number was added accordingly. The second one is "Time Gap(s)," which calculates the difference between the current time of the travel mode for a user and the next consecutive time of travel in seconds for the same travel segment.

The next field, "Distance(m)," records the difference in distance between two consecutive travel modes in a travel segment in meters. Speed was calculated by dividing the distance by the time gap and was represented by the field "Speed(m/s)." The last one, "Acceleration(m/s^2)," depicts acceleration, calculated by dividing the difference between two consecutive speeds over the corresponding time gap.

Table 8

Additional Features Planned to be Extracted

Field	Description	Type
Field 1	Travel Count	Number
Field 2	Time Gap(s)	String
Field 3	Distance(m)	String
Field 4	Speed(m/s)	String
Field 5	Acceleration(m/s^2)	String

A data exploration plan was devised to be executed on the derived and raw features, which will help gain more insights about the data at hand. The exploration plan identifies the existing relationship, which can be used to understand the pattern and distribution better. In addition, it also detects outliers, which can impact the model's performance in the upcoming stage.

Figure 34

Exploratory Analysis Plan – Transformations, Cleaning, and Extractions

Category	EDA Task	Description
Data Wrangling	Handling Nulls	Checking the presence of null values and decide on the optimal way to handle it
	Redundancy Check	Checking for the presence of duplicate records and handling them
	Eliminate rows with incorrect latitude and longitude	Eliminate the entire row incase of wrong latitude and longitude values. For instance, the latitude is greater than 90 or less than zero; the longitude is greater than 180 or less than -180
	Eliminate records with same start and end time	Eliminate the records where the start and end time are the same
	Eliminate records with different travel modes for same time period	Eliminate all the records in case if the time period is same and have different transportation modes.
Transformation	Check the datatype	Assign the right datatype for the features
	Check uniformity of datetime features	Check the datetime format followed universally is the same
	Check the uniformity of decimal placement	Check the number of decimal places are uniform all across the dataset.
Additional Feature Extraction	Splitting travel segments	If two consecutive trips do not have the same travel start time and if the time difference is more than 30 minutes, then a new feature 'Travel Count' is added to split the travel segments
	Remove trips with less difference in time	If two consecutive trips fall into the same travel count feature and they have a time gap of less than 50 seconds, then it will be dropped from the dataset.
	Time gap calculation	If two consecutive trips have the same travel count then their difference in time will be assigned as the time gap between then. If they do not have the same travel count then 'N.A' will be assigned.
	Distance gap calculation	If two consecutive trips have the same travel count then their difference in latitude and longitude will be assigned as the distance gap. If they do not have the same travel count then 'N.A' will be assigned.
	Speed calculation	If the time gap is not 'N.A' then speed will be calculated by dividing respective distance gap over time gap. For others it will be assigned as 'N.A'.
	Acceleration calculation	If calculated speed is not 'N.A', then the division of the difference between the speeds of two consecutive trips over the time gap is calculated as acceleration. The others will be assigned as 'N.A'.

The first step in this process was identifying the distribution of various features, which can be achieved using several available graphical approaches, such as histograms, boxplots, and scatterplots. Some of the identified scenarios which were explored are, Top five transportation modes based on the time gap, Average distance per transportation mode, Average time gap per

transportation mode, Average speed per transportation mode, Number of records based on the transportation modes, Histogram of speed Histogram of the time gap Histogram of distance.

Insights to be Attained from Exploratory Data Analysis Planned

The scenarios mentioned above help gain insights into the collected data. And these insights, in turn, lays a foundation for understanding the data in a better way. Additionally, it also assists in comprehending how these features are feasible and can be used effectively to achieve the project's goal. Understanding the features also aids in identifying and analyzing the underlying redundancy, unnecessary features, or discrepancy factors that could influence the model's performance and prediction result. Furthermore, it aids in determining the appropriate direction the project should take. Exploring the original features and the additional ones mentioned in the previous section helps identify other potential features that can be derived from these and can be utilized to enhance the model's efficiency.

Additionally, it helps determine any underlying bias in the data, and if it exists, then a contingency plan can be prepared to address this. The exploratory analysis is performed to calculate the average distance, speed, and time gap for each transportation mode, help identify the data that fall in a specific range and give an outline of the expectation from a particular mode regarding these factors. Moreover, histograms of speed, distance, and time gap present the distribution of modes across the data, and it also enables detecting the category to which most of the data belongs, and in case any elimination of such data is required, then it can be performed.

Furthermore, other exploratory analyses could be performed through data profiling of pandas. This analysis gives an overview of each feature, such as the maximum, minimum, mean, and standard deviation measures. It also helps detect the unique records present for each feature and any missing pattern. In addition, the data profiling also shows the features' interaction, which

helps determine the relationship and dependency between them. Another helpful task that can be performed is the exploration of the correlation matrix. To summarize, the exploratory analysis helps understand the data from all the different dimensions, which will be helpful will building and evaluating the model. Before proceeding with the implementation of the exploratory analysis plan, a few inconsistencies should be handled and are explained in next section.

Data Pre-Processing

Data cleaning is an essential part of data management. Data is collected in various ways to accomplish the project's goal or requirement, and it can be performed manually through a survey, interview, or telephone conversation. Other methods involve tracking (online or transactional), online media platforms such as marketing and social media, and instrument-based, which can include non-humans or humans as subjects. In the case of this project, which is travel mode detection, the data collected was through GPS trackers over a long period condensed into a .plt and .txt file format . The files could sometimes be impure.

The collected data may sometimes be inaccurate, missing, redundant, or inconsistent, leading to incorrect prediction in the modeling phase and, thus, the inability to achieve the project's goal. Therefore, data cleaning involves reviewing the data to remove or update any dirty ones, thus ensuring a better quality of the data used in the subsequent phases. Data pre-processing involves a wide range of procedures to help eliminate discrepancies within the extracted raw data. Hence, an intensive pre-processing process was performed with utmost attention, which will be explained in the following section.

Format Conversion and Original Features Descriptions

Before executing the exploratory data analysis, it was necessary to make sure the raw dataset was merged into one file and converted into an apt format. Hence, the data collected

undergoes further transformation. The users assessed for this study have trajectory-related data in a .plt file and the respective modes in a label file in .txt format. The .plt files in the dataset were a cluster of individual files containing records of every user's data based on time stamps. For working in the following stages, it was crucial to merge the .plt and label files into a suitable format to attain better insights. Therefore, based on the user, these two files were converted and merged into a comma-separated values (CSV) file format (see Appendix A for the code of this process). Figure 35 below represents a sample of this merged file and table 9 represents the fields in the generated file and the respective types.

The first field in Table 9 is "People_Num," which shows the user id. The second is "Time," which defines the recorded time in YYYY-MM-DD HH:mm:ss format. The next two features listed in table 9 are travel start time and end time, representing the start and end timestamps, respectively. The fields "Lat'" and "Lon" also show the latitude and longitude information recorded to six-digit decimal degrees. The feature "Alt" records the altitude in feet, and an altitude entry of "-777" is considered invalid. Furthermore, the last one is the "Transportation Mode," which shows the mode of transport. The data collection plan provided an overview of the feature's characteristics, which can be grouped into various criteria such as what kind of data, who collected the data, when it was collected, and the instruments involved.

Figure 35

Sample of User's Log and Label Merged File

	People_Num	Time	Travel Start Time	Travel End Time	Lat	Lon	Alt	Transportation Mode
0	104	2008-03-28 08:44:30	2008-03-28 08:42:00	2008-03-28 09:50:00	39.962098	116.301595	0.0	bus
1	104	2008-03-28 08:48:30	2008-03-28 08:42:00	2008-03-28 09:50:00	39.948270	116.303298	0.0	bus
2	104	2008-03-28 08:48:33	2008-03-28 08:42:00	2008-03-28 09:50:00	39.948220	116.303337	0.0	bus
3	104	2008-03-28 08:48:39	2008-03-28 08:42:00	2008-03-28 09:50:00	39.948120	116.303378	0.0	bus
4	104	2008-03-28 08:48:42	2008-03-28 08:42:00	2008-03-28 09:50:00	39.948110	116.303418	0.0	bus
5	104	2008-03-28 08:48:48	2008-03-28 08:42:00	2008-03-28 09:50:00	39.947997	116.303410	0.0	bus

Table 9

Merged CSV File Fields

Field	Description	Type
Field 1	People_Num	Number
Field 2	Time	Datetime
Field 3	Travel Start Time	String
Field 4	Travel End Time	String
Field 5	Lat	Decimal
Field 6	Lon	Decimal
Field 7	Alt	Decimal
Field 8	Transportation Mode	String

Handling Incomplete/Missing Data

Missing or incomplete data can happen for many reasons, such as human errors or machine issues. In the case of collecting data by survey, a respondent may need help understanding the questionnaire, or a matching option is not present to answer, or it might happen because the respondent is not interested in replying to a particular question. In other scenarios where the collection method involves instruments, there may be some error in the machine while recording the observation, the user not utilizing it in the observation period, or it is a faulty instrument that does not work as expected. Thus, to avoid any misleading results caused by these kinds of data, a proper strategy was implemented to address the issues explained in the next step (see Appendix A for the code of the entire pre-processing steps).

Handling Users with Missing Target Label .txt Files

The proposed project aims to classify vehicle modes based on users' trajectory data with high accuracy and thus seeks to generalize the latent patterns present in the trajectories data to classify vehicle modes for various localities accurately. Therefore, it was crucial to identify and consider only those data where proper travel mode labels are associated with the other logging

records, such as user ID, travel start, end time, and altitude. Thus, all the trajectory files were verified to check whether a label text file was associated with it. If the file was present, all those trajectories were considered for subsequent steps. If a user's trajectory .plt file did not have a .txt label attached, it was eliminated from the subsequent steps. This process of elimination ensured the identification of only the potential users whose transportation labels could be predicted successfully. In this process, 73 potential user log files were identified and selected for the next stage. Other determining steps might influence the decision regarding the file selection and will be discussed in the forthcoming sections.

Handling Nulls

As mentioned in the previous section, there can be many reasons for missing values, and thus this leads to null values in the resulting dataset. The reason that some features may have null values could be attributed to a large number of identified users and the massive size of the dataset. Many strategies and approaches exist to handle null values, such as deleting the corresponding rows or columns if null exists for more than 50% of the data or replacing the values with the central tendency factor(mean, median, or mode), or using different interpolation strategies. Figure 36 below shows the result of this function call and shows that there are zero null values.

Figure 36

Number of Null Values in Each Feature

```
People_Num            0
Time                  0
Travel Start Time     0
Travel End Time       0
Lat                   0
Lon                   0
Alt                   0
Transportation Mode   0
```

Many python libraries are available to handle the null values, such as Pandas and NumPy. For this project, the Pandas library was used, and the 'IsNull' function was called, which checks if any missing and null values are present. Figure 36 shows that for each feature, the sum of null records is zero; thus, no null or missing values exist in the dataset, and data is free from this kind of record. Since the source file does not contain null records, the dataset can be passed on to the next cleansing stage without any modifications. Before proceeding with further cleaning, an initial exploratory analysis was performed to understand the data further so that it could be helpful in further pre-processing stages. The initial exploratory analysis performed will be explained in the forthcoming sections.

Handling Noisy Data

While recording or collecting data, some unwanted data items or features that do not help explain the features or play any role in establishing a relationship between the target and feature may get retrieved in the process. Additionally, it could affect the model's prediction accuracy and cause skewness in the conclusion. Therefore, it was necessary to identify and handle these kinds of data.

Eliminating Outliers in Latitude and Longitude

The collected GPS trajectories contain location information for each user's associated travel, and thus the latitude and longitude are logged. There are certain abnormalities that can be directly observed since the optimal values always falls within a range for these geo-graphic features. A latitude is invalid if it does not fall between zero and 90 degrees, and similarly, a longitude that does not fit in the range of 180 or -180 degrees is considered incorrect. Figure 37 below depicts some of the sample records with outliers in latitude longitude and altitude whereas Figure 38 below shows how there was no such records existing after elimination.

Figure 37

Sample Records with Outliers in Latitude, Longitude and Altitude

People_Num	Time	Travel Start Time	Travel End Time	Lat	Lon	Alt	Transportation Mode
104	2008-03-28 08:44:30	2008-03-28 08:42:00	2008-03-28 09:50:00	91.06	117.01	0	bus
104	2008-03-28 08:48:30	2008-03-28 08:42:00	2008-03-28 09:50:00	91.02	118.01	0	bus
104	2008-03-28 08:50:03	2008-03-28 08:42:00	2008-03-28 09:50:00	42.94	181.02	0	bus
104	2008-03-28 08:50:26	2008-03-28 08:42:00	2008-03-28 09:50:00	39.94	181.03	0	bus
104	2008-03-28 08:49:14	2008-03-28 08:42:00	2008-03-28 09:50:00	39.95	116.3	-777	bus

Figure 38

After Elimination of Outliers in Latitude, Longitude and Altitude

Result after elimination of records with abnormal latitude and longitude:

```
No trajectories with abnormal latitude and longitude are present after elimination.
```

People_Num	Time	Travel Start Time	Travel End Time	Lat	Lon	Alt	Transportation Mode

Result after elimination of records with invalid altitude:

```
No trajectories with an altitude of -777.
```

People_Num	Time	Travel Start Time	Travel End Time	Lat	Lon	Alt	Transportation Mode

Handling Outliers in Altitude

The altitude/elevation for each user was also logged along with the latitude and longitude information to get an accurate picture of location data. Since altitude contains too many unique distinct values, it does not offer any insight that can enhance the models' predictions. Though it

could be considered invalid, obliterating the record will result in data loss. Hence, to avoid that, the value for altitude is logged as -777 when invalid. The entire row was not eliminated as a result, and the feature altitude will be eliminated in the forthcoming step. Figure 9 displays sample records with the invalid code -777 assigned to altitude and Figure 10 shows that no such records are present after elimination.

Checking for Records with the Same Start and End Time

Another possibility of the data being noisy in the case of this project was a scenario when a trajectory starts and ends at the same time. Identical start and end times mean that the travel time is zero seconds which is not valuable since no travel mode can be detected. The second similar scenario was if two trips have the same start and end time, indicating that it belongs to the same trip. Additionally, it showed that the user has not changed the travel mode and was in the same mode at that time. There was a possibility of such inconsistencies influencing the prediction of the model. Hence a check was performed to verify the existence of such records. Since there were no such trajectories, there was no change in the record count in this stage.

Eliminate Trajectories with Multiple Travel Modes Starting Simultaneously

Another scenario that could be considered noisy data was related to the relationship between transportation mode and time. Two consecutive recorded trajectories start and end simultaneously and have a difference of zero seconds in time but assigned different travel modes; it could be considered incorrect data. A single GPS tracker cannot have different travel modes for trajectories starting simultaneously. Hence this kind of information can be considered misleading. Thus, this kind of data was considered inconsistent and removed from the dataset. Figure 39 depicts sample records of user 104 with trajectories starting at the same time from the same location but using different modes of transport. Such records were identified and

eliminated. The overall change in record count at the end of the pre-processing stage is depicted in the figure 42.

Figure 39

Sample of Records with Multiple Travel Modes Starting Simultaneously by Same User

People_Num	Time	Travel Start Time	Travel End Time	Lat	Lon	Alt	Transportation Mode
104	2008-03-28 08:51:16	2008-03-28 08:42:00	2008-03-28 09:50:00	39.94	116.30	0	bus
104	2008-03-28 08:51:19	2008-03-28 08:42:00	2008-03-28 09:50:00	39.94	116.30	0	bike
104	2008-03-28 08:51:22	2008-03-28 08:42:00	2008-03-28 09:50:00	39.94	116.30	0	car
104	2008-03-28 08:51:25	2008-03-28 08:42:00	2008-03-28 09:50:00	39.94	116.30	0	walk
104	2008-03-28 08:51:31	2008-03-28 08:42:00	2008-03-28 09:50:00	39.94	116.30	0	airplane
104	2008-03-28 08:51:34	2008-03-28 08:42:00	2008-03-28 09:50:00	39.93	116.30	0	run

Handling Inconsistent Data

Inconsistency is another data quality issue that should be addressed. During the collection process, primarily, the data was accumulated from many sources and then aggregated. Thus, the same information was gathered in a different format as each source maintains it in a unique pattern. For instance, while collecting the travel time for a user, one source may store it in HH:MM format, whereas another source, can have a structure of HH:MM:SS. This leads to inconsistent data in the final dataset to be considered. Therefore, it was essential to identify and handle such data to mitigate their effect on model performance. Some scenarios which could account for the inconsistency factor were identified, and a solution approach was implemented accordingly.

Checking the .plt File Metadata

The vector-based plotter(.plt) file is a trajectory file that records each user's log details based on the time stamp, such as the travel date, time, latitude, longitude, and altitude. The first six lines in each file describe the file, which is not valuable in the modeling phase and thus was

removed before using it in the subsequent steps. The validation of the .plt file metadata was essential since it may not be the same across multiple files. Such discrepancies could lead to data corruption while merging. It was also mandatory to verify that the order of features in the metadata is similar across all the .plt files. If a discrepancy is found, it was mandatory to eliminate that .plt file from the merging process. For instance, the number of columns for one .plt file may be higher or lower than the other one, and it was also possible that the order of the columns appears to be interchanged while comparing. In such cases, the file can be considered corrupt and could lead to data inconsistency. Therefore, it was required that each file was validated to eliminate the inconsistency issue that could have happened due to metadata before initiating the merging process. The result ensured that the metadata remains the same for all the files processed in the subsequent steps. Figure 40 represent the plt file metadata.

Checking the Label File Metadata

The label file contains the transportation modes for each user based on the timestamp in a text file. Each file has three fields: the travel start time, end time, and the corresponding mode of travel. Each label file must be validated, as a possible mismatch in the metadata could lead to data inconsistency and corruption issues. For instance, a label file for a user can have a start and end time, but the transportation label may be missing. Similarly, another user could have a label file where the transportation label is missing. In this case, when these files are merged, it can lead to missing data in the final dataset and data corruption. Thus, ensuring that each file contains a similar number of columns recorded and follows the same order as required. If any of these issues mentioned above happen, it is necessary to eliminate such files from the merging step. Hence, each label file was checked to ensure that these were resolved before using it in subsequent steps. The metadata of .txt file which contains class labels is depicted in Figure 41 respectively.

Figure 40

Sample Metadata of PLT File

PLT format:

Line 1…6 are useless in this dataset, and can be ignored. Points are described in following lines, one for each line.

Field 1: Latitude in decimal degrees.

Field 2: Longitude in decimal degrees.

Field 3: All set to 0 for this dataset.

Field 4: Altitude in feet (-777 if not valid).

Field 5: Date - number of days (with fractional part) that have passed since 12/30/1899.

Field 6: Date as a string.

Field 7: Time as a string.

Example:

39.906631,116.385564,0,492,40097.5864583333,2009-10-11,14:04:30

39.906554,116.385625,0,492,40097.5865162037,2009-10-11,14:04:35

Figure 41

Sample Metadata of TXT File

TXT Format:

Field 1: Start Time as String

Field 2: End Time as String

Field 3: Transportation Mode as String

Validating the Metadata of the Merged File for Inconsistencies

When two or more files are combined to make the file easier to process, the chances of inconsistencies rise if there is any mismatch in the metadata of these files. The project aims to identify the travel modes; thus, the trajectory details and the corresponding mode details based on a time stamp were required for each user. Consequently, the log(.plt) and the label(.txt) files

were merged for this purpose. The merged file may have data inconsistency or corruption issues, such as a metadata mismatch during this process. Hence, it requires a thorough validation of any such problems. The merged file was reviewed to detect issues with the metadata regarding the number and order of the fields.

In addition, it was crucial to validate the order of the fields to avoid any data corruption issues. Hence, a proper check was done to detect such scenarios in the merged file. If such an issue existed, the trajectories that violated this were dropped from the subsequent steps. Figure 42 shows the metadata of the merged .csv file with its respective data types.

Figure 42

Metadata of Merged File

```
People_Num              int64
Time                   object
Travel Start Time      object
Travel End Time        object
Lat                   float64
Lon                   float64
Alt                   float64
Transportation Mode    object
dtype: object
```

Handling Duplicates

Redundant or duplicate data is a common data quality issue. It can occur for various reasons, such as multiple representations of the same data and erroneous entry of the exact user details. If this redundancy is not identified and removed, then this leads to overfitting the model in the future. It will work well with the training data but will not generalize better for the new test dataset. Additionally, it can cause an issue in the validation step. If the same data is present for both the training and validation set, then the performance for the latter will be higher as the model has learned about the same data in the training phase. Therefore, removing the redundant

records in the cleaning phase was necessary to avoid the abovementioned issues. This project checks this issue by calling the "duplicated()" function on the dataset provided by the Pandas library, which returned 41046 redundant entries. All these identified records were dropped while keeping the last occurrence of each instance intact. Pandas drop_duplicates() method was used to handle such redundant occurrences. Figure 43 shows samples of the duplicate records and figure 44 shows the changes in record count.

Figure 43

Samples of Duplicated Records

People_Num	Time	Travel Start Time	Travel End Time	Lat	Lon	Alt	Transportation Mode
167	2008-06-05 01:57:49	2008-06-05 01:56:58	2008-06-05 02:17:45	40.01	116.32	221.0	bike
167	2008-06-05 01:57:52	2008-06-05 01:56:58	2008-06-05 02:17:45	40.01	116.32	221.0	bike
167	2008-06-05 01:58:06	2008-06-05 01:56:58	2008-06-05 02:17:45	40.01	116.32	169.0	bike
167	2008-06-05 02:00:02	2008-06-05 01:56:58	2008-06-05 02:17:45	40.01	116.32	123.0	bike
167	2008-06-05 02:00:07	2008-06-05 01:56:58	2008-06-05 02:17:45	40.01	116.32	122.0	bike
167	2008-06-05 02:00:43	2008-06-05 01:56:58	2008-06-05 02:17:45	40.00	116.32	120.0	bike
167	2008-07-06 10:57:36	2008-07-06 09:15:31	2008-07-06 11:08:55	40.00	116.41	184.0	bus
167	2008-07-09 05:08:53	2008-07-09 04:42:59	2008-07-09 05:22:09	39.98	116.31	270.0	bike
167	2008-07-09 12:34:22	2008-07-09 12:34:11	2008-07-09 12:46:33	39.99	116.33	171.0	bike
167	2008-05-30 04:59:51	2008-05-30 04:15:38	2008-05-30 05:27:51	39.97	116.42	151.0	walk
167	2008-05-30 05:00:36	2008-05-30 04:15:38	2008-05-30 05:27:51	39.97	116.42	155.0	walk
167	2008-05-30 05:02:34	2008-05-30 04:15:38	2008-05-30 05:27:51	39.97	116.42	143.0	walk
167	2008-05-30 05:04:56	2008-05-30 04:15:38	2008-05-30 05:27:51	39.97	116.42	96.0	walk
167	2008-05-22 11:01:01	2008-05-22 10:50:16	2008-05-22 11:14:57	39.97	116.36	313.0	bus
167	2008-05-22 11:14:45	2008-05-22 10:50:16	2008-05-22 11:14:57	39.97	116.42	109.0	bus
167	2008-06-07 03:23:42	2008-06-07 01:02:16	2008-06-07 04:46:47	39.76	118.74	163.0	train
167	2008-06-07 03:26:34	2008-06-07 01:02:16	2008-06-07 04:46:47	39.75	118.80	159.0	train
167	2008-06-07 03:43:31	2008-06-07 01:02:16	2008-06-07 04:46:47	39.70	119.16	108.0	train
167	2008-06-03 00:53:15	2008-06-03 00:44:34	2008-06-03 01:32:39	40.01	116.32	121.0	bike
167	2008-05-21 11:52:30	2008-05-21 10:06:50	2008-05-21 12:54:08	39.98	116.31	243.0	walk

Figure 44 shows the record count before the initiation of preprocessing and how it ended after the various pre-processing steps, such as eliminating inconsistencies, noisy data, and

redundancies. It is worth noting that around 41000+ records were eliminated as the end result of the initial pre-processing. The same is also depicted in the data statistics section.

Figure 44

Record Count Change Before and After

Before

```
People_Num : 2819105
Time : 2819105
Travel Start Time : 2819105
Travel End Time : 2819105
Lat : 2819105
Lon : 2819105
Alt : 2819105
```

After

```
People_Num : 2778059
Time : 2778059
Travel Start Time : 2778059
Travel End Time : 2778059
Lat : 2778059
Lon : 2778059
Alt : 2778059
```

Initial Exploratory Data Analysis

Exploratory analysis of the raw data will help identify the trends and patterns; this will help to understand the data before any assumptions are made. Additionally, it aids in spotting errors, outliers, and the relationship between the variables. A graphical and statistical representation of data during the EDA process will assist in summarizing the critical observation. This section explains various EDA tasks performed based on the plan discussed in the previous section. Figure 45 the descriptive statistics of the raw data before processing. It summarizes some fundamental statistics, such as the central tendency for the quantitative features, number of records, frequency, and unique records. It shows more than 2.8 million records present, and the number of unique transportation modes is 11. Figure 46 represents the descriptive statistics of the data after performing some necessary pre-processing steps. The data was checked and handled for any missing or inconsistent issues such as null or duplicate records or datatype mismatch.

After conducting these steps, the figure shows that the number of records had reduced and the travel time falls under the range of 12th April 2007 and 31st December 2011.

Figure 45

Descriptive Statistics of the Raw Data Before Pre-Processing

	People_Num	Time	Travel Start Time	Travel End Time	Lat	Lon	Alt	Transportation Mode
count	2.819105e+06	2819105	2819105	2819105	2.819105e+06	2.819105e+06	2.819105e+06	2819105
unique	NaN	2232852	5039	5037	NaN	NaN	NaN	11
top	NaN	2011-10-19 01:02:06	2011-09-07 10:04:33	2011-09-07 13:56:58	NaN	NaN	NaN	walk
freq	NaN	6	22080	22080	NaN	NaN	NaN	833440
mean	1.021490e+02	NaN	NaN	NaN	3.974628e+01	1.146419e+02	3.465051e+02	NaN
std	3.790700e+01	NaN	NaN	NaN	2.505067e+00	2.009298e+01	1.653855e+03	NaN
min	5.200000e+01	NaN	NaN	NaN	1.824990e+01	-1.799696e+02	-2.306100e+04	NaN
25%	6.500000e+01	NaN	NaN	NaN	3.993991e+01	1.163125e+02	8.700000e+01	NaN
50%	8.500000e+01	NaN	NaN	NaN	3.997623e+01	1.163291e+02	1.500000e+02	NaN
75%	1.280000e+02	NaN	NaN	NaN	3.999660e+01	1.164019e+02	2.140000e+02	NaN
max	1.750000e+02	NaN	NaN	NaN	5.876549e+01	1.799969e+02	5.196190e+04	NaN

Figure 46

Descriptive Statistics of the Raw Data After Pre-Processing

	People_Num	Time	Travel Start Time	Travel End Time	Lat	Lon	Alt	Transportation Mode
count	2.778059e+06	2778059	2778059	2778059	2.778059e+06	2.778059e+06	2.778059e+06	2778059
unique	NaN	2232852	5039	5037	NaN	NaN	NaN	11
top	NaN	2011-10-19 00:56:51	2008-06-09 10:54:26	2008-06-09 14:56:52	NaN	NaN	NaN	walk
freq	NaN	6	21258	21258	NaN	NaN	NaN	816918
first	NaN	2007-04-12 10:21:27	2007-04-12 10:21:16	2007-04-12 10:26:25	NaN	NaN	NaN	NaN
last	NaN	2011-12-31 15:18:07	2011-12-31 15:16:52	2011-12-31 15:18:07	NaN	NaN	NaN	NaN
mean	1.026812e+02	NaN	NaN	NaN	3.974320e+01	1.146180e+02	3.514975e+02	NaN
std	3.791926e+01	NaN	NaN	NaN	2.523369e+00	2.023990e+01	1.665171e+03	NaN
min	5.200000e+01	NaN	NaN	NaN	1.824990e+01	-1.799696e+02	-2.306100e+04	NaN
25%	6.500000e+01	NaN	NaN	NaN	3.994009e+01	1.163129e+02	9.190000e+01	NaN
50%	8.500000e+01	NaN	NaN	NaN	3.997645e+01	1.163293e+02	1.509000e+02	NaN
75%	1.280000e+02	NaN	NaN	NaN	3.999703e+01	1.164029e+02	2.160000e+02	NaN
max	1.750000e+02	NaN	NaN	NaN	5.876549e+01	1.799969e+02	5.196190e+04	NaN

In addition to the statistics discussed previously, data profiling on the raw data was performed to gain more insights about the data, such as understanding the correlation between the features or identifying the missing values. Figures 47 and 48 depict the profiling analysis of each feature. It shows that the features did not have any missing values, and most belong to the transportation mode type walk rather than the other categories.

Figure 47

Data Profiling on Features of the Dataset

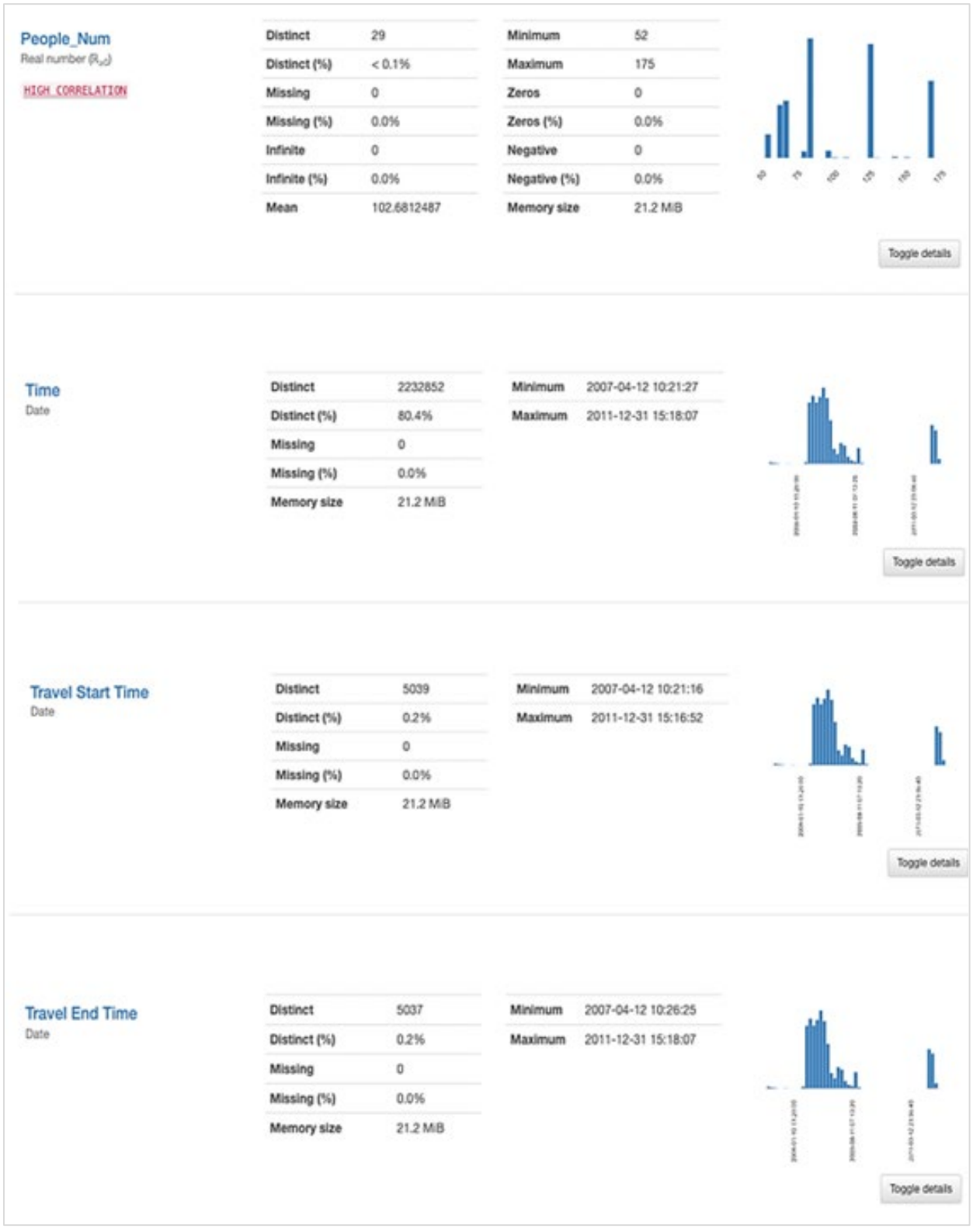


Figure 48

Data Profiling Results on Features of the Dataset Continued

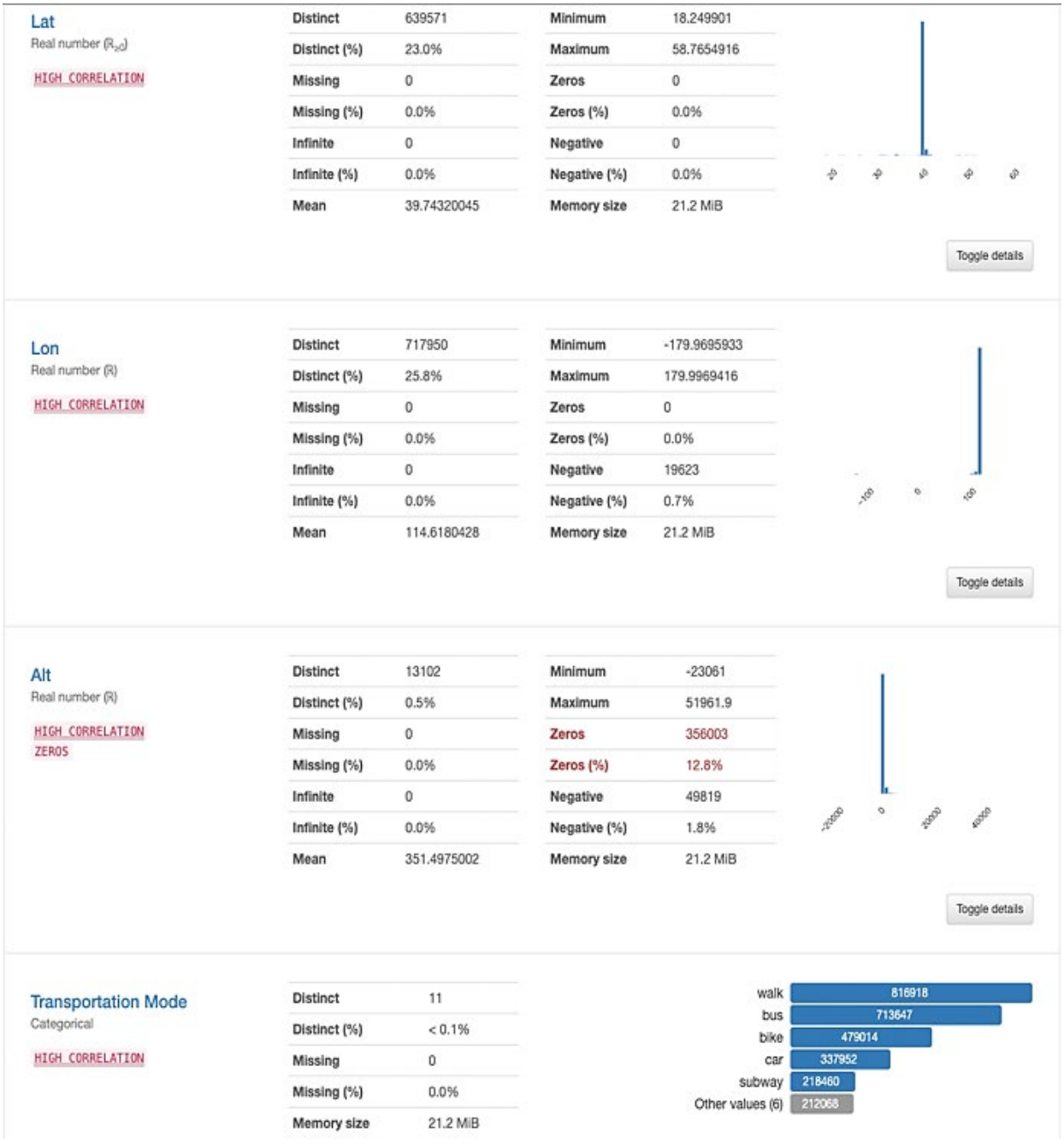
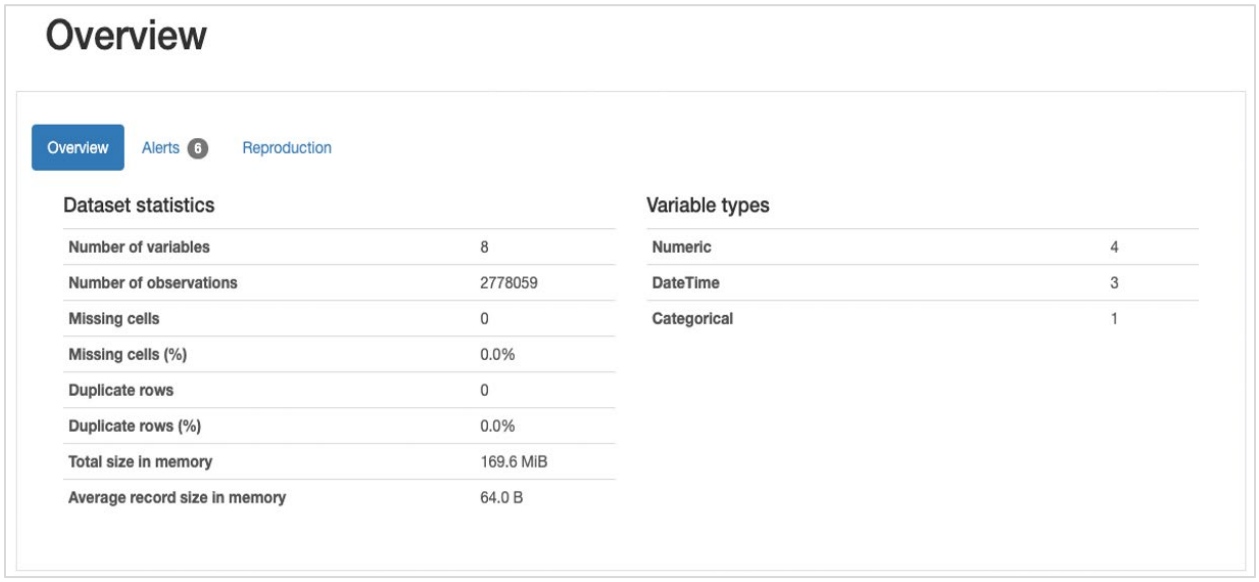


Figure 49 represents the summary of the data profiling step's results after performing the necessary pre-processing steps, such as handling missing values or inconsistencies. The overview is inclusive of all the feature's statistics.

Figure 49

Overall Results of Data Profiling on the Dataset



Observing the above-displayed visualizations, it was evident that eliminating nulls and duplicates resulted in the missing cells and duplicate rows values being zero. Data profiling can also explain the distribution of data, highs, lows, zeros, and negatives in each feature. Understanding the memory utilized by the dataset can help while devising the modeling plan.

Data Transformation

For curating appropriate data, deriving meaningful insights, and making it usable across all systems, it is vital to develop a data transformation strategy. This transformation can be performed for the data's structure, format, or values. Transformation is a crucial step in other data management processes, such as data conversion or integration, as it can standardize, shape, or ensures consistency between datasets. Multiple approaches or methods are available to implement transformation, such as smoothing, aggregation, normalization, standardization, regularization, discretization, and reduction. The transformation goal differs from business to business based on the requirement. Some may need a file conversion into a particular format,

whereas another one can utilize the same file without any modification. Additionally, the dependency on transformation steps is primarily because of the tremendous amount of data being generated. Therefore, this process maximizes the data value and also assists in managing and reducing information overload.

Checking the Consistency of Decimal Places

The number of decimal places for the decimal values can differ from one feature to the other. Data should be rounded off properly, not too much or too little. If it is not done correctly, some valuable information may not be captured. Thus, a strategy should be followed to maintain the same number of decimal places for all the decimal numbers. Primarily, there were three decimal numbers in the dataset; the first one was latitude, the second one was longitude, and the last one was altitude. In general, the location-related information, latitude, and longitude, were stored with 15 decimal digits right to the decimal point. Thus, there was a possibility of a mismatch in the decimal places while recording these values. For instance, for one user, this information was captured so that latitude was three decimal places, but for another, the same field may get recorded with ten decimal places. Such data leads to inconsistency issues, and the resulting conclusion will have a different format. Therefore, for both latitude and longitude, six decimal places were allowed. Thus, all the trajectories had the same format for all the users. Similarly, the data was rounded off to one decimal place for altitude, and the same format was maintained**.**

Verifying Data Types

The feature's data type specifies what kind of values are acceptable. Various data types can represent each feature, such as string, integer, float, and DateTime. In addition, the fundamental performance analysis on the numerical values such as mean, median, and standard

deviation can be performed. The dataset used for this project has different features containing information related to location, travel time, date, and user id. All of these features mentioned above have different types of values which serve a particular purpose. For instance, the "Travel Start Time" represents the starting date and time of a trip for a user. If there is a need to compare the time difference between two trips in seconds or minutes, then it cannot be performed like standard subtraction on numbers. Thus, keeping the data type intact for all the features in the respective field was essential based on the value recorded.

Additionally, if a mismatch exists, this leads to an incorrect feature being derived in the extraction process. Therefore, the location-related features such as latitude, longitude, and altitude were maintained in float data type as it logged the values in decimal degrees. Additionally, the time-related information, such as travel start time and travel end time, had a datatype of Datetime. Lastly, the other features, such as user id and transportation mode, were maintained in string data type. Figure 50 depicts the features before and after data type change.

Figure 50

Assigning Correct Data Types

BEFORE CHANGE

```
People_Num              int64
Time                   object
Travel Start Time      object
Travel End Time        object
Lat                   float64
Lon                   float64
Alt                   float64
Transportation Mode    object
dtype: object
```

AFTER CHANGE

```
People_Num                    int64
Time                 datetime64[ns]
Travel Start Time    datetime64[ns]
Travel End Time      datetime64[ns]
Lat                         float64
Lon                         float64
Alt                         float64
Transportation Mode          object
Travel Count                  int64
dtype: object
```

Figure 50 shows the datatypes that were originally assigned to the dataset while importing and the reassigned datatypes which are more suited for the features.

Derived Features Extraction

The merged CSV dataset contains essential features, but it was crucial to determine what insights could be derived in addition to the existing ones. Hence it was necessary to decide on the features that could be collected from the raw dataset. While deriving, the key factors that should be considered are how it makes the data more quantifiable and how beneficial it was to enhance the models' prediction. The new features must be generated to aid while detecting the modes effectively and efficiently. The additional attributes can also be crucial when performing dimensionality reduction since these tend to be more quantifiable than the rest. In this section, the emphasis was put on such additional derived features and their descriptions.

The first such derived feature would be "Travel count," which could be considered marking all the travels by a user that occur continuously with few pit stops in the middle as a single trip even though there were multiple trajectories recorded for it. The logic behind the generation of this feature was if the travel start time was not the same and the delay between two trips was more than 30 minutes, then a travel count number was added accordingly. The next one was "Time Gap(s)," which calculated the difference between the current time of the travel mode for a user and the next consecutive time of travel in seconds for the same travel segment. The time gap referred to the pitstop time and helped understand whether the trip could be considered singular or multiple. The following field, "Distance(m)," recorded the difference in distance between two consecutive travel modes in a travel segment in meters. Speed was calculated by dividing the space by the time gap and was represented by the field "Speed(m/s)." The last one, "Acceleration(m/s^2)," depicted acceleration, calculated by dividing the difference between

two consecutive speeds over the corresponding time gap. Figure 51 below shows the derived features extracted and explained in this section (see Appendix A for the code of this process).

Figure 51

Sample Records along with Derived Features

	Unnamed: 0	People_Num	Time	Travel Start Time	Travel End Time	Lat	Lon	Alt	Transportation Mode	Travel Count	Time Gap(s)	Distance(m)	Speed(m/s)	Acceleration(m/s^2)
0	0	104	2008-03-28 08:44:30	2008-03-28 08:42:00	2008-03-28 09:50:00	39.962098	116.301595	0.0	bus	1	240	1542.2920914409665	6.43	-0.01
1	1	104	2008-03-28 08:48:30	2008-03-28 08:42:00	2008-03-28 09:50:00	39.948270	116.303298	0.0	bus	1	50	143.9636385353144	2.88	0.19
2	2	104	2008-03-28 08:49:20	2008-03-28 08:42:00	2008-03-28 09:50:00	39.947045	116.303850	0.0	bus	1	58	706.3668121584448	12.18	-0.04
3	3	104	2008-03-28 08:50:18	2008-03-28 08:42:00	2008-03-28 09:50:00	39.940685	116.304043	0.0	bus	1	50	490.10067854245034	9.8	-0.02
4	4	104	2008-03-28 08:51:08	2008-03-28 08:42:00	2008-03-28 09:50:00	39.936278	116.303712	0.0	bus	1	52	452.6519866162259	8.7	-0.08
5	5	104	2008-03-28 08:52:00	2008-03-28 08:42:00	2008-03-28 09:50:00	39.932202	116.303695	0.0	bus	1	52	227.727881360944	4.38	-0.0

Data Aggregation

The original data and the derived extracted features provide insights regarding the data to a more considerable extent, and the derived features appear more quantifiable. Due to this factor, it is advised to look for ways to represent the data in a way that helps enhance the model's predictions. That is when the process of data aggregation comes to play. Data aggregation can be defined as a way to condense the current information and attain a high-level analysis of the features, which will help gain new insights from the data. Data can be aggregated in various ways, such as calculating the minimum and maximum value, percentile, average, and the total sum of the numerical features. Especially in scenarios where records could be combined using specific common features exists. Hence, combining data from multiple sources has been carried out in this step and summarized in a uniform pattern. The feature travel count was added to the records in the previously mentioned steps. A number was assigned to the user's trips by grouping

the trajectories based on specified conditions. Here in this step, this travel count number can be used for further grouping so that statistical information regarding the travel can be calculated.

In conclusion, the fields derived in this step are, firstly, for consecutive trips having the exact travel count, the maximum speed is calculated after aggregating the individual speeds for the corresponding trip. Maximum acceleration was also calculated similarly. The 75th and 95th percentile for speed and acceleration were calculated by retrieving the individual values associated with the similar travel count. Additionally, some features, such as mean and standard deviation for speed and acceleration, were derived similarly. Lastly, the records with similar travel counts were aggregated to obtain the total time and distance. Figure 52 shows the result of data aggregation , and it depicts the newly generated statistical fields after grouping based on the travel count feature (see Appendix A for the code of this process).

Figure 52

Data Aggregation Results

	Travel Count	Transportation Mode	Max Speed(m/s)	95% Speed(m/s)	75% Speed(m/s)	Mean Speed(m/s)	Speed Std	Max Acceleration(m/s^2)	95% Acceleration(m/s^2)	75% Acceleration(m/s^2)
0	1	bus	12.18	10.46	5.87	3.41	3.83	0.19	0.11	0.04
1	2	bus	15.97	15.84	15.51	11.80	5.18	0.19	0.14	0.07
2	5	walk	1.83	1.72	1.51	1.34	0.29	0.01	0.01	0.01
3	7	bus	12.78	11.97	9.67	6.78	3.56	0.16	0.16	0.08
4	8	walk	1.50	1.50	1.45	1.34	0.14	0.01	0.01	0.00

Mean Acceleration(m/s^2)	Acceleration Std	Non 0 Mean Speed(m/s)	Non 0 Mean Acceleration(m/s^2)	Total Time(s)	Total Distance(m)
0.03	0.05	3.41	0.03	3811	6465.72
0.04	0.05	11.80	0.04	939	9533.53
0.00	0.00	1.34	0.00	601	791.24
0.06	0.05	6.78	0.06	866	5771.59
0.00	0.00	1.34	0.00	588	783.50

Additional Exploratory Data Analysis

After extracting the additional derived features, it is advisable to perform exploratory analysis on the derived columns since they are more quantifiable, which could lead to uncovered insights. The section below represents the exploratory data analysis performed after additional feature extraction. Figure 53 depicts multiple bar charts classified based on transportation mode; firstly, it shows the number of modes in each category where the category walk has the highest count of 44,968, whereas contrastingly, the motorcycle has the minor count. Secondly, the average time taken approximately falls between 50 to 101 seconds. Additionally, mode-type airplanes covered the highest average distance compared to the other modes, which is understandable. Lastly, the average speed of the modes has a similar pattern as shown in the average distance.

Figure 53

Measures based on Transportation Mode

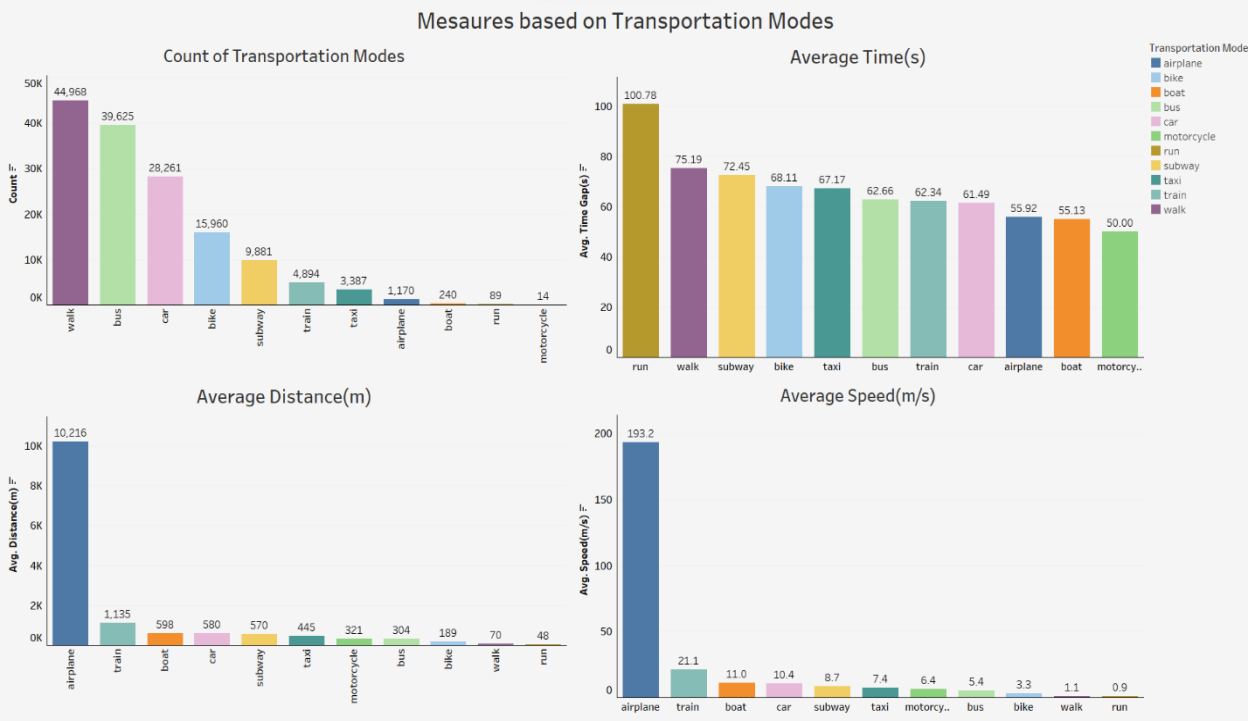
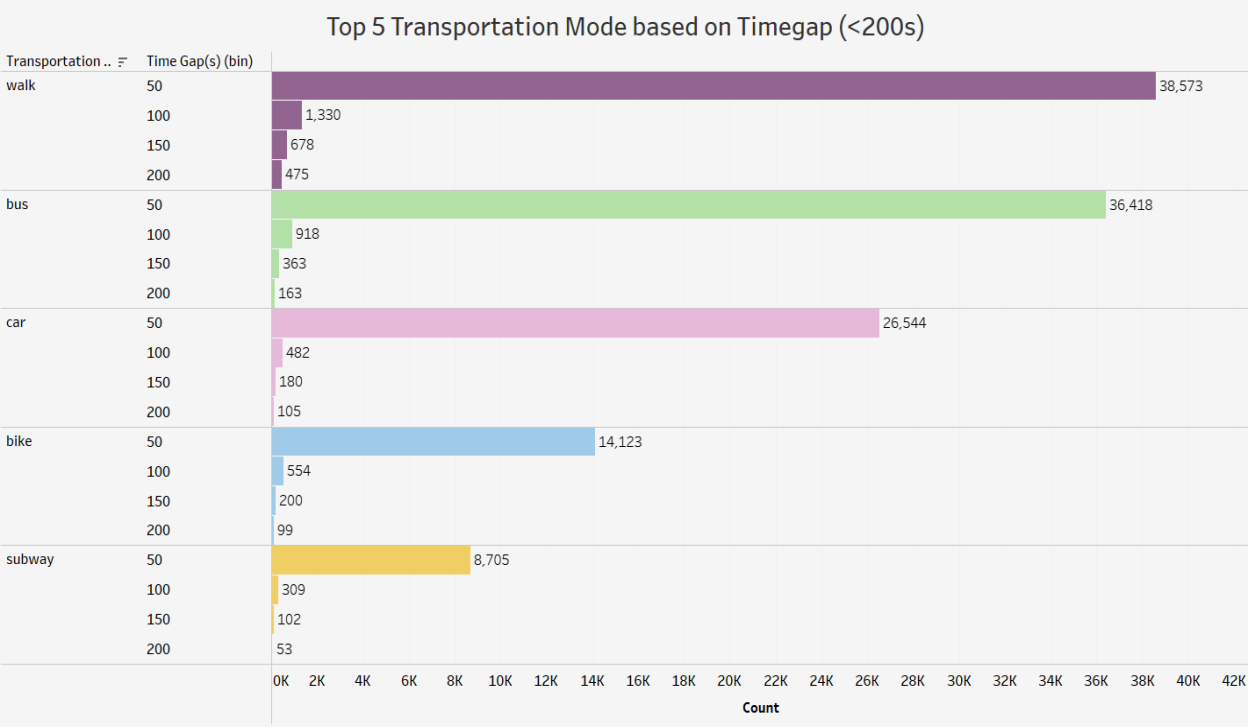


Figure 54 shows the top five transportation modes where the time gap is less than 200 seconds. The reason behind visualizing data in the mentioned time gap range is that most of the data is populated between this range due to the sheer variance in the distribution of modes.

Figure 54

Top 5 Transportation Mode based on Time Gap (<200s)



Figures 55 illustrates the histogram of speed for the entire dataset at the top, and the bottom part visualizes the same, with the data window of range greater than or equal to 30 meters/ second. While considering the entire range, it is observed that a significantly larger volume of data falls under 30 meters/second. This can be emphasized due to specific transportation modes that usually fall in the specified range. It is also vital to consider other transportation modes, such as airplanes which are outliers compared to the others in the performance aspect. Visualizing this data can help attain insights that were not possible with the entire view.

Figure 55

Histogram of Speed and Speed < 30 m/s

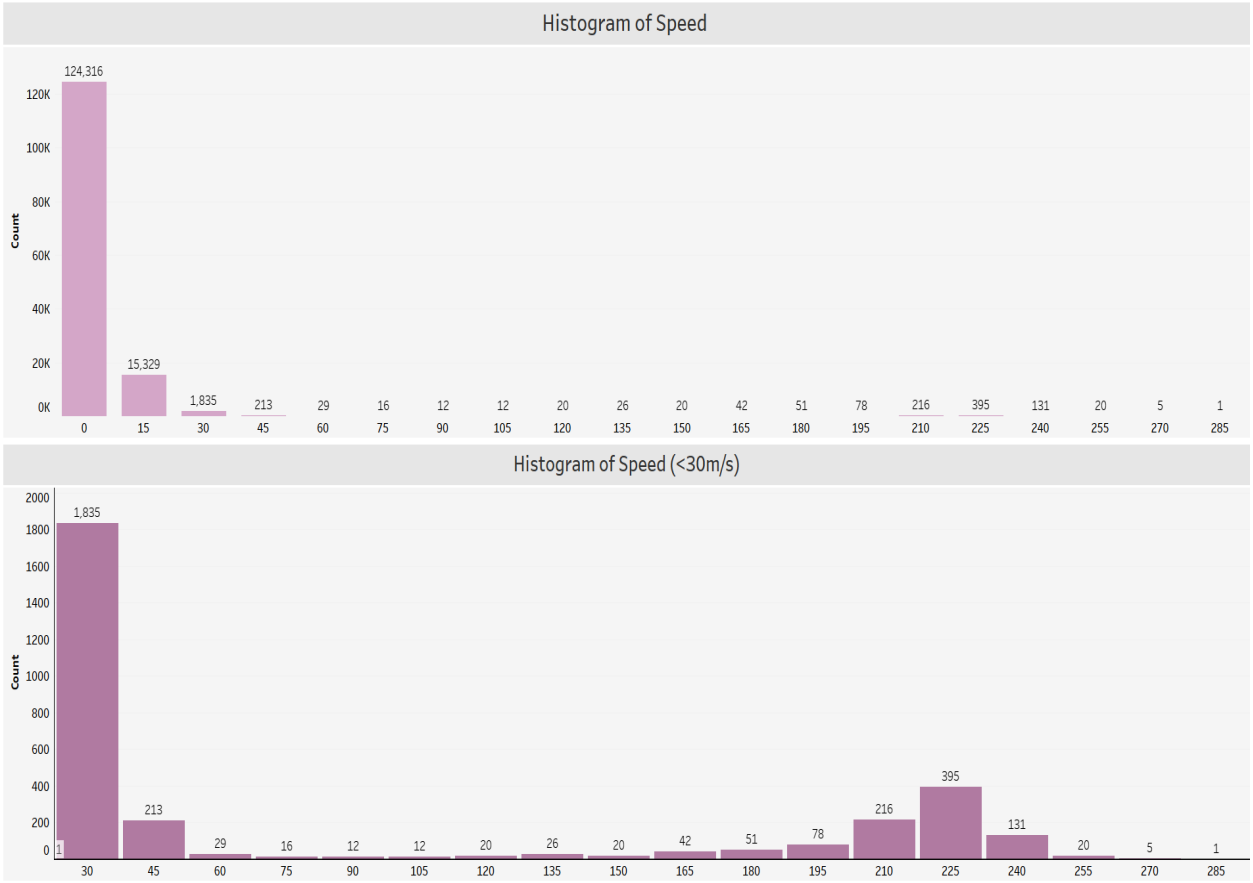


Figure 56 portrays the distance distribution ranging below 1000 meters, showing an exponential curve. It indicates that most of the data falls below 100 meters, and as the distance gradually increases, it shows a decline. The reason behind visualizing only the data below 1000 meters was to understand better the distribution of data in the densely populated section. It also helps understand and gradually predict the mode of transport used for that specific distance in a specified time frame.

Figure 57 depicts the time distribution restricting the range to less than 900 seconds, and it reveals a sudden decline in the number of records from the time shift from 50 to 100 seconds and a gradual drop as the time increases.

Figure 56

Sample of User's .txt File

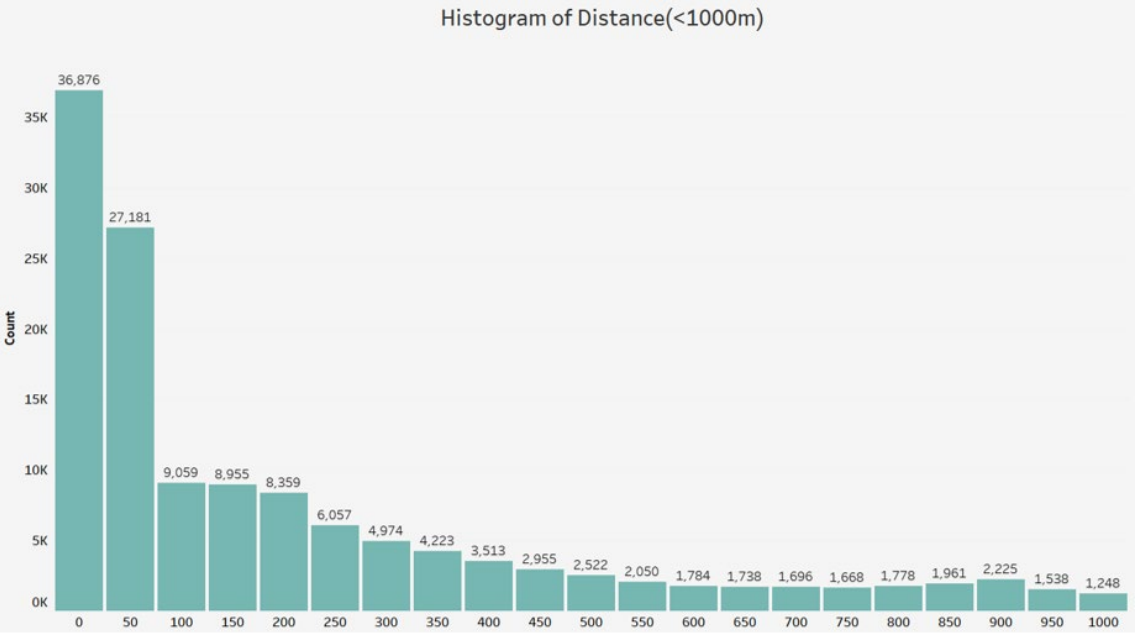
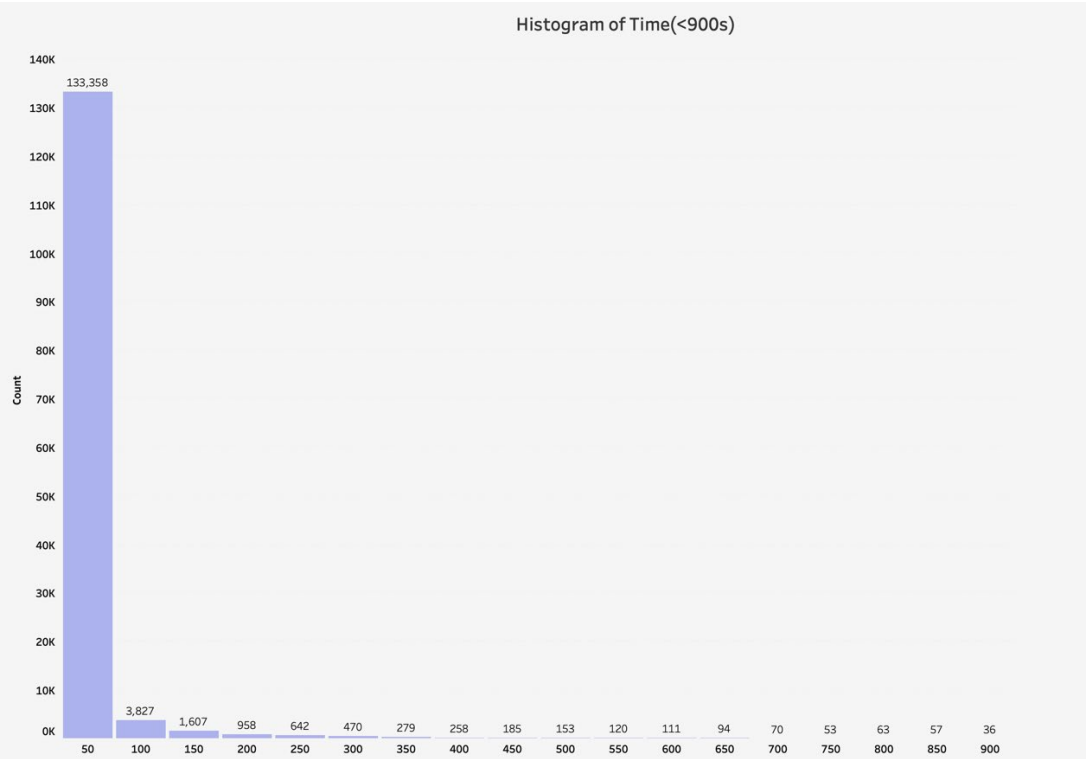


Figure 57

Sample of User's .txt File



Data Standardization

The data often derived from all the pre-processing is sometimes in a normalized range, and the vast difference between the minimum and maximum range of the data can often skew the results. Hence, a necessity arises for converting the data so it can be confined to a standard range. The process of such conversion can be defined as standardization. Standardization is the process of scaling the features in a dataset so that it can fit in a precisely defined range. The technique used to achieve this can be logically explained as fitting the data into a normal distribution curve with the mean being zero and the resulting standard deviation being one. Standardization is mandated because, in most cases, it ensures that the extent of the features is consistent. The possibility of vast outliers can be limited in this way and, as a result, can enhance the predictions.

The standardization was performed in this project using a StandardScaler from the scikit-learn library, implemented during the modeling phase of the project. In this step, the data was transformed into a standardized range and passed on to different models while executing. Figure 58 displays the result of passing the data through StandardScaler.

Figure 58

StandarScaler Results

```
[[-0.57725638 -0.36064775 -0.68514532 -0.71842107 -0.63942502 -0.75000622]
 [-0.60200882 -0.35953161 -0.67269288 -0.71842107 -0.63942502 -0.75000622]
 [ 0.83363229  1.00328162  0.5632115   0.42896792  0.1636005   0.08555144]
 [-0.47747314 -0.29367904 -0.41430481 -0.58343413 -0.37174985 -0.471487  ]
 [-0.07292557 -0.12737341 -0.12478565  0.0240071   0.1636005   0.36407066]
 [-0.41481854 -0.24345251 -0.50458498 -0.58343413 -0.63942502 -0.75000622]
 [-0.60510287 -0.36288004 -0.69448465 -0.71842107 -0.63942502 -0.75000622]
 [-0.34210827 -0.13183799 -0.277328   -0.31346025 -0.37174985 -0.471487  ]
 [-0.62753476 -0.3718092  -0.72872885 -0.78591454 -0.63942502 -0.75000622]
 [ 0.88391067  1.35821578  1.16092848  0.09150057 -0.37174985 -0.75000622]
 [-0.01723259 -0.01799119 -0.03450548  0.09150057  0.1636005   0.36407066]
 [-0.6584753  -0.43431333 -0.7442944  -0.78591454 -0.90710019 -1.02852544]
 [ 1.10358851  0.6740188   1.47223941  0.9014222   0.96662601  1.19962832]]
```

Data Regularization

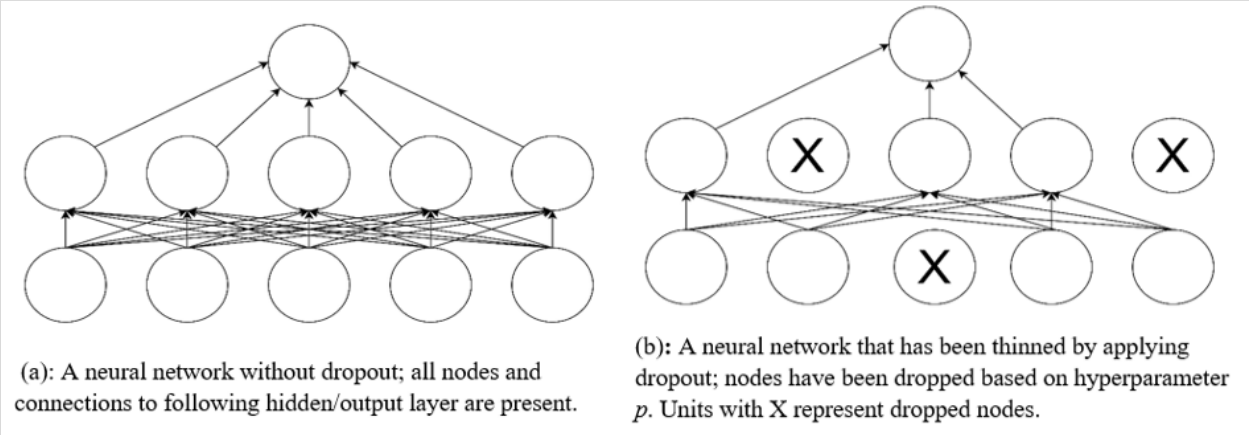
Regularization is one of the transformation techniques which is a part of data engineering. The method can act as a restraint for the model to fit the data precisely and not end up overfitting. The unnecessary complexity of the model can be eliminated with the help of regularization. Especially in cases where a larger volume of data is to be processed, reducing the complexity can be crucial since it can help smoother progress. Overfitting can lead to either abnormally high accuracy or vice versa. Hence it is vital to constantly keep track of the data and check if it is balanced. Poor performance and abnormal prediction can be signification signs that can help identify the overfitting issue. A common way to detect it is by using a validation set with different noise compared to the training set, and if it tends to overfit, then it is a positive case. Drop Out, Lasso or Ridge regularizations are the most commonly utilized regularization techniques among the existing ones. Lasso regularization can be performed by adding a penalty based on the absolute value of parameters which can be scaled by a value often referred to as lambda. Ridge regularization is similar to Lasso's, but it differs because the penalty here is calculated as the square of the parameters instead of absolutes. Dropout is another regularization technique that is more advantageous when using a neural network algorithm. Dropout is relatively simple. It can be explained as eliminating neurons to reduce the model's complexity while not sacrificing its preciseness. Various algorithms were used to test travel mode prediction, and one among the chosen models is LSTM (Long Short-Term Memory). Dropout was opted as the data regularization technique to be implemented in this project.

Regularization is essential for model generalization and prevention of overtraining and will be implemented in the LSTM model through Dropout. Dropout can be implemented between the LSTM layer and one or more subsequent fully connected layers. This regularization

method controls the number of nodes that will be dropped in the network during training. The purpose of Dropout is to prevent co-adapting, where the model becomes too reliant on particular nodes in the fully connected layer. Dropout is controlled by a hyper-parameter and can be optimized. This method is interpretable, and the effects can be approximated during model testing. This method also has the benefit of reducing computational and memory overhead during training by reducing the size of the fully connected network. Dropout was implemented in this project while building the Long-Short term memory model. Figure 59 depicts how the Dropout regularization works.

Figure 59

How Dropout Regularization Works



(a): A neural network without dropout; all nodes and connections to following hidden/output layer are present.

(b): A neural network that has been thinned by applying dropout; nodes have been dropped based on hyperparameter *p*. Units with X represent dropped nodes.

Dropout, when applied, can help address model overtraining. After model training, a loss plot at each epoch can be used to interpret model performance. Figure 30 represents the loss that happens during training and validation. The left image represents overtraining; if loss during validation comes close to training loss but then diverges, this indicates overtraining, and the model will not generalize well. The LSTM can demonstrate overtraining without dropout applied. The right image represents the results once Drop Out is applied. Once applied, though more epochs are needed for validation loss to match training loss, they will not diverge in

subsequent epochs, as shown in Figure 60. Regarding testing after training, the LSTM model with dropout applied demonstrates higher performance with the test data. Figure 61 illustrates a significant gain in predictive performance, showing a greater generalization ability.

Figure 60

Train Vs Validation Loss With and Without Dropout

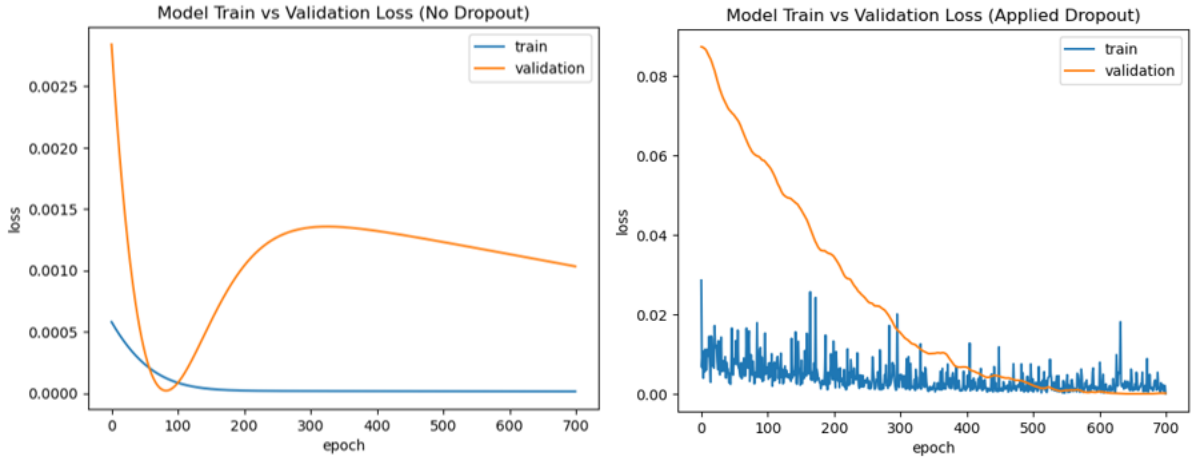
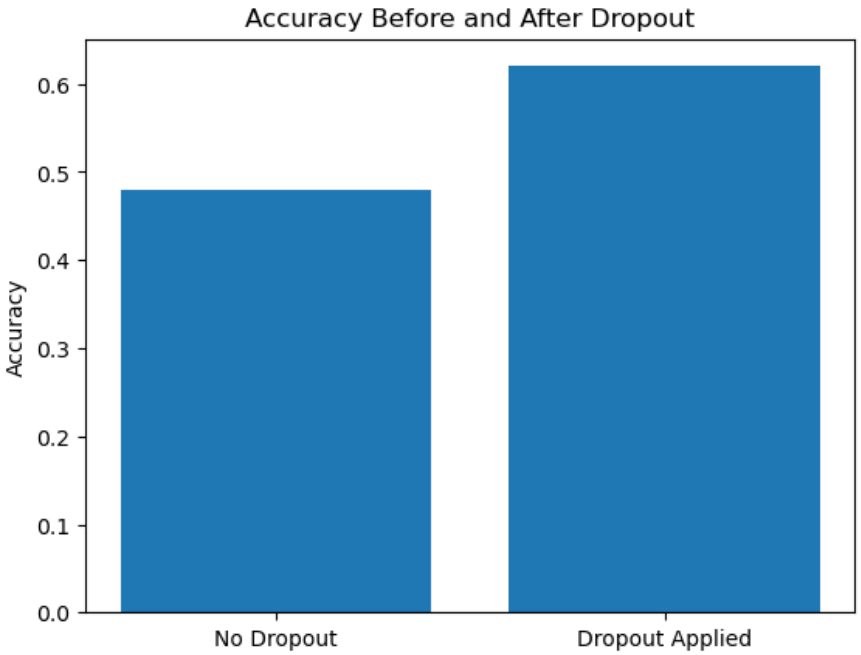


Figure 61

Base Model Accuracy Before and After Dropout



Dimensionality Reduction

A commonly used proverb from the late medieval period, 'too many cooks spoil the broth,' can be applied here when explaining dimensionality reduction. Dimensionality can be explained as the number of features present in a dataset. It is undeniable that too many features can sometimes skew the results of the predictions unfavorably. It is crucial to identify the quintessential features for the model's prediction. Identifying these features can help narrow the decision-making process of eliminating or keeping the rest. When correctly done, Dimensionality reduction can play a crucial role in enhancing the model's results. Various methods exist to implement dimensionality reduction, such as Principal Component Analysis and Singular Value Decomposition. Excluding the pre-defined techniques, it can also be achieved by understanding correlations between the features and the problem at the root level. Certain data have specific characteristics influencing the technique used to deduct the dimensionality. Some may even act left, and deducting the dimensionality can adversely impact the results of predictions. Hence, it is essential to understand the dataset's characteristics and then decide whether to reduce the dimensionality. One advantage of dimensionality reduction is that it can decrease the model's overhead when dealing with larger volumes of data.

In the case of this project, initial dimensionality reduction was performed through eyeball validation, and some statistical features that did not seem to enhance the model's prediction to a significant extent were identified and dropped. Some factors eliminated in this step were attributed to the statistical percentile calculations recorded. The presence of the percentile was beneficial during profiling but could not be considered the same while modeling, especially for models like SVM, which is a support vector machine. The resultant dataset, after dimensionality reduction, can be passed along to the standard scaler, followed by the train and test split.

There also arises a question of why not Principal Component Analysis (PCA). The reason for not using PCA can be attributed to the kind of data utilized in this project. After passing the data through the PCA, a trial run with the Support Vector Machine model was performed. The results of this trial run were compared against another similar run of the Support Vector Machine (SVM) model, but this time without passing the dataset through PCA. The results of both models were compared, and the observations recorded are staggeringly different. First, the cross-validation of the SVM model done using a PCA dataset took longer than the dataset without PCA. This can be because, unlike other dimensionality reduction problems, the unique case of GPS Data should be considered. The clusters created during PCA were added to the original data, and processing prolonged the training time. Figure 62 shows the difference in running time between the models with and without using PCA and we could see why PCA was not the ideal approach. To justify this further, the weighted scores were compared which will be discussed in the following part.

Figure 62

Time Taken for Training and Testing With and Without PCA

```
Training SVC ...
Average CV performance for SVC - Without PCA : 0.72733 (in 1.08944 seconds)
Training SVC ...
Average CV performance for SVC- With PCA : 0.573426 (in 108.128 seconds)
```

Figure 63 and 64 shows the execution of SVM models without and with PCA respectively. The difference in the weighted average score and accuracy attained with and without PCA shows that the model performs well without PCA. SVD works well for reduction when the dataset is sparse and contains zeros but that is not the case with the dataset of this project. Hence, the dimensionality reduction with PCA and SVD were not considered ideal for this project which resulted in the usage of the initially opted method.

Figure 63

Results Attained Without PCA

```
SVM Macro weighted F1-score: 95.66%
SVM Macro weighted F1-score: 82.40%
```

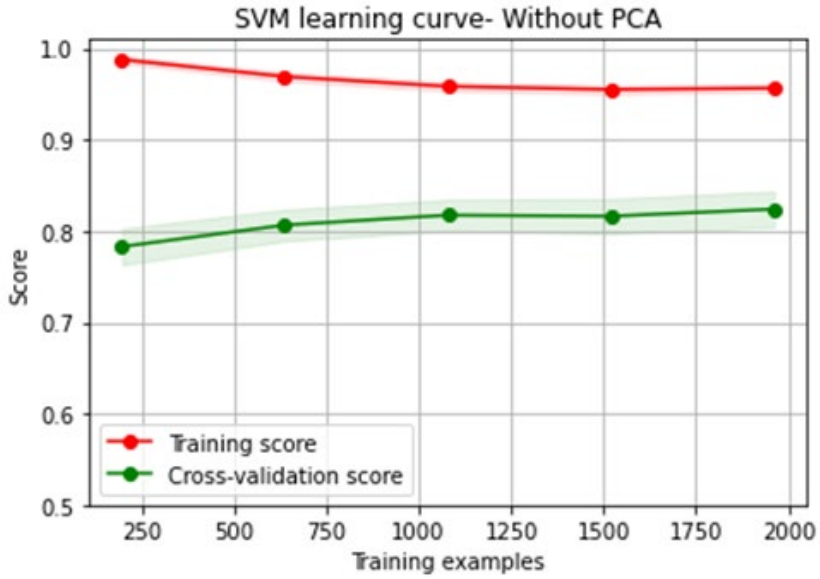
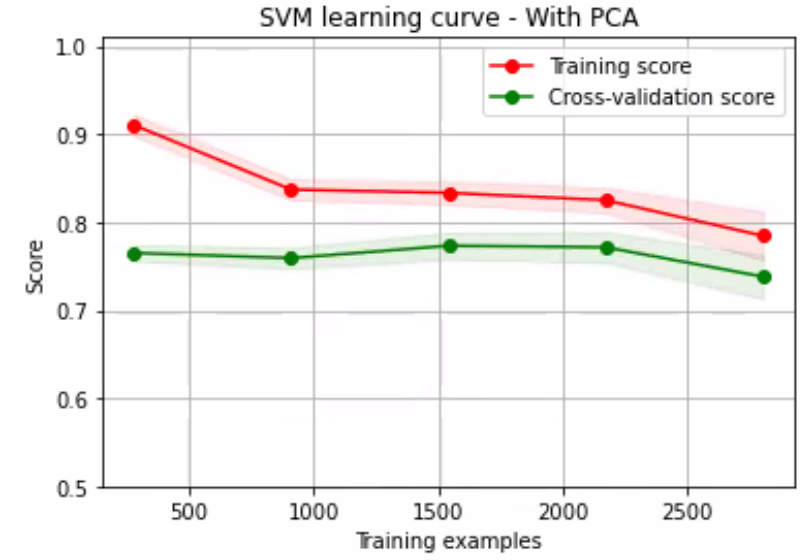


Figure 64

Results Attained With PCA

```
SVM Macro weighted F1-score: 78.46%
SVM Macro weighted F1-score: 73.81%
```



Data Preparation

As discussed in the previous sections, the essential pre-processing steps were performed to ensure that the raw dataset which contained inaccuracies was rectified before passing it on to the modeling phase. In order for the accuracy of the prediction to be precise, the modeling phase requires an optimum test and train split of data which will be discussed later on. First and foremost, an efficient and effective model selection depends on the type of data at hand and the distribution pattern. Thus, based on the data, four models were identified as suitable to perform the classification of the travel modes. The possibility of other suitable models is also viable, but the model selection is made considering the system compatibility. The models selected were Support Vector Machine (SVM), Random Forest (RF), Decision Tree (DT), and Long short-term memory (LSTM). Back to the test and train split, the optimal ratio chosen was 70:30, which can be justified by considering the learning process. Setting aside a higher percentage of records for the training phase ensures the model is well trained and the accuracy of the test could reflect the result. Also, stratify was used while performing the test and train split, which ensures that both subsets are proportional when considering the class label criteria. Supposedly if there exists any imbalance or bias, that can be tackled with the addition of stratify.

The data range for the selected features varies sometimes; thus, the model's prediction can lead to a bias towards the one with higher magnitude values. Hence, a feature scaling using StandardScaler by scikit-learn was utilized to scale the train and test data, ensuring all the features had a mean of one and a standard deviation of zero. This was discussed in detail in the standardization section. It is also ideal to create a model that avoids overfitting and can generalize better. A stratified K-fold cross-validation approach was implemented, focusing on this idea. This ensured that it generated the test sets to include the same distribution of classes. In

addition, it was configured to shuffle the samples of each class before splitting them into batches by using the shuffle parameter as true. The total number of times the batch is run was decided through the K-fold, and it identifies the optimum number of times the validation will be performed with a different sample set. The resultant prediction accuracy of each of the batches ran together as the final outcome.

Furthermore, evaluating the model's performance using an appropriate metric is necessary. The metric used here was the F1 score or F-measure, which helps evaluate the prediction performance. As per the data and the project's objective, the model is related to multi-class classification, and thus an averaging method for F1 score calculation is implemented. A weighted average F1 score was calculated by considering the mean of all the individual F1 scores of each class. A weight or support proportion was also considered during this calculation, which is the number of class occurrences in the dataset.

Figure 65 and 66 represent the first five records of the training feature set and the labels after the split was performed. Similarly, figure 67 and 68 depict the first five records of the feature set and target labels in the testing set after the split.

Figure 65

Sample Records from X Train

	Max Speed(m/s)	Non 0 Mean Speed(m/s)	Speed Std	Max Acceleration(m/s^2)	Acceleration Std	Non 0 Mean Acceleration(m/s^2)
4870	1.97	1.16	0.37	0.02	0.01	0.01
1536	1.65	1.17	0.41	0.02	0.01	0.01
162	20.21	13.38	4.38	0.19	0.04	0.04
3026	3.26	1.76	1.24	0.04	0.02	0.02
886	8.49	3.25	2.17	0.13	0.04	0.05

Figure 66

Sample Records from X Test

```
4870     walk
1536     walk
162     train
3026     bike
886       bus
```

Figure 67

Sample Records from Y Train

	Max Speed(m/s)	Non 0 Mean Speed(m/s)	Speed Std	Max Acceleration(m/s^2)	Acceleration Std	Non 0 Mean Acceleration(m/s^2)
849	13.64	4.35	3.42	0.16	0.04	0.05
3915	1.81	0.92	0.67	0.03	0.01	0.02
3139	4.75	1.67	1.77	0.05	0.02	0.02
980	4.63	1.50	0.92	0.06	0.02	0.01
293	2.01	1.09	0.48	0.03	0.01	0.01

Figure 68

Sample Records from Y Test

```
849       bus
3915     walk
3139     walk
980      walk
293      walk
```

Data Statistics

This section explains the data statistics in detail and shows how the total number of records changed while various steps were performed. The chosen dataset contained the trajectory details and the transportation labels in .plt and .txt file formats, respectively, for 73 users.

However, the drawback was that handling this large volume of data was not possible with the configuration of the system available. Also, the model's performance was primarily impacted due to this compatibility issue. The system used for this project was a regular student machine, unlike in real-time scenarios where the availability of more powerful engines helps tackle the compatibility issue. Thus, keeping this in mind, only the trajectory details for seven users were selected for the subsequent steps. The selected seven users account for more than 2.8 million trajectories. After selecting the raw data, it is necessary to perform the pre-processing steps to eliminate inconsistencies and redundancies.

As the initial step, the chosen .plt and .txt files were merged into a single file in .csv format. The size of the trajectory file was approximately 273 MB, which the system easily handled. Elimination of redundancy was done following the merge, and the total number of records was reduced from 2.81+ million to 2.77+ million. The next step was eliminating the inconsistencies, which further reduced the data from 2.77+ million to 2.44+ million. Some of the inconsistencies handled in this step were eliminating the abnormal values in the altitude, latitude, and longitude columns. During the derived feature extraction, the total trajectories were grouped so that the trips with pit stops were combined into one with the time gap between the trips recorded.

This resulted in further elimination, and the resultant set contained 144k+ records. These records were further reduced using statistical calculation, and the resultant set was then split to train and test sets using the 70:30 ratio. The extensive decomposition resulted in a further reduction of the records, which is expected because the characteristics of the travel modes can only be so much. The data at hand sufficed the requirements for the model to perform exceptionally. Figure 69 depicts the detailed statistics.

Figure 69

Data Statistics

Stage	Procedure	Source type	Number of Files	Statistics (Row x Columns)	Description
Raw Data	Extraction	.plt	7	2819105 x 8	The source trajectories were for 7 users chosen of out 73 recorded as .plt files with the respective transportation mode recorded in .txt file. The row count represents the total records of all 7 users
	Extraction	.txt	7	2819105 x 8	
Data Pre-Processing	Format Conversion and Merging	.csv	1	2819105 x 8	The .plt and .txt files of various users were merged into a single. csv file
	Elimination of Redundancies	DataFrame	NA	2778059 x 8	In this step, the redundancies in the recorded trajectories were elimitated
	Elmination of Inconsistencies	DataFrame	NA	2441281 x 9	In this step, the inconsitencies with the lat, long, alt, start and end time were eliminated
Data Transformation	Derived Feature Extraction	DataFrame	NA	148489 x 15	Additional quantifiable features were extracted from the original features
	Data Aggregation (Grouping)	DataFrame	NA	5005 x 16	In this step, the trajectories were grouped based on the travel count to determine the general characteristics of travel modes under various parameters. Since the travel modes were less than 15, the grouping ended with 5005 records with various tendencies that seem to appear recurring among the trajectories. The grouping may seem to end with a lesser record count, but considering the project necessity, this suffices the requirement as we approach it from the perspective of the various travel modes.
	Data Standardization	DataFrame	NA	5005 x 16	The data was then standardized to fit within a range so that the valid outliers doesn't need to be eliminated
Data Preparation	Training Set - Source	DataFrame	NA	3504 x 15	The record count displayed for each of the dataframe here represents how many records the testing and training set ends up with while splitting using the ratio mentioned.
	Training Set - Target	DataFrame	NA	3504 x 1	
	Testing Set - Source	DataFrame	NA	1501 x 15	
	Testing Set - Target	DataFrame	NA	1501 x 1	
	Split Ratio	NA	NA	70:30	

Data Analytic Results

Data analytics is essential to understand not only the features by itself but also how they relate to one another. For instance, speed can help determine the time taken for travel, and data analytics can help understand to what extent these features are related and how they could be the driving force behind the predictions. Hence, in this phase, the data is further examined to understand the features' correlation better. In order to achieve this, different data analytics visualization approaches, such as KDE plot, pair plot, scatter plot, and heat map were implemented. Figure 70 depicts the KDE plot of four quantifiable features max speed, 75% speed, 97% speed, and mean speed which was done to understand the population density.

Figure 70

KDE Plot – Part 1

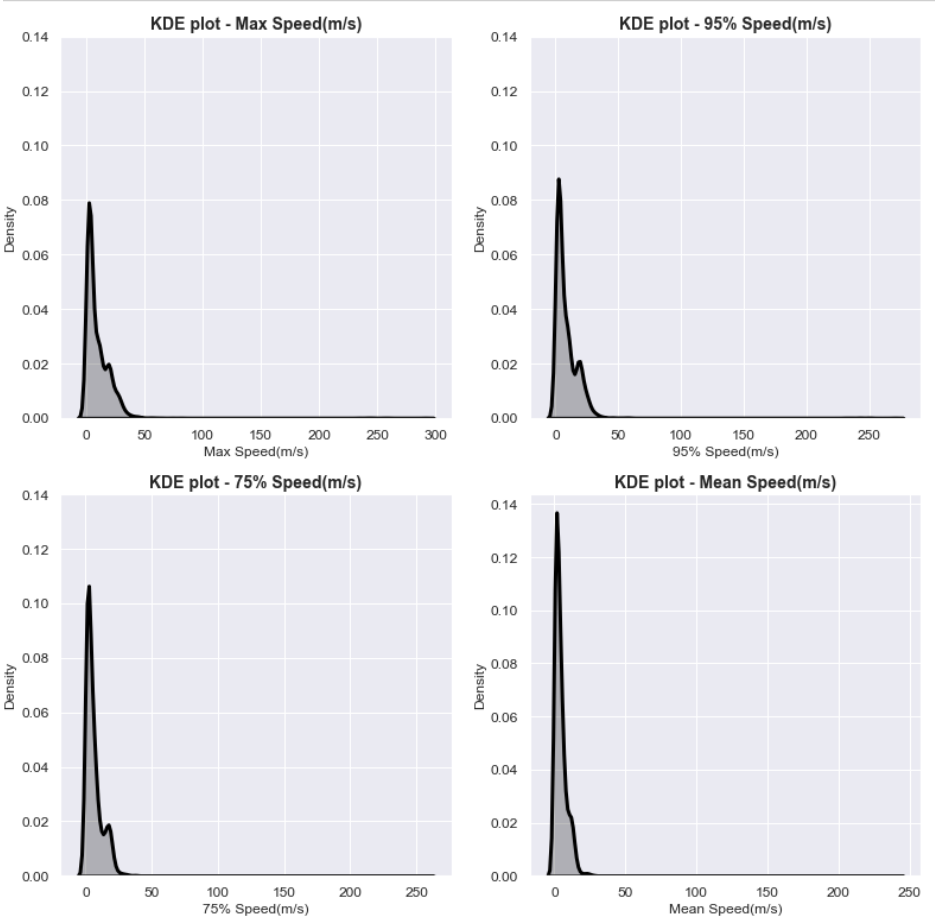


Figure 71 represents the KDE plots for quantifiable features related to acceleration, such as max acceleration, 95% and 75% acceleration, and mean acceleration. This helps gain insight into how the population density is distributed for the acceleration features. Similarly, the density distribution of the other four features, which were the standard deviation of speed and acceleration, total time in seconds, and total distance in meters, were also depicted in figure 72. Figure 73 portrays the scatter plot of max speed versus max acceleration and total distance against total time.

Figure 71

KDE Plot – Part 2

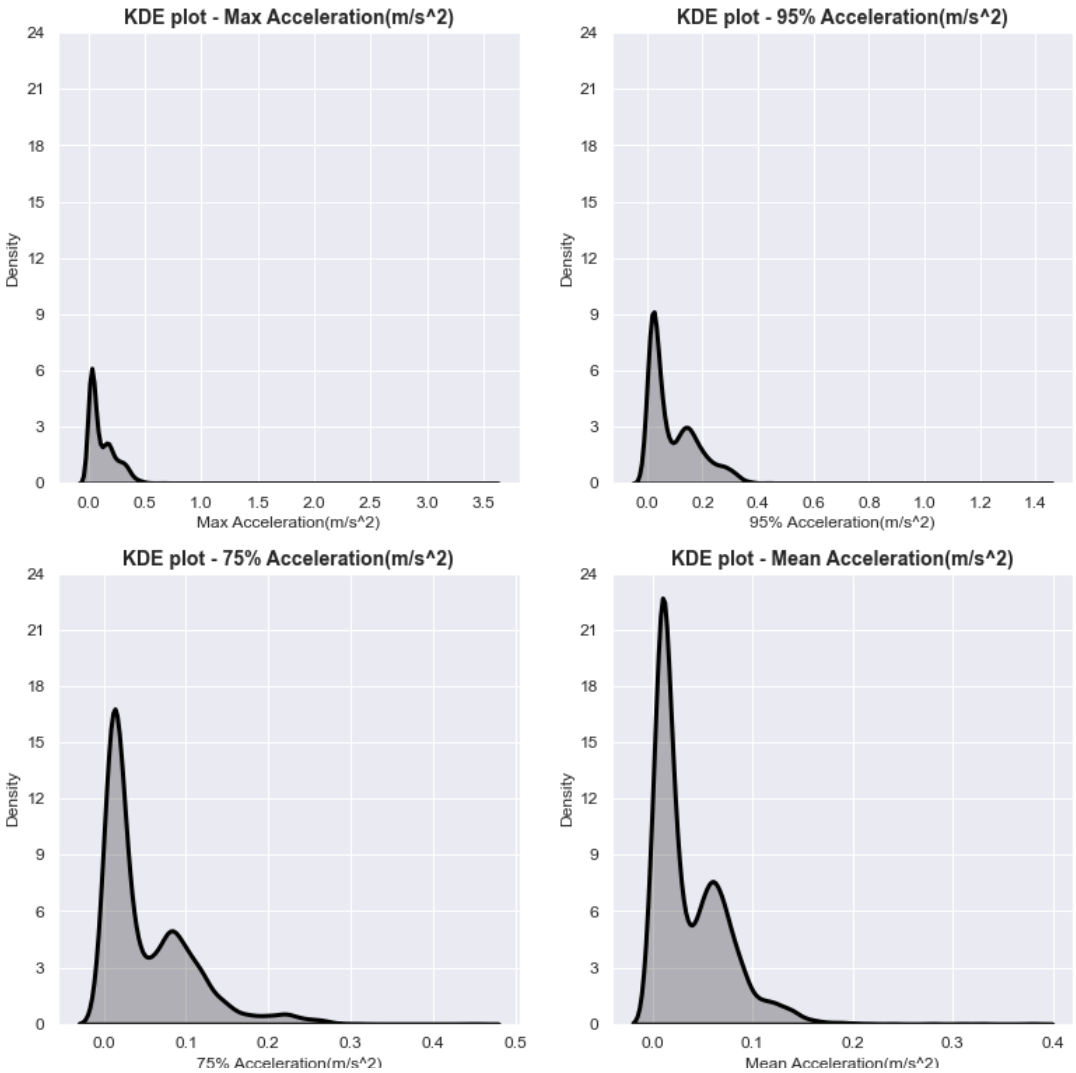


Figure 72

KDE Plot – Part 3

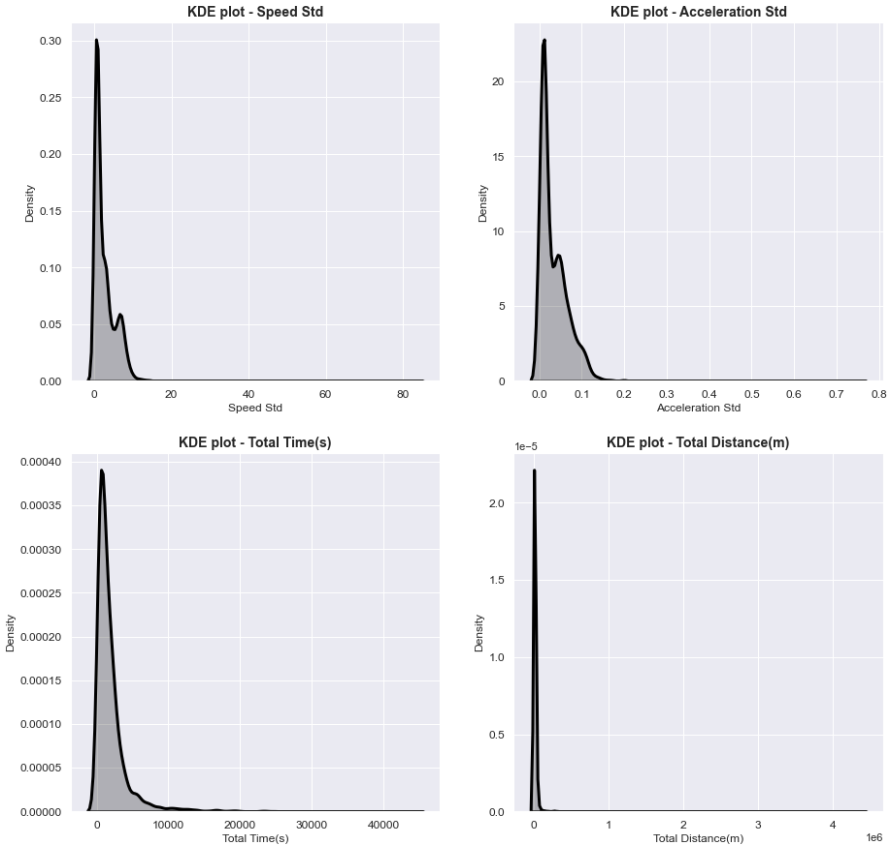


Figure 73

Scatter Plot

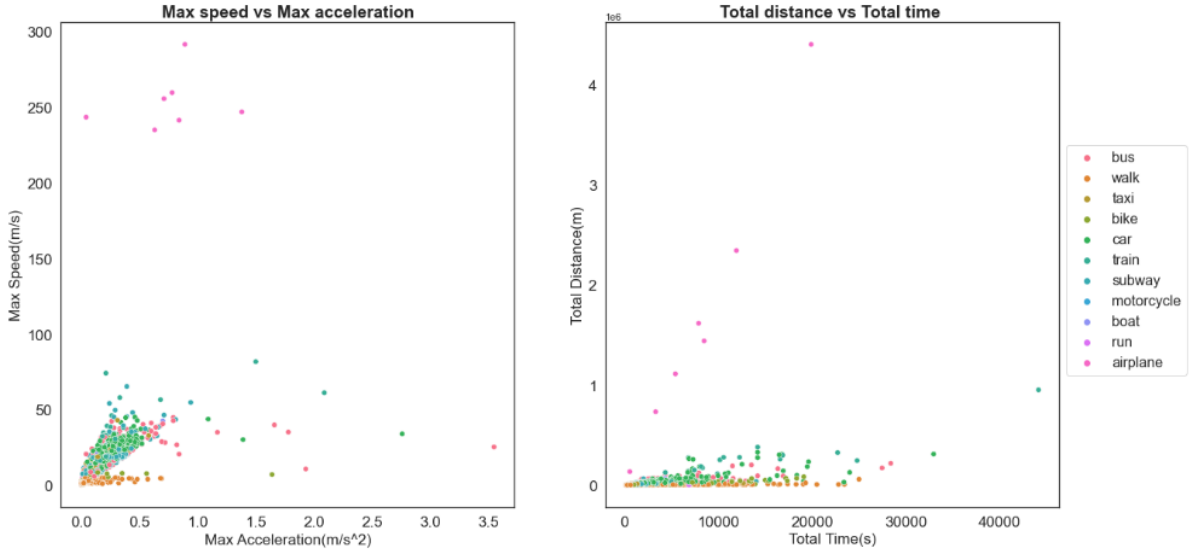


Figure 71 shows the distribution of data based on various transportation modes. While observing the distribution of the dots in the plot, a specific pattern stands out. There appears to be a proportional relationship between maximum speed and acceleration, similarly between the total time and distance. The various colored dots distinguish the different travel modes, and when observing closely, the ones that hang appear to be outliers belonging to the travel mode airplane. This is understandable because of the sheer difference between the range of airplanes and the other transportation modes. Another vital thing to notice is that both features are directly proportional. Similarly, the comparison between time and distance was also correlated. The outliers in this plot relate to the travel mode being an airplane. The time taken to cover a distance by walking is lesser than by bus or other travel modes. This is explicitly visible when the various colored dots are compared side to side. To examine this further, a pair plot was used, which will be explained briefly in the forthcoming section.

Checking correlation

Relationships between the features can help determine the driving forces influencing specific behavioral patterns, and analyzing such relationships can help determine how it helps enhance the predictions. However, a relationship that looks interoperable can sometimes be different from how it should be perceived. The causation factor also must be taken into consideration. For instance, feature A might look like it influences the value of feature B, but the causation factor might imply otherwise. There could be potential feature C which is the bedrock influencing aspect but masked due to feature B's existence. Developing a correlation scheme will aid in identifying the actual relations while also considering the causation. Hence it is necessary to consider the Correlation Matrix as a part of data engineering. Recognizing the flaws with the

dataset relationship can be crucial while developing a model that will be hosted in real-life scenarios.

Therefore, keeping the points mentioned above in mind, an intensive process was designed to identify the correlation between the features while detecting the travel modes. Pandas profiling has been used to determine various correlations, such as Spearman's, Pearson's, Kendall's, and Phik. Not stopping with the correlations, the interactions between several fields were also measured interactively using pandas profiling so that it is possible to cross reference all the possible features. The result of this interaction was also visualized so that the results were easily readable. The Correlation Matrix and visualization of the interactions will be helpful in future steps. Some of the critical insights attained from this process are that most of the generated statistical features have a high correlation, acceleration is proportional to speed and vice versa, and time appears to be the only feature with the least correlations. Figures 74 and 75 below depict Correlation Matrices.

Figure 74

Spearman's, Pearson's Correlation Matrices

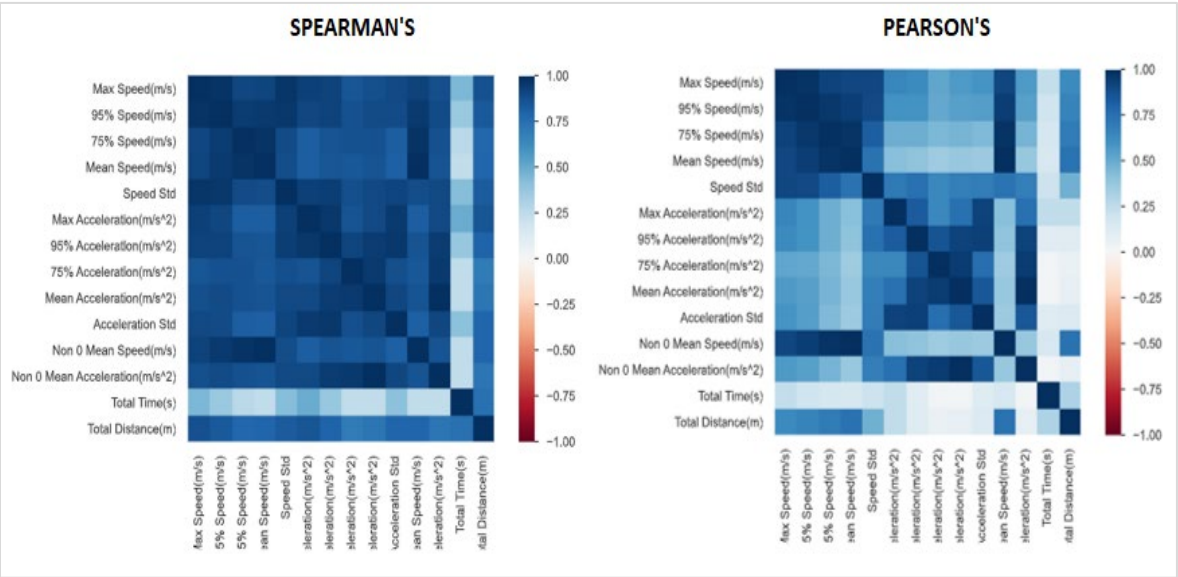
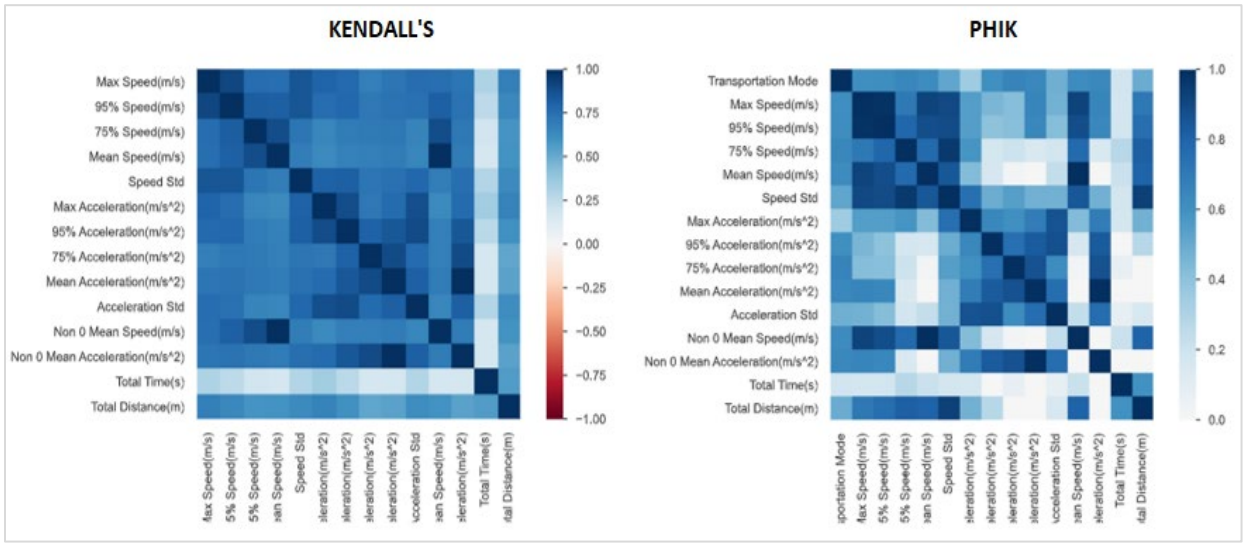


Figure 75

Kendall's and Phik Correlation Matrices

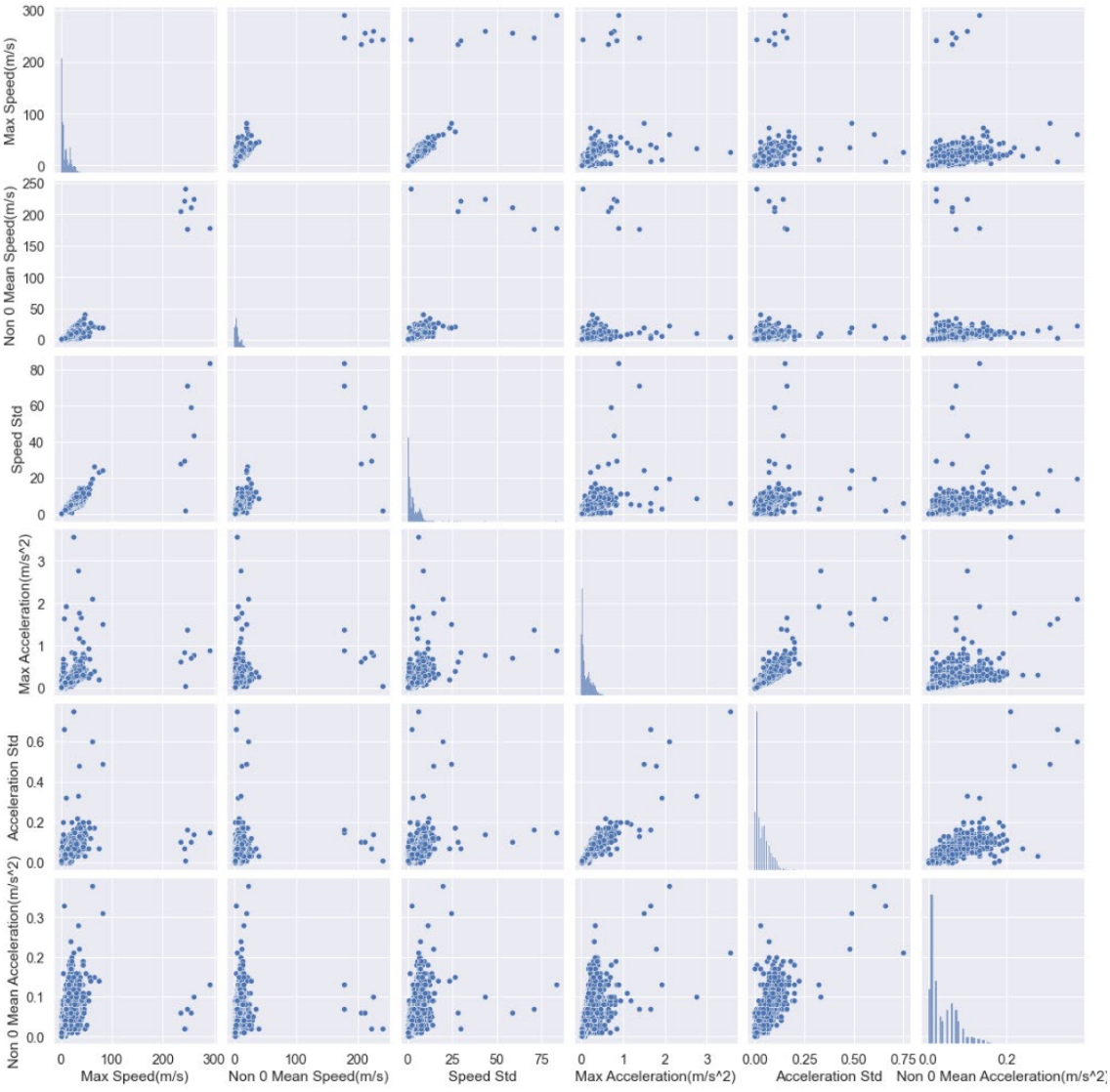


As depicted in Figures 74 and 75, it is evident that the correlation matrix can help determine the relationship between different fields, and similar correlations are attained from various methods.

A pair plot was also implemented to understand the correlation between the features, shown in figure 76. This demonstrates the relationship between six features: Max Speed, Non 0 Mean Speed, Speed Std, Max Acceleration, Acceleration Std, and Non 0 Mean Acceleration. There were a few noticeable recurring patterns among the correlation of these features: The values were densely populated on the lower ranges; most features were directly proportional to the compared correlated feature. The sparsity of the data points in the high ranges can be attributed to the presence of travel modes which can stand out as an outlier due to sheer variance in the performance. The plots placed diagonally were represented as a histogram. This is because they were correlated against each other, which is impossible due to the likelihood of it not being able to influence itself. Hence, a histogram representing the population density's concentration across the board was visualized.

Figure 76

Pair Plot



Understanding these correlations using the various visualized plot played a crucial role in deciding how the data could be utilized so that the predictions achieved using them could be enhanced to the maximum possible extent. Examining the visualization helped to understand the presence of valid acceptable outliers and, in turn, helped decide how to handle that. The future process involving modeling and evaluation appears to be optimistic following the results of various phases discussed in this chapter.

Model Development

Model Proposal

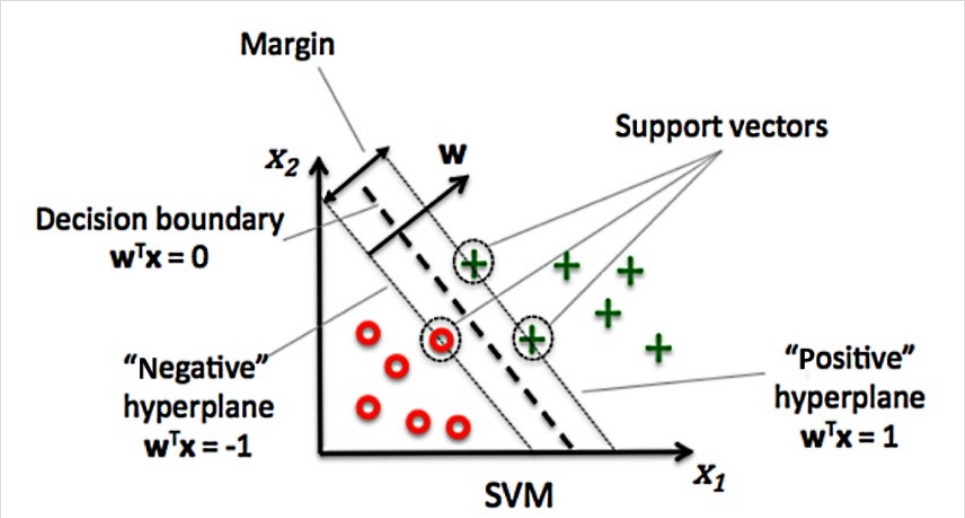
Following the data preparation and splitting into training and test sets, the data was passed on to the modeling phase. This section covers the in-depth background research process performed to assist in the selection of models. The section also covers the chosen algorithms' working methodology and the optimization performed.

Support Vector Machine

The first model identified as a potential for execution is the Support Vector Machines (SVM) which constitutes both classification and regression. SVM employs both linear and non-linear enactments and is particularly beneficial when dealing with data that contains extremely complicated datasets. When dealing with margin separations, SVM typically performs well and is less prone to overfitting. Numerous papers already published pertinent to the topic at hand attest that SVM is one of the most popular algorithms for classification problems. SVM uses a hyperplane to assist in generalizing the distinction between the classes. A line that separates and establishes the boundary between the classes is known as a hyperplane. If the data is linearly separable, SVM works by finding a maximum margin and adding a penalty each time a point crosses the margin line. These margins can otherwise be called Soft Margins, and the commonly used penalty type is Hinge loss. Hence, a hinge loss is directly proportional to the number of violations. Suppose the data is not linearly separable; then the SVM Makes use of a kernel. Assuming the data is linearly non-separable, the SVM employs kernel methods to make it separable potentially. Figure 78, retrieved from Ahmad (n.d.) (Section 2, Traditional Supervised Learning Algorithms, Understanding classification algorithms, The SVM Algorithm), depicts the fundamental components of SVM algorithmic structure.

Figure 78

Mechanism of Support Vector Machine



Detailed background research on existing works helped understand how SVM could be employed in trajectory detection. The section below will explain the papers explored as a part of background research. The working methodology of the SVM algorithm will be discussed in detail in the forthcoming sections.

Technology and Solution Survey. The exponential growth of data collected by GPS Devices acted as a gateway that steered the interest of researchers and academics to determine how this data can be put to good use. That curiosity led to various research and solutions. These previously existing works can aid while looking for ways to approach a similar issue. For the devised model's performance to be at its utmost potential, it was essential to review the existing similar works and analyze the pros and cons. Knowledge gained from this extensive technological and literature survey can help be wary of the issues and roadblocks that could arise down the line. As a precaution, it can also help devise solutions to tackle the occurrence of such cases. Hence, it was necessary to thoroughly analyze some of the existing works to comprehend the current methodologies and limitations.

Jahangiri and Hesham (2015) proposed an approach to develop a multilevel classifier that could aid in identifying the transportation modes such as running, walking, and driving. The devised approach included the usage of K-Nearest Neighbor, Support Vector Machine(SV), Decision Tree, and Random Forest Methodologies. The tree-based classifiers were equipped with additional ensemble methods, such as bagging, to avoid overfitting. The data used were recorded from smartphone sensors, including a list of sensors such as a gyroscope, rotation vector, and accelerometer. The optimal model selection and validations were carried out using K-fold validation. The proposed approach also included extracting features identified following a certain ethical logic, and the feature's relevancy was prioritized during extraction (p. 1).

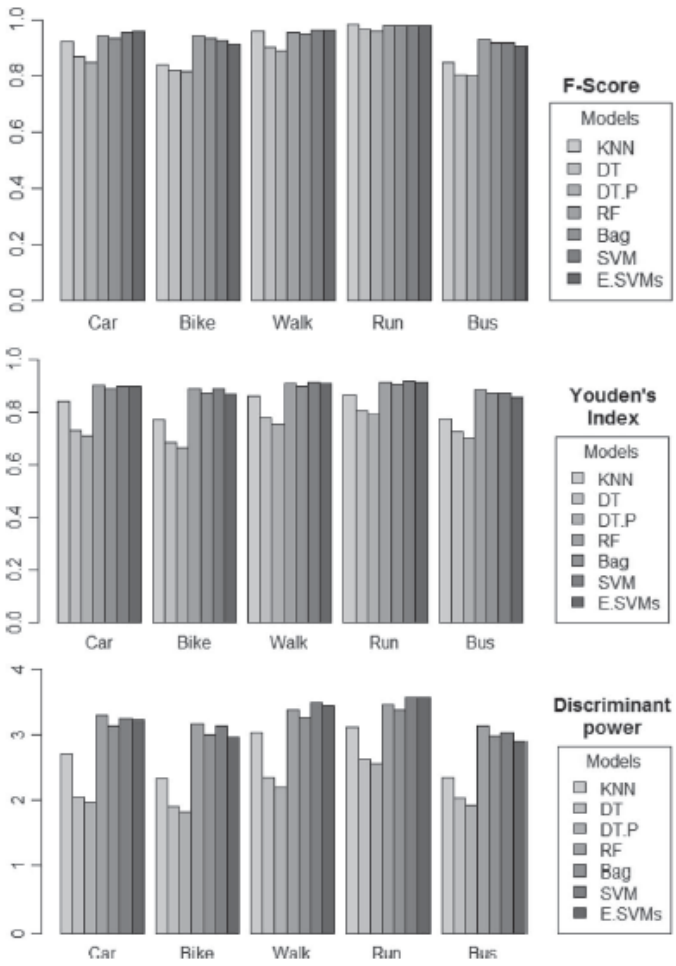
The data collection phase involved developing a smartphone application that stored data from all the smartphone sensors such as GPS, Gyroscope, Rotation Vector, and Accelerometer. Employees were instructed to make multiple trips carrying the smartphone with the application installed. They were also instructed to carry the mobile in different locations such as backpack, hand, and wallet to eliminate the preconceived notions from sensor positioning. Additional to the self-explanatory features, other features that can be more statistical and absolute were also prioritized. Time window monitoring helped describe the variables that can emphasize the time dependency of the features. Spectral entropy was an addition that helped measure the highs and lows of the distribution and attain insights. A high spectral entropy implied that the distribution was flat and vice versa (pp. 3-4).

The model development phase of this study incorporated SVM, Decision tree, Random Forest, Bagging, and K-Nearest Neighbours for comparison. K-fold Cross-Validation helped during the model selection process, and the data was normalized using scaling ranging from a negative one to one. ANOVA test, correlation-based feature selection, Chi-Squared, and

Information Gain methods helped decide on the most relevant features to be selected. As a result, 80 features were selected to be the most relevant and were passed on to the modeling phase. The performance of the models was measured using a five-fold Cross-Validation operation on various chosen models (pp. 4,6-7). The Model's performance was evaluated using four metrics: the F-Score, Younden's Index, Discriminant Power, and Overall Accuracy. The evaluation results illustrated that the Random Forest and SVM models recorded the best performance. Figure 79, retrieved from Jahangiri and Hesham (2015), depicts models' comparison through various performance measures (pp. 1,4,6-7,10-11).

Figure 79

Model Comparison Results



When looking into the misclassification rates, a key factor stands out. Some of the limitations of the work can be attributed to the misclassification rates. As expected, the cars and bus modes were the most difficult to distinguish, and the bus mode was misclassified around 4 to 6 % of the time The proposed study has various possibilities for future directions that can be applied to various research problems. Some enhancements can include adding mode data, devising approaches to examine the sequence, adding mode transportation modes, and availing real-time data usage. Conducting error analysis can help understand where the model failed to classify at a staggering rate (p. 11).

Asgari and Clemencon (2018) proposed an approach to detect travel modes, which, unlike most existing works, concentrated on combining fine-grated data (GPS) along with coarse-grained data(GSM). Most of the existing works at that time were concentrated only on fine-grained data, and this study had a unique approach. The reasoning behind this exciting approach was to combat the prominent battery issues while using GPS Data, which in turn steered the project toward a more expensive side. Another critical point was that this helped cover more ground since GPS could not cover underground trajectories. TDM algorithms were investigated during this study through both segment-based and sequence-based machine learning approaches (p. 1).

The data used for the study was collected from the app Geo4Cast, which was designed to filter the stationary records and record only the trajectories that detected motion patterns. The application was designed to work at a lower frequency while collecting data, which helped combat the battery issue. The sampling rate of the collection process was 2 minutes, but there was a possibility of not abiding by it due to the prevalence of underground movements. A total of 5 transportation modes were recorded from 8 individuals and consisted of more than 100 hours

of multi-modal trajectories (p. 4).

Both sequence-based and segment-based machine learning algorithms were tested. Sequence-based included Conditional Random Field (CRF), and segment-based included SVM, Decision Tree, and Random Forest. The methodologies were trained to classify the recorded transportation modes, and Cross-Validation was implemented as an evaluation method with the total job count assigned to 100. The experiment's results showed that the classifiers' accuracy improved after adding the extra features, and the highest accuracy among the segment based was recorded by SVM. When looking into the sequence-based results, the generalized accuracy of 83% dropped to 71% when it was implicated to validate that individual trajectory were classified correctly. This can be rectified by utilizing a more extensive training set. Figure 80, retrieved from Asgari and Clemencon (2018), depicts the results of various machine learning algorithms with represented features (p. 5-6).

The limitation of this work was that separated GPS and Cellular data were unavailable, making it impossible to compare the results with other cases involving fine-grained or coarse-grained data. The project can be extended by embedding both algorithms and implementing them after increasing the training size and adding more diverse transportation modes. Synthetic machine learning approaches could also be explored as a future enhancement (p. 7).

Figure 80

Model Comparison Results

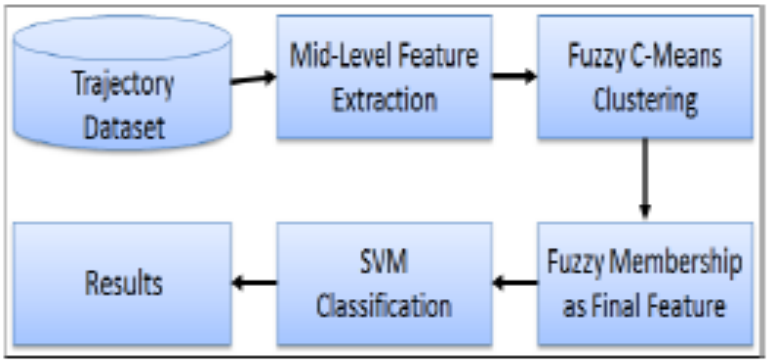
	Features								Classifiers			
	speed	Acc	len	dur	bus-c	metro-c	train-c	dev	SVM(1Vs1)	SVM(1VsR)	DT	RF
1	√								68.9%	68.9%	64%	58.2%
2	√	√							70.7%	70.5 %	68.3%	64.5%
3	√	√	√						71.6%	71.2%	69%	67.9%
4	√	√	√	√					71.2%	70.9%	69.5 %	68.3%
5	√	√	√	√	√	√	√		76.8%	76.5%	71.4%	75%
6	√	√	√	√	√	√	√	√	77.7%	78.8%	72 %	74.5%

Saini et al.(2017) proposed an approach for trajectory classification but mainly concentrated on the security and surveillance standpoint. The development of smart cities has increased the necessity for a remarkable surveillance system. The study concentrated mainly on identifying ways to eliminate and filter out the abnormalities in traffic data and adequately classify the flow of movement. The proposed approach used a Fuzzy C-Means (FCM) clustering technique accompanied by SVM to classify object trajectories carefully. The features identified by FCM were then passed on to SVM for classification. (p. 1-2).

The methodology proposed consisted of extracting essential features on top of which FCM clustering was applied. Some extracted features were position coordinates, angular features, and velocity from the video trajectories. The resultant fuzzy membership will then be classified using an SVM classifier. Figure 81, retrieved from Saini et al. (2017), depicts an overview of the proposed model (p. 1-2).

Figure 81

Overview of the Model

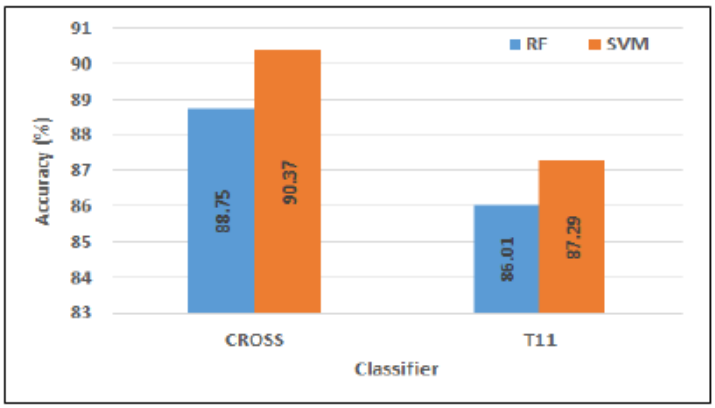


The devised approach was tested with two public datasets, CROSS, and T11. The accuracies recorded by Random Forest and SVM models are depicted in figure 82, retrieved from Saini et al. (2017). Limitations of this work include the availability of surveillance footage and extraction of robust trajectories. Future enhancements to this work could be the inclusion of

combining approaches such as decision fusion of multiple classifiers to boost the classification process (p. 4).

Figure 82

Model Performance Comparison



Asad et al. (2021) devised an approach to analyze mobility and develop a prediction model to support smart transportation, especially with the birth of Coronavirus (COVID-19). The pandemic had an adverse effect on all sectors of business, including the transportation sector. The proposed approach paid careful consideration to various factors such as data selection and processing, integration, and normalization. Multilevel classification algorithms were prioritized to predict transportation modes, including driving, walking, running, and other diverse modes. The study also delved into analyzing the regional mobility of various continents. Six machine learning algorithms were chosen for this study (p. 1).

The data collection process was carried out in six regions encapsulating different transportation modes. The original dataset contained regional information in a detailed manner, along with trajectory records, which made formulating it into a sequential format tiring. This can be attributed to the sheer size of the dataset. Hence, to avoid that aggregation opted which assist random fluctuations that exploited the dataset's prominent trends. The data was then normalized

to suit the requirement. Normalization assisted in enhancing the prediction of mobility trends by utilizing the arrival dates. The data points normalization was achieved by comparing travel times from the start of COVID-19 regions against the free flow periods (p. 2).

The processed data was then passed along to various machine-learning algorithms, which analyses the evaluation metrics used to predict the trends in moving patterns based on various transportation modes and regions. Various scikit − learn libraries were used to evaluate the performance of the algorithms. Some evaluation methods used were accuracy, recall, precision, and F-measure. The results illustrate that SVM had the highest accuracy at 91.2%, followed by Random Forest at 84.5% for the various modes and regions. Figures 83 and 84, retrieved from Asad et al. (2021), depict the results of various models while analyzing regional and worldwide data. The limitation surrounding this work would be the requirement of reasonable modifications to the existing transportation system, which could not be cost-effective. The future scope of the project could include exploring how this data can be used in other research areas, such as targeted marketing and city planning. (pp. 1, 4, 6).

Figure 83

Worldwide Model Performance Comparison

MOBILITY PREDICTION CLASSIFICATION FOR DIFFERENT TRANSPORTATION MODES ACROSS THE WORLD

Machine Learning Classifier	Accuracy	Precision	Recall	F-Measure
Support Vector Machine (SVM)	91.25	0.91	0.90	0.90
Naive Bayes (NB)	86.25	0.86	0.85	0.86
Multi-Layer Perceptron (MLP)	83.0	0.83	0.82	0.83
Logistic Regression (LR)	81.9	0.81	0.80	0.81
K-Nearest Neighbour (KNN)	81.9	0.81	0.80	0.81
Random Forest (RT)	86.0	0.86	0.85	0.86

Figure 84

Regional Model Performance Comparison

MOBILITY PREDICTION CLASSIFICATION FOR DIFFERENT TRANSPORTATION REGIONS

Machine Learning Classifier	Accuracy	Precision	Recall	F-Measure
Support Vector Machine (SVM)	84.23	0.84	0.82	0.82
Narve Bayes (NB)	82.51	0.82	0.81	0.82
Multi-Layer Perceptron (MLP)	81.56	0.81	0.81	0.81
Logistic Regression (LR)	80.36	0.80	0.79	0.80
K-Nearest Neighbour (KNN)	79.57	0.79	0.78	0.79
Random Forest (RT)	84.5	0.84	0.83	0.84

Raktrakulthum and Netramai (2017) proposed a way to classify vehicles, especially in congested traffic situations. The congested traffic can lead to complexity in predictions due to the different vehicles packed together and moving at a low speed. In this study, an approach to generate a three-dimensional point cloud on top of the acquired images was built. The system was developed keeping the cost in mind, and the developed three-dimensional point cloud was used in the modeling phase to deliver predictions. The approach used K-Nearest Neighbors and SVM with hyperparameter tuning to classify whether the vehicle is a car or a bus (p. 1).

The images processed to extract the three-dimensional point cloud were extracted from a video file from a left and right camera. Selected regions of interest were identified and picked for subsequent steps. The data was then processed using feature extraction, segmenting, filtering, and tracking techniques. A disparity map was calculated using the Semi Global Block Matching algorithm. The results were then passed on to Weight Least Squares which produced the final point cloud. The extracted features were put to use in K-Nearest Neighbors and SVM algorithms. The

training and testing data was split to avoid bias and was grouped based on the estimated average speed of the vehicle. The result demonstrated how well the prediction worked in highly congested traffic situations. The predictions attained by both the K-Nearest Neighbors and SVM algorithms were exponentially high at 95.8% compared to the other models. When looking into the motorcycle, SVM performed better than the comparative models (pp. 1,4).

The results made it evident that even during various levels of congestion, the SVM and K-Nearest Models were performing at exceptional levels compared to the others. Figure 85, retrieved from Raktrakulthum and Netramai (2017), depicts the accuracy attained by various models for classifying the bus and car modes of transportation. The limitations of the work can be attributed to the clarity and quality of extracted images which better equipment can enhance, with the downside being cost requirement. Future enhancement of the proposed study can include additional travel modes such as buses and trucks. The study can be further improved by attempting to lower the FDR while improving the three-dimensional point cloud calculations (pp. 5-6).

Figure 85

Model Performance Comparison on Various Traffic Situations

		low congestion		moderate congestion		high congestion	
		TPR (%)	FDR (%)	TPR (%)	FDR (%)	TPR (%)	FDR (%)
car	KNN	97.5	4.1	99.3	2.7	95.8	19.3
	SVM	97.5	2.5	99.3	2.0	95.8	14.8
	diff	0	1.6	0	0.7	0	4.5
motorcycle	KNN	54.8	8.9	63.5	2.4	57.2	15.5
	SVM	74.2	5.5	85.7	11.5	82.2	13.2
	diff	19.4	3.4	22.2	9.1	25.0	2.3

Similarities between the Existing Works. Looking into the various existing studies laid a foundation for the necessary understanding to step toward the execution of the project. The research papers helped attain insights into the various phases of a project cycle and the complications attached to it. The research papers played a significant role in the decision-making involved in data source selection, algorithms, methodologies, and architectures related to the scenarios. In addition to the insight attained, resemblances were also observed between these works, which helped identify the hurdles that could occur and the precautious measure that can help avoid them. Some of such observed parallels will be discussed in this section. Firstly, there were a few similarities in the selection of data sources. The approaches proposed by Su et al. (2016), Zhu et al. (2021), and Asad et al. (2021) used data collected solely by GPS as a data source. Likewise, the research papers published by Raktrakulthum and Netramai (2017) and Saini et al.(2017) used video records as a source with enabled GPS tracking. Asgari and Clemencon (2018) and Jahangiri and Hesham (2015) proposed travel mode detection studies that utilized GPS data and other phone sensors. The second similarity observed was between Jahangiri and Hesham (2015) and Asgari and Clemencon (2018), which used a Cross-Validation approach with K-fold. The third similarity observed was regarding the types of transportation modes present in the chosen data source since all the papers covered basic transportation modes. The fourth observed similarity was the presence of the Support Vector Machine model as one of the chosen ones.

Contrasts between the Existing Works. During the tenure of this extensive background research, a few contrasting opinions and methodologies used by these papers were also observed. These discrepancies showcase how a problem can be approached from various angles. The contrasting opinions were primarily associated with crucial factors, such as data source selection, methodologies choice, and algorithms used. Interestingly, in some cases, a research paper's

limitations were combated by an alternative approach suggested in another paper. Some such observed contrasting opinions are discussed here. Firstly, the discrepancy observed was on the project objective. Asgari and Clemencon (2018), Zhu et al. (2021), Saini et al.(2017), and Jahangiri and Hesham (2015) primarily focused on travel mode detection without any background implications. In contrast, Raktrakulthum and Netramai (2017) focused on travel mode detection, especially in congested traffic, and Asad et al. (2021) concentrated on mobility trend analysis assisted by travel mode detection. The second discrepancy observed was the choice of data source, a few preferred pure GPS data, whereas others preferred video records to analyze and extra smartphone sensor data. It can be explained in detail as the studies by Su et al. (2016), Zhu et al. (2021), and Asad et al. (2021) used data collected solely from GPS. In contrast, Raktrakulthum and Netramai (2017) and Saini et al.(2017) preferred utilizing video records, and Asgari and Clemencon (2018) and Jahangiri and Hesham (2015) utilized smartphone sensor data. The third observed discrepancy was in the evaluation methodology used, Su et al. (2016) and Saini et al.(2017) preferred accuracy. In contrast, Jahangiri and Hesham (2015) utilized four metrics: the F-Score, Younden's Index, Discriminant Power, and Overall Accuracy, and Asad et al. (2021) utilized F-Measure. The fourth contrasting opinion was related to the total number of features extracted. Some papers preferred limiting the number of features, whereas others opted to enhance the selection by increasing the total features chosen. For instance, Zhu et al. (2021) employed extensive feature extraction. These contrasts help identify, pick and choose what was necessary to suffice the goal

and what was not. The in-depth background research helped understand the various pros and cons that existed during the tenure of similar research. Identifying those helps the project stay on track and veer away from any hurdles that could be caused. The research also helped better gain

insight into the chosen Support Vector Machine algorithm. The section below will further examine the working methodology of Support Vector Machines and explore the algorithm and pseudocode.

Working Methodology of Support Vector Machine. This section explains the methodology behind the chosen Support Vector Machine (SVM) algorithm. SVM consists of both classification and regression, employing linear and non-linear enactments. They are accommodating when dealing with complex datasets. The overview of the model provided at the beginning of the model proposal section acts as an introduction to the model, whereas this section drives deeps into the framework itself. One benefit of SVM is that it does not skew the results due to the presence of outliers. Using both linear and non-linear technology will depend on the complexity of the data. SVM often has good accuracy and performs well in complex domains with distinct margins. Some advantages of SVM include its efficiency while dealing with high dimensional cases, memory efficiency due to the usage of a subset of training points known as support vector, usage of kernels for specific decisive functions, and the accessibility to define custom kernels.

Rasmus Pedersen (2006) introduced the background of the Support vector machine as one of the initial algorithms produced as a part of statistical learning theory frameworks, which was later extended to tackle binary classifications and misclassifications. It was also suggested that the SVM could be extended to other frameworks, such as regression, ranking, clustering, and novelty detection. Figure 86, retrieved from Rasmus Pedersen (2006), depicts a pseudocode for training the SVM algorithm.

The algorithm depicted in figure 86, retrieved from Rasmus Pedersen (2006), depicts the usage of sequential minimal optimization (SMO) for training. The optimizations performed were

also highlighted in the steps of the provided pseudocode's line 5. The algorithm suggests the optimization of two Lagrange multipliers simultaneously, which can be attributed to the utilization of SMO. The training also involves the SVM choosing the optimal kernel for the task, and the optimizations for training are respective to the constraints imposed. The pseudocode also suggests that SVM imposes a three-stage approach similar to other Machine learning algorithms, such as training, testing, and predictions (p. 3).

Figure 86

Pseudocode of SVM Algorithm

Algorithm 1 Training an SVM

Require: $\boldsymbol{X}$ and $\boldsymbol{y}$ loaded with training labeled data, $\boldsymbol{\alpha} \Leftarrow 0$ or $\boldsymbol{\alpha} \Leftarrow$ partially trained SVM

1: $C \Leftarrow$ some value (10 for example)
2: **repeat**
3: **for all** $\{\boldsymbol{x_i}, y_i\}, \{\boldsymbol{x_j}, y_j\}$ **do**
4: Optimize α_i and α_j
5: **end for**
6: **until** no changes in $\boldsymbol{\alpha}$ or other resource constraint criteria met

Ensure: Retain only the support vectors ($\boldsymbol{\alpha_i} > 0$)

Cortes and Vapnik (1995) proposed a paper detailing the support vector networks, which provided insights into the internal working of the support vector machine framework in an in-depth fashion. The paper explored the idea of high-dimensional feature space consisting of input vectors that were non-linear mapped (p. 1).

The construction of a linear decision in the feature space which resulted in higher generalization ability, was discussed. The general idea behind SVM was related to cases where the training data could be separated without error before this. The implementation extended the possibilities of utilizing SVM to non-separable instances as well. The proposed study also

explored the utilization of polynomial transformations with the high generalization ability of SVM (p. 1).

A support-vector network decision rule can be constructed by solving a quadratic optimization (1) retrieved from Cortes and Vapnik (1995) (p. 9) (Equation 26).

$$W(\Lambda) = \Lambda^T 1 - \frac{1}{2}\left[\Lambda^T D\Lambda + \frac{\delta^2}{C}\right] \quad (1)$$

The constraints to be satisfied while solving (1) is depicted in (2) both retrieved from Cortes and Vapnik (1995) (p. 9) (Equation 27, 29).

$$0 \leq \Lambda \leq \delta 1, \quad \Lambda^T Y = 0 \quad (2)$$

The matrix D_{ij} values can be attained by (3) retrieved from Cortes and Vapnik (1995) (p.12).

$$D_{ij} = y_i\, y_j\, K\left(x_i, x_j\right), \qquad i, j = 1, \ldots\ldots, l \quad (3)$$

The variables used in (3) retrieved from Cortes and Vapnik (1995) can be identified by the elements present in the training set, and the K (u, v) can be explained as the function used to determine the convolution of the dot-products. The study also discussed the solution to the optimization issue by solving them intermittently while getting determined by the training data, and the solution also constitutes the support vector generalization (p. 1).

Ghosh et al. (2019) discussed the possibilities of SVM utilization for linear and non-linear classifications. The research emphasized using kernel functions to generate the decision boundaries involving data that are not linearly separable. The paper also dug into the utilization of SVM in real-life scenarios along with the future aspects of expansion. The study also included an in-depth introduction to the SVM algorithm, recognizing its compatibility while dealing with regression and classification problems. During the early days, the SVM algorithm was equipped only to classify the linearly separable data with the help of hyperplanes. Later, a new way of utilizing kernel functions to implement the non-linear classifications was introduced, which

catapulted SVM into becoming one of the most dominant classification algorithms. Future studies by various researchers added the implementation of unsupervised learning to suffice the usage of data without class labels (p. 1).

SVM can be mainly depicted as a supervised learning technique formulated as (1) retrieved from Ghosh et al. (2019). The variable present can be explained as $V(y, f(x))$, implying the generic loss function which calculates the error given the value of x and predicting in place of y, $P(x, y)$ depicting the unknown probabilistic function. This demonstrates the need to identify the function to minimize error expectations as given in (4) retrieved from Ghosh et al. (2019) (p. 1) (Equation 1).

$$\int V\left(y, f(x)\right)P\ (x, y)\ dx\ dy \tag{4}$$

In regards to the binary classification, the boundaries are set as depicted in (5), retrieved from Ghosh et al. (2019), which is often different from multi-level classifiers. Classification can be explained as the process by which the algorithm utilizes the decision-making boundaries with the help of training data (p.1).

$$h_e(x) \geq 0.5 \rightarrow y = 1\ , h_e(x) < 0.5 \rightarrow y = 0 \tag{5}$$

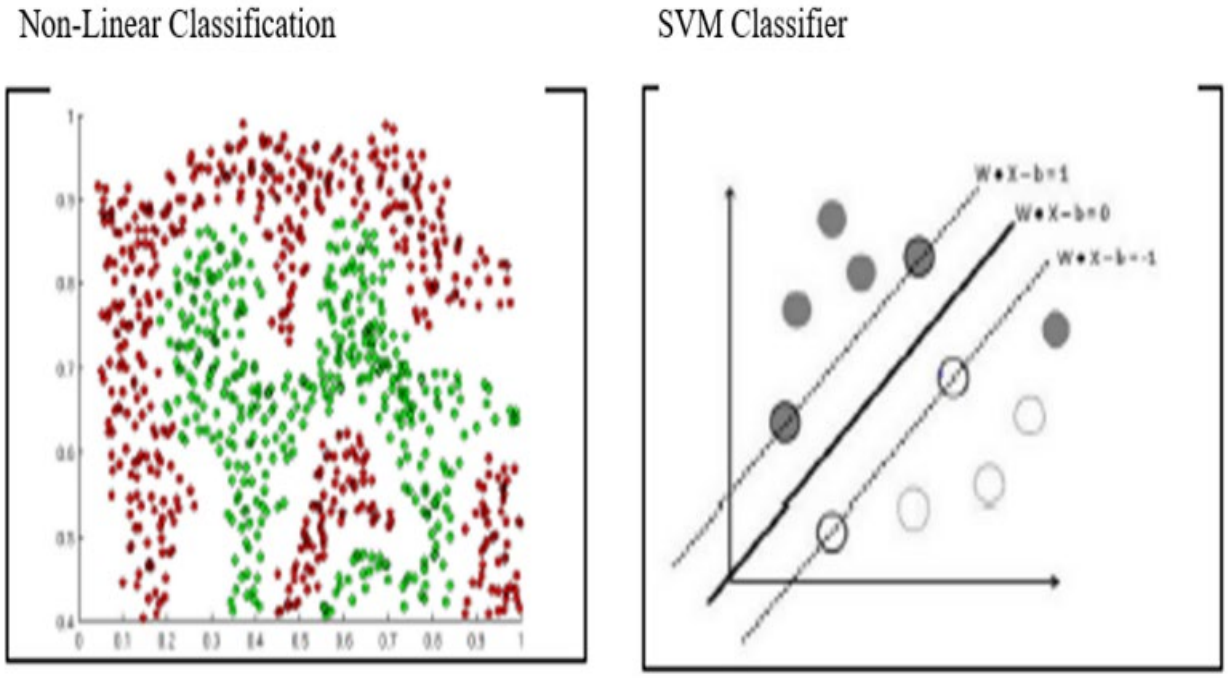
Here the decision boundary can also be explained as the technique used in regression analysis, a statistical procedure to estimate the variable relations. The regression function, which helps with the curve fitting, uses the cost estimation from the training data depicted as (6), retrieved from Ghosh et al. (2019). The variables in the equation can be explained as J implying the function of squared error cost on training data, m represents the number of samples trained, θ_0 represents the parameters for the hypothesis h_θ and θ_1 represent parameters for the hypothesis h_θ. xi represents the independent feature, and Yi represents the actual value which helps to find the minimized error (p. 1) (Equation 2).

$$J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x_i) - Y_i)^2 \quad (6)$$

SVM is capable of handling multiple categories, which can be classified as continuous. Figure 87, retrieved from Ghosh et al. (2019), depicts a sample of non-linear classification along with the associated SVM classifier (p. 2).

Figure 87

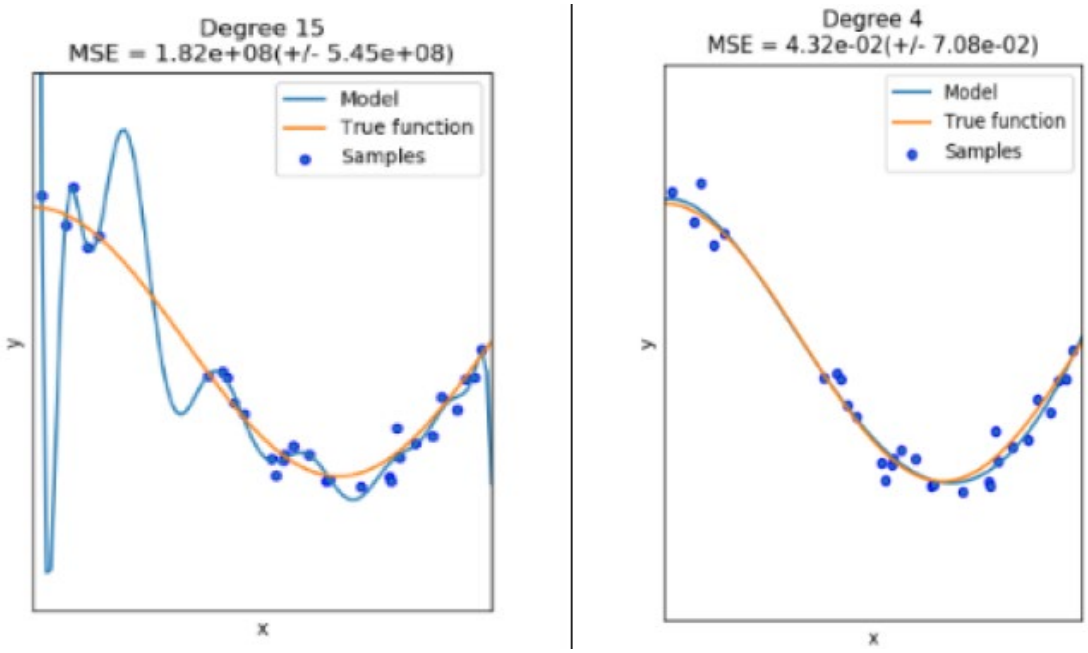
Sample of Non-Linear Classification and SVM Classifier



The circles closer to the hyperplane are the support vectors, and the filled circles can be classified as outliers that ought to be ignored to avoid overfitting. The primary goal of the SVM can be explained as the capability to minimize the distance between the edges of the hyperplane and consequently reduce the generalization error. Figure 88, retrieved from Ghosh et al. (2019), depicts SVM's efficiency in higher dimensional spaces that tend to be slow while learning. Higher dimension space means that the number of feature vectors is greater than the training sample (p. 1-2).

Figure 88

Sample of Over-Fitting and Almost Perfect Fit



Optimization of Support Vector Machine

An efficient machine learning algorithm can be attained by experimenting with the parameter that can be tuned. The process helps identify an optimal set of parameters that promotes the efficiency of the prediction process to an exceptional level. The parameters that can optimize the results of an SVM's predictions will be discussed in detail in this section.

Random State. The random state helps ensure how the generated split of data can be reproduced. The scikit-learn function facilitates random combinations when generating the training and testing data split. The value fed to the random state acts as the seed in the random state generator, confirming that the generated random numbers follow a specific required order.

Kernel. One of the essential hyperparameters induced while optimizing the SVM is the Kernel parameter, which helps map the observations into various feature spaces. The parameter aids in making the observations linearly separable with ease. Some options available for

customized usage of kernels are linear, polynomial, and radial. Choosing the right hyperparameter can affect the separability of the classifications to a greater extent. This also helps increase the performance of the algorithm.

Gamma. While utilizing the RBF or a poly kernel, a hyperparameter known as Gamma comes in handy. Especially in scenarios where the complexity is exponential, utilizing the Gamma parameter with a poly kernel helps. The Gamma parameter can be explained as the feature of the SVM algorithm that controls how much the distance of influence of a training point can be assigned. A low Gamma value means a larger radius which implies grouping more points together, whereas a high value indicates the opposite.

C. A parameter that helps add a penalty to each point that is not classified correctly is called C. A smaller value of C indicates the penalty is minor, implying that the margin-based boundary can be enormous. When a larger value is assigned to the C parameter, the SVM attempts to minimize the total number of misclassifications that occur by reducing the width of the margin.

Degree. When the chosen kernel for the SVM algorithm tends to be poly, the parameter degree comes to play. Degree implies the degree of the polynomial, which can be used to identify the hyperplane while splitting the data.

Random Forest

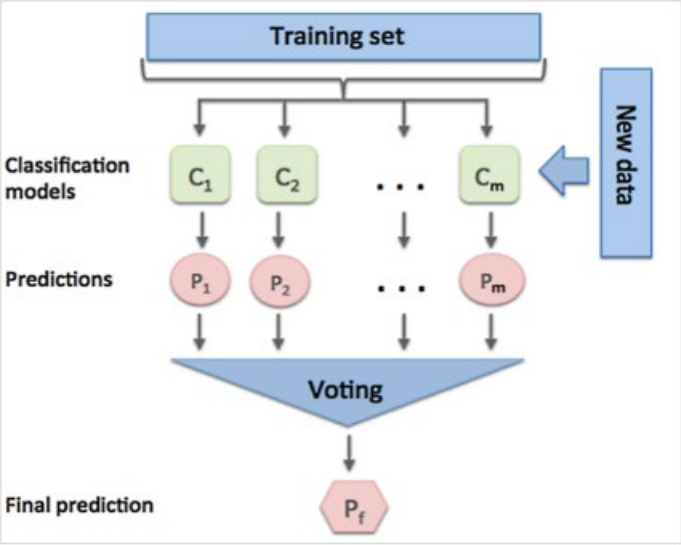
The model chosen to classify the travel mode was the Random Forest. This model is implemented on a bagging ensemble based on decision trees; it applies to classification and regression. This supervised ML model combines multiple decision trees into one for an ensemble algorithm. This was done as a prediction from a single decision tree could be incorrect; hence a combined average or voting will improve the model's performance. However, in a Random

Forest, the selection splits are not computed considering all the features but only via a random subset comprising a relatively lesser number of features to reduce the variance. Each decision tree is trained using the relevant dataset's common impurity criterion. This model works well in a high-dimensional environment with a large training data size. The best functioning solutions of this model are determining the maximum features per tree, having bootstrap samples available, and splitting performance. The warm start procedure can be performed using this algorithm.

Figure 89, retrieved from Ahmad (n.d.) (Section 2, Traditional Supervised Learning Algorithms section, understanding classification algorithms section, Understanding the ensemble methods section, Using Random Forest for prediction), depicts an example of how the Random Forest Model for classification.

Figure 89

Random Forest Mechanism



The technique provides unique advantages, like creating several classification trees without adding noise. The model randomly selects variables and data from the source set to generate a forecast. A Random Forest classifier is useful when working with various data sets

with a high degree of dimension. Furthermore, Random Forest methods have faster training rates, which results in less expensive computing. It was interesting to see how many studies employed the Random Forest Classifier as their selected model while doing background research on the topic. The Random Forest prediction depicted in figure 2 shows the model's working. The model will first get a set of inputs for training after the train and test split. The data is then simultaneously set to multiple decision trees. These trees have a different set of features based on which it is trained. The test data is passed to the trained model to predict the outcome. A classification model is implemented; therefore, a majority is taken, and that class will be predicted as the outcome. To better understand the chosen model, background research was conducted to understand the architecture, workflow, and hyperparameter tuning that can be added to this model. The section below covers the in-depth background research and working methodologies, along with optimization performed in detail.

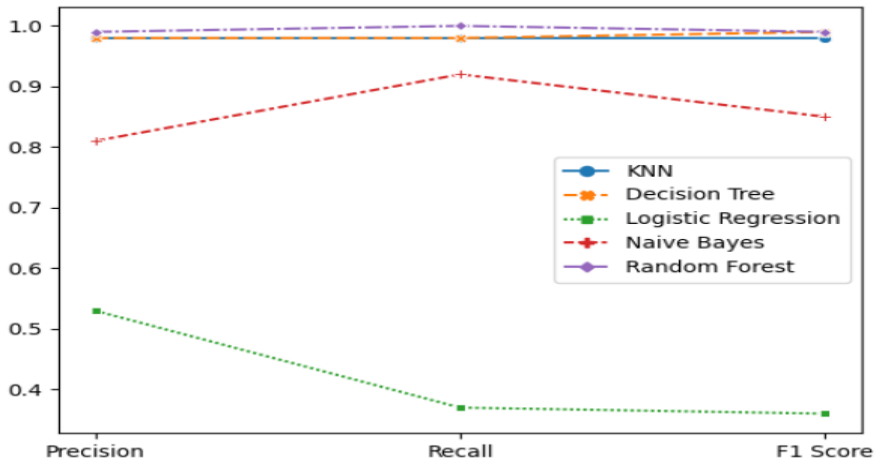
Technology and Solutions Survey. Recently, the exponential growth of data gathered by GPS-enabled devices has helped open a new avenue with interest targeted toward how this data can be put to good use. One such identified usage can be attributed to analyzing the trajectories to understand human behavior and decision-making skills. Various areas in which this data can be used are being explored and published frequently. In order to comprehend the current methodologies and their limitations, a thorough examination of related research papers was accomplished. This section discusses a few of these works.

Tišljari' et al. (2021) proposed an approach to classify the trip modes from data obtained from the cellular network. The data study was conducted in the Croatian city of Rijeka. An origin-destination matrix method was implemented, where this matrix will contain information regarding the number of trips in a given time. Each trip was considered unimodal, and the

observed area used the traffic model to generate the trips. Some of the other steps carried out as a part of the workflow were cleaning the data of anomalies that were present due to highly skewed results. This was handled by removing the data point, as parametric assumptions were not adopted by this method. Additionally, aggregated features were added to the existing feature set, such as duration, speed, and distance (pp. 1-2). The machine learning algorithms that were used are Decision Tree (DT), Random Forest (RF), K-Nearest Neighbor (KNN), Navies Bayes (NB), and Logistic Regression (LR). Furthermore, a standard split of 70: 30 was done to obtain the train and test split. Evaluation metrics were executed to analyze the models' performance, including recall, F1 score, accuracy, and confusion matrix. From observation, the RF model had the best performance, with an accuracy of 99.93%. The limitation of models, such as LR and NB, was that it was not best suited for classification. The use of automatic feature extraction from the origin-destination matrix, which can be enhanced by passing the heat map to a Convolutional Neural Network (CNN) as input to estimate the traffic in different areas, was recommended for further research (pp. 2-4). Figure 90 shows the different evaluation metrics for the chosen five ML models retrieved from Tišljari' et al. (2021) (p. 4).

Figure 90

Precision, Recall, and F1 Score for ML Algorithms



Lazar et al. (2019) suggested using unique longitudinal life course data to predict the travel modes the user regularly uses. The machine learning model was used to understand the pattern in the data. Furthermore, sequence clustering and tree-based ML were combined to interpret the travel mode. These models were coupled with a TreeExplanier for analyzing the mid to long-term outcomes. The proposed method used the trajectory sequences to cluster them based on characteristics such as longitudinal and latitude (pp. 1-2).

Optimal Matching (OM) was carried out to measure the differences in the distance; following this; a joint sequence approach was performed to calculate the differences in multiple life dimensions of these trajectories. A package called TraMineR was used to implement the steps mentioned above. In addition to these steps, hyper-parameter turning was done to generalize the model. This method combined the clusters with tree-based ML models such as RF, XGboost, Catboost, and LightGBM. With the tree-based models arises the overfitting issue; this was resolved using the gradient boosting method. The ensemble tree will have a robust accuracy when using large datasets (pp. 2-3).

In order to offer context for evaluation and interpretation, five life course clusters were derived. These clusters were grouped based on observable life trajectories; this will help obtain data across these populations to avoid bias towards a particular group. The feature selection was made using a SHAP value; a positive value indicates a higher likelihood of using the mode. Using this value helped categorize the modes and predict the likelihood that the user will choose a particular mode (p. 3).

As a result, this may enhance the newly proposed TreeExplainer method to discern between predictor relevance locally and globally more clearly, as well as predictor interactions among subpopulations across different life history settings. The performance of approaches for

forecasting the "used own" car mode was good. Considering the mode of "use public" transit and "walk/bike," Random Forest and XGBoost exceed CatBoost and LightGBM, especially in the Recall and F1 measures. Figure 91, retrieved from Lazar et al. (2019), shows the classification performance using the ML models (pp. 3-4).

Figure 91

Classification Performance on Test Data

	Method	Acc.	Prec.	Recall	F1
used public	Random Forest	**0.8216**	0.6021	0.5217	**0.5590**
	XGBoost	0.8193	0.6045	0.4813	0.5359
	CatBoost	0.7761	0.4804	0.4048	0.4394
	LightGBM	0.8008	0.6274	0.1994	0.3026
walk/bike	Random Forest	0.8807	0.6066	0.3882	0.4735
	XGBoost	**0.8908**	0.6894	0.3812	**0.4909**
	CatBoost	0.8622	0.5031	0.1882	0.2740
	LightGBM	0.8700	0.8049	0.0776	0.1416
used own	Random Forest	**0.8391**	0.8495	0.9529	**0.8983**
	XGBoost	0.8258	0.8420	0.9433	0.8898
	CatBoost	0.7530	0.8479	0.8147	0.8310
	LightGBM	0.7949	0.7936	0.9795	0.8768

F. de S. Soares, Revoredo et al. (2019) study demonstrated how to combine travel mode and trip detection into a single unit. The purpose of these trips will contribute to identifying the regularly used transportation and the reason behind the choice of mode at a given time. As a

result, two new research problems—the identification of travel modes and the forecasting of trips—have emerged in recent years. Most earlier works addressed these issues separately, resulting in needless duplication of preprocessing operations. For both issues, a single preprocessing technique was suggested to gather the location histories from the smartphone's sensors. Two Auto ML techniques were implemented, namely Random Search and Bayesian optimization. These models used cross-validation techniques to pass the training and test data to ensure an average accuracy of the model can be obtained. This will ensure that different train sets will be used to train the model during each stage to avoid bias. These models were evaluated using accuracy, precision, recall, and F-measure (pp. 1-3).

Random Forest performed best in most circumstances, while Up-Sampling is the most recommended resampling method. Bayesian Optimization was the most effective automated ML strategy during the five-minute search period. The cross-validation tests' highest accuracy for travel mode recognition and trip purpose prediction was 88% and 81%, respectively. The Support Vector Machine (SVM) was used as a hierarchical classification detection algorithm to classify the motorized and non-motorized vehicles. These classifiers worked parallel with the Auto ML models to predict the different modes. In order to further analyze, a time window was configured through such hyperparameter testing an optimal time window was identified. With just 60 seconds of location data from a smartphone, our solution can identify the mode of travel and predict the purpose of a trip. This enables context-aware information systems to use this contextual data to offer individualized services and more accurately anticipate user behavior concerning mobility patterns and transportation preferences. Future studies carried out can explore incorporating additional sensor data and other contextual information that can enhance the proposed solution's performance and evaluate a prototype that can be field tested in real-time.

Figure 22, retrieved from F. de S. Soares, Revoredo et al. (2019), depicts the performance analysis of the two Auto ML models (pp. 3-5).

Li et al. (2021) suggested a method that uses GPS trajectory data and Geographic Information System (GIS) information to predict travel modes. The methodology used feature selection to classify the mode. The most common mode was walking, often a part during a change in motorized travel modes or an initial path to reach a mode. As part of the initial analysis, the walk modes were classified by setting a threshold value over speed, acceleration, time, and distance (pp. 1-2).

Adding features such as average, variance, and percentile to velocity, acceleration, and speed opened room for more relationships between the features. GIS information was used to obtain details such as the start and end points of a trip and a ratio between the low-speed point in modes like buses and subways. The proposed method uses local data to calculate the features based on the GPS point on the grid. The second method uses the Point-Of-Interest query provided by the online mapping. The ML algorithm used for the classification of the travel modes was Decision Tree, AdaBoost, Random Forest, Extreme Gradient Boosting (XGBoost), LightGBM, and Artificial Neural Network (ANN) (pp. 6-8).

Random Forest classification algorithm to infer different modes of transportation, such as walking, biking, taking the bus, driving a car, and taking the subway, our method achieves 91.1% accuracy when applied to the GeoLife GPS trajectory dataset. Our method's use of GIS features increased overall accuracy by 2.5% and recall for the bus and subway transportation mode categories by 3.4% and 18.5%, respectively. High inference accuracy on the GeoLife dataset demonstrates that this algorithm can produce satisfactory results when used with GPS trajectory data with low capture frequency, potentially lowering the frequency at which GPS data

must be captured to identify different modes of transportation. Thus, the amount of energy a mobile device uses to collect GPS data can be decreased. Incorporating our GIS features, with a minor increase in data storage and processing resource consumption, will likely improve the inference accuracy of many algorithms used in identifying the modes of transportation from GPS trajectory data. Figure 92, retrieved from Li et al. (2021), depicts the Random Forest algorithm's classification using different feature sets (pp. 9-11).

Figure 92

Random Forest Classification Accuracy

Classification accuracy with different features using RF algorithm.

(%)

Feature	Train samples accuracy	Test samples accuracy
1–11	100	88.5
1–13	100	89.0
1–15	100	91.1

Confusion matrix of RF algorithm with Features 1–11.

		Inference					Recall (%)
		Bike	Bus	Car	Walking	Subway	
Ground truth	Bike	**263**	5	0	28	0	88.9
	Bus	2	**298**	41	10	0	84.9
	Car	3	41	**177**	4	7	76.3
	Walking	11	5	1	**658**	1	97.3
	Subway	1	3	17	10	**77**	71.3
Precision (%)		93.9	84.7	75.0	92.7	90.6	–

Confusion matrix of RF algorithm with Features 1–13.

		Inference					Recall (%)
		Bike	Bus	Car	Walking	Subway	
Ground truth	Bike	**266**	4	0	26	0	89.9
	Bus	2	**300**	38	11	0	85.5
	Car	5	37	**179**	3	8	77.2
	Walking	14	7	0	**654**	1	96.7
	Subway	1	3	17	6	**81**	75.0
Precision (%)		92.4	85.5	76.5	93.4	90.0	–

Saini et al. (2017) proposed a trajectory classification that combines the Support Vector Machine (SVM) and Random Forest with Fuzzy C-Means (FCM) clustering techniques (SVM). Understanding the flow of movements in the monitoring area is made more accessible by filtering or categorizing rare or abnormal events in traffic data. In order to safeguard the region under observation, object trajectories from films are extracted and analyzed to look for the odd behavior of objects. The methodology incorporated was to have mid-level feature extraction; these are then passed to clustering using Fuzzy C-Means. These clustered results are then considered the final set of features that will be passed to the models (p.1).

Position coordinates, velocity, and angular features were taken from the input video trajectories and were the three features that were extracted during the mid-level extraction step. Firstly, in trajectory classification problems, position coordinates have been regarded as a critical characteristic that uses 2D domain positioning data for the trajectory. Secondly, velocity, which is the difference between two subsequent video or GPS trajectory points, velocity characteristics are calculated. Since different classes have different trajectories' speeds, this information is beneficial in the categorization process. Lastly, applying angular features to a variety of trajectory classification issues. The angle of a point was calculated by computing the sine and cosine angles made using the coordinate axes. These features are then passed to the Fuzzy C-Means to obtain the final feature set (pp. 1-3).

Additionally, the model's performance using a different Random Forest (RF) based classifier was compared. The models were evaluated on the CROSS [12] and T11 [18] datasets, two publicly accessible datasets. The accuracy of the Random Forest model using the CROSS data was 88.75%, and the T11 data was 86.01%. When comparing the to the SVM model, it is less. The categorization rates can be raised in the future by taking more substantial elements

from the trajectory data. In addition, combining techniques like decision fusion of various classifiers might help the classification process. Figure 82, retrieved from Saini et al. (2017), shows the accuracy of the base model Random Forest and clustering model Support Vector Machine (SVM), and the SVM model has better accuracy in both datasets (pp. 3-4).

Similarities between the Existing Works. Several research papers were examined as part of the literature review to offer perceptions into applying various algorithms and data aspects and incorporating additional methods to building a prediction model. The similarities between the studies' methodologies and data sources demonstrate which ones are appropriate and frequently utilized when considering travel mode detection and classification. The first similarity was the use of trajectory data, such as GPS data, GIS information, and cellular-network data; in the research paper by Saini et al. (2017) used a public dataset that contains different trajectory classes collected through cellular networks. The work of Li et al. (2021) used the combined data of GPS and GIS to classify the trips and travel modes. F. de S. Soares, Revoredo et al. (2019) used data collected through smartphone sensors. Secondly, based on the additional feature extraction from raw data, the work of Saini et al. (2017) used mid-level feature extraction to obtain features such as velocity, angular features, and position coordinates. Li et al. (2021) extracted the average, variance, and percentile features of velocity, acceleration, and speed. Lazar et al. (2019) used an optimal matching matrix to obtain distance differences. Thirdly, all the papers used machine-learning algorithms RF for modeling. Lastly, the transportation mode covered in all the chosen research papers had a wide range of motorized and non-motorized classes for classification—some of the most commonly existing walk, bus, subways, jog, and car.

Contrasts between the Existing Works. There are some parallels between the works mentioned below but also some differences. These disparities relate to the method incorporated

aim is approached, how many features are taken into account, and how evaluation methods are used. In this part, these opposing observations are thoroughly examined. The first difference in the dataset chosen in some cases, two to three different datasets were used to see how the model performs the work by Saini et al. (2017) used two publicly available data. Tišljari' et al. (2021) used the data collected within a given region through cellphone sensors to understand patterns better. On the other hand, Gao et al. (2020) collected data from GPS sensors and devised ways to reduce the energy consumption taken for this process. Secondly, how the features extracted in Saini et al. (2017) paper the feature extracted was passed through a cluster to obtain a final set which is then passed to modeling. Wang et al. (2018) extracted features through aggregation and set threshold values to classify each mode. Lastly, studies used different evaluation metrics to estimate the model's performance. Wang et al. (2018) and Gao et al. (2020) used accuracy as a metric, whereas Li et al. (2021) used recall to evaluate. Furthermore, F. de S. Soares, Revoredo et al. (2019), Lazar et al. (2019), and Tišljari' et al. (2021) to measure the performance of the model used F1 score, precision, recall, and accuracy. Understanding these similarities and differences in the paper will help devise a plan to understand how the objective can be achieved along with the ways to achieve.

Working Methodology of Random Forest. Background research gave a better insight into how Random Forest is best suited for classification problems. This section will cover the working of the RF model in detail, as it is an ensemble technique that combines multiple decision trees during the training phase. The decision tree uses an ID3 algorithm to split the data based on the best feature; this step is carried out until there is no more feature to split. The tree obtained will have a root node, leaf node, and sub-trees. Figure 93 below, taken from Kelleher et al. (2015) (pp. 172-173), shows the ID3 algorithm.

Figure 93

ID3 Algorithm

Algorithm 4.1 Pseudocode description of the ID3 algorithm.

Require: set of descriptive features d
Require: set of training instances $\mathcal{D}$
1: if all the instances in $\mathcal{D}$ have the same target level C then
2: return a decision tree consisting of a leaf node with label C
3: else if d is empty then
4: return a decision tree consisting of a leaf node with the label of the majority target level in $\mathcal{D}$
5: else if $\mathcal{D}$ is empty then
6: return a decision tree consisting of a leaf node with the label of the majority target level of the dataset of the immediate parent node
7: else
8: d [*best*] $\leftarrow \arg\max_{d \in \mathrm{d}} IG(d, \mathcal{D})$
9: make a new node, $Node_{\mathrm{d}[best]}$, and label it with d [*best*]
10: partition $\mathcal{D}$ using d [*best*]
11: remove d [*best*] from d
12: for each partition $\mathcal{D}_i$ of $\mathcal{D}$ do
13: grow a branch from $Node_{\mathrm{d}[best]}$ to the decision tree created by rerunning *ID3* with $\mathcal{D} = \mathcal{D}_i$

The ID3 algorithm requires a set of features present in the dataset. As a next step, this data will be split for training and testing. The first check is done to see if all the instances have the same target level; if that is present, then a decision tree is returned with the target class as a leaf node. If the previous check fails, the second check is done on the dataset feature; if empty, a leaf node containing the majority target class is returned as the decision tree. The third check step is done if all these previous two checks fail where if the training set is empty, then the tree will return the leaf node with the target class that is majority with parent node. The next iteration will happen when all the above three checks fail. Information gain is calculated for the feature

set, and the feature with the maximum gain is selected. The chosen feature is the parent node based on which the data is split. Following the split, the feature is removed from the feature. These steps are carried out until it passes the check, where the decision tree is finally obtained. The Random Forest model creates multiple independent decision trees with different feature sets. Figure 94, retrieved from Bonaccorso (2020) (Ensemble Learning, Random Forests, para. 18), shows the algorithm of the Random Forest Model.

Figure 94

Random Forest Algorithm

1. Set the number of decision trees *Nc*
2. For *i=1* to *Nc*:
    1. Create a dataset *Di* sampling with replacements from the original dataset *X*
3. Set the number of features to consider during each split *Nf* (for example, *sqrt(n)*)
4. Set an impurity measure (for example, Gini impurity)
5. Define an optional maximum depth for each tree
6. For *i=1* to *Nc*:
    1. Random forest:
        1. Train the decision tree *di(x)* using the dataset *Di* and selecting the best split among *Nf* features randomly sampled
    2. Extra-trees:
        1. Train the decision tree *di(x)* using the dataset *Di*, computing before each split *n* random thresholds and selecting the one that yields the least impurity
7. Define an output function averaging the single outputs or employing a majority vote

The Random Forest algorithm works by first setting the number of independent decision trees. The next step involves sampling records and features from the dataset for each decision

tree. An impurity measure is chosen. Following this step, we need to set the maximum depth of the tree to avoid overfitting issues. The next step will have iterations for each decision tree to be built based on the impurity measure of the best feature split is done, and this is carried out until all the tree has a target class that is pure or until maximum depth is reached. The Random Forest model is trained, and the majority vote determines this model's outcome, and the selected class is used as the forecast for a specific record in this model. This will improve the model's performance and avoid the overfitting issues that might arise due to under-processed data. The model can be enhanced by applying ensemble techniques to reduce the overfitting of data, reduce bias and improve the model's prediction accuracy.

Optimization of Random Forest

Random Forest, when compared to a decision tree, is optimized and robust. To handle the data, this model must be optimized to avoid the issues of underfitting or overfitting the model. This can be carried out by adjusting the parameter of the sklearn random classifier model. The result of this optimization will improve the prediction accuracy of the RF model.

Estimator. Random Forest often tends to overfit, which can be attributed to how the working mechanism was designed. To combat this issue, the "n_estimator" parameter was designed, which enables the user to customize the number of decision trees a model can have.

Maximum Depth. The maximum depth can be set to ensure the leaf node does not expand further. In certain scenarios, an estimator can be void of use; in such cases, the overfitting can be dealt with by setting diverse values for the "max_depth" parameter.

Minimum Sample Split. A Random Forest constitutes many decision trees interweaved together to form a cohesive algorithm. The trees are often split till each leaf node acquires a classification value. Optimal splitting of the trees can cater to the model's precision, and for this

purpose, the "min_sample_split" parameter is used. The purpose of the parameter can be explained as identifying the least number of samples needed for splitting an internal node.

Random State. This scikit-learn parameter helps decide on the reproduction of split data. There is often a random permutation happening while data is split; customizing this can help diversify the learning mechanism. The value assigned to a random state can be described as a seed that facilitates the data split in a specified manner.

Number of Jobs. Random Forest is a combination of decision trees which implies that multiple jobs can be run simultaneously. Optimizing the "n_jobs" parameter helps assign how many jobs can run parallelly. This often helps with enhancing the performance of the algorithm. The values of this parameter can be set to a none or negative one. When the value is set to none, representing a positive one, the process is carried out in the backend. When the value is set as negative, the model will use all the processors to train the model.

Minimum Sample Leaf. The model needs to generalize when a new set of data is passed to achieve a model that is not overfitted; a value can be set for the "min_samples_leaf" to ensure the least number of samples are needed to be present at the leaf node. Split points should have at least a minimum number of samples in both the left and right branches taken into consideration.

Minimum Impurity Decrease. The nodes at each level will have certain features based on which the split was carried out. This can be controlled by setting the parameter "min_impurity_decrease," which will split the feature when decreasing the impurity measure.

Criterion. This will set the measure of impurity to determine the split will be carried on which descriptive feature. The measure can be calculated using the Gini index and entropy to calculate the information gain that can be obtained. Table 10 shows the formulas to calculate the Gini index, entropy, and information gain retrieved from Kelleher et al. (2015) (pp. 164, 168,

186). The information shown in Kelleher et al. (2015) (p. 168, Equation 4.4) depicts the measure that will be calculated for every feature present in the dataset. These measures can be used to decide the split of the features in the decision tree.

Table 10

Equation for Measure of Impurity

Name	Formula	Description
Entropy	$H(t, D) = \sum_{i=1}^{l} (P(t = i) X \log_2(P(t = i)))$	Dataset *D* contains target features *t*. *levels (t)* is the set of levels in the domain of the target feature *t*. *P (t = l)* is the probability of a randomly selected instance having the target feature level *l*.
Gini Index	$Gini(t, D) = 1 - \sum_{levels \in (t)} P(t = l)^2$	Dataset *D* contains target feature *t*. *levels (t)* are the set of levels in the domain of the target feature. *P (t = l)* is the probability of an instance having the target level *l*.
Information Gain	$IG(d, D) = H(t, D) - rem(d, D)$	*d* is a particular descriptive feature, and *D* is the feature to be split on. *H(t, D)* is the entropy of the dataset. *rem(d, D)* is the remainder entropy of the features in the dataset.

Maximum Features. The dataset will have multiple descriptive features that will help predict the classes. The parameter "max_features" can have values such as sqrt, log2, none, integer, or float. If the chosen value is an integer, then all the set number of features at each split. By setting this value, we can control the number of features considered at each stage.

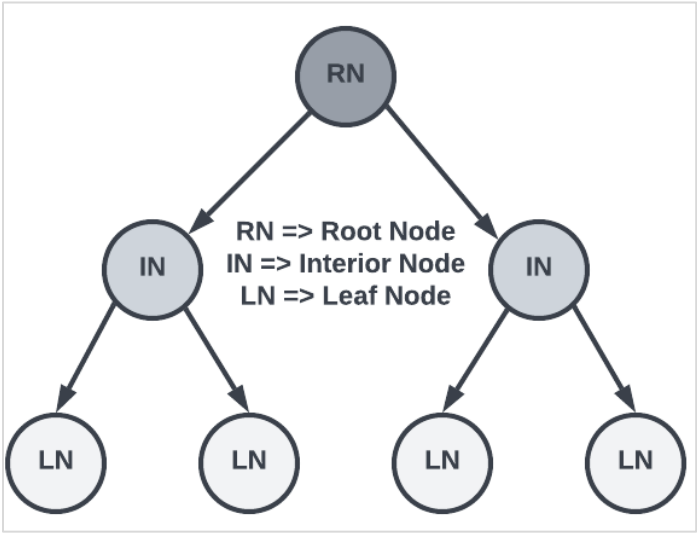
The Random Forest model can be tuned using a variety of hyperparameters, which are predetermined before the learning process starts. Hyperparameter tuning Random Forests will enhance the model's performance while minimizing the loss and producing improved output. The upcoming section will cover the tools, software, hardware requirements, and platforms used.

Decision Tree

A decision tree was chosen as a viable model for carrying out the transportation mode classification. A decision tree makes a decision based on a set of rules. Figure 95 depicts a decision tree and its components. It demonstrates that the tree has a root node from which interior nodes are generated based on a condition represented by the branch and that a leaf node demonstrates the classification result.

Figure 95

Decision Tree Structure



A decision tree's goal is to find a pure set, meaning that the tree's end nodes result in a single target level. To find this out, a measure of impurity becomes helpful, indicating which set of features has less mix concerning different target levels. This impurity can be measured by two measures which are entropy and the Gini index. Based on these measures, the informativeness of features present in the dataset can be identified. Thus, an appropriate root node to split the dataset for the prediction can be identified, and therefore with fewer steps, a prediction result can be achieved. Implementation of a decision tree is easy as well as the interpretation. In addition, there are no pre-assumptions made about the classifier structure as it runs on the observed data. Moreover, as entropy and the Gini index are used to check the impurity, it is not required to perform normalization. The points mentioned above indicate the advantages of using a decision tree model. However, some drawbacks could also be handled during the implementation. Some of those drawbacks are the instability of the structure of the tree with the addition of new features and the tendency to overfit. These drawbacks can be handled if proper hyperparameters are chosen during the implementation phase. For instance, the overfitting issue can be handled by introducing a tree pruning or ensemble method.

Technological and Solution Survey. It is crucial to identify a model appropriate for detecting and classifying the transportation modes from the trajectories with better prediction accuracy. In order to achieve this goal, a thorough analysis of past research works helps show the correct path to follow. This section explains the past research works which were referenced to achieve the project’s objective.

Safitri and Surjandari (2017) proposed a methodology to predict the travel mode-switching behavior of both Transjakarta passengers and non-passengers in the Jakarta Greater Area region. A decision tree model was utilized to predict the switching behavior. This study

aimed to improve public transport planning, which was only accessible to one percent of the population. The data was collected using an online survey, which is a questionnaire with three parts. The first part was related to the demographic details of the survey respondents, the second was related to the Transjakarta passengers, and the last was associated with the non-passengers. The indicators used in the survey are the number of trips, journey duration and distance, travel mode, trip purpose, cost of travel only for public transport, public transport availability, vehicle ownership, driver's license, and level of exposure to traffic. In total, 239 valid survey responses were collected, from which 120 belonged to the passengers and the rest to non-passengers. In the first stage, a travel behavior survey was performed to understand the number of respondents and their travel behavior in each group based on the survey response, such as residence, place of daily activities, profession, and satisfaction. The approach suggested predicting the travel mode choice was a decision tree using the C4.5 algorithm. A decision tree was structured using attributes such as driving license ownership, private vehicle ownership, alternative transportation mode, and consideration of other transportation modes. The evaluation result on the 120 datasets of passenger data showed that 66.99% of data were correctly classified, whereas the remaining 33.01% were incorrectly classified. Figure 96, which is retrieved from Safitri and Surjandari (2017), demonstrates the decision tree of the passengers to switch to other transportation modes (pp. 1-4).

Similarly, the decision tree for the non-passengers showed 61% of data correctly classified and the remaining 39% incorrect. Figure 97 depicts the decision tree of the non-passengers decision to switch travel mode as retrieved from Safitri and Surjandari (2017). The outcome of the study implied that 60.8% of the passengers and 57.1% of non-passengers preferred to switch their current travel mode. Thus, this study emphasized that more focus on the

availability and easy accessibility to Transjakarta services is required. The limitation associated with the study was the number of samples for conducting the survey. Future research recommended a deeper analysis of the transportation mode selections and using a decision tree with a different algorithm than C4.5 (pp. 4-5).

Figure 96

Decision Tree Predicting Passenger's Mode Switch Decision

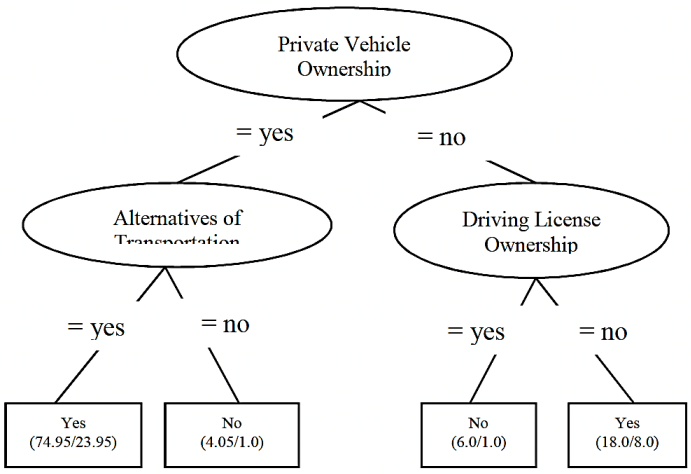
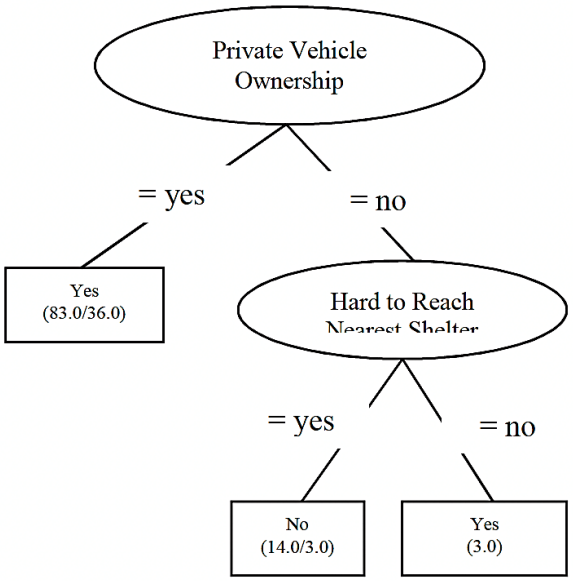


Figure 97

Decision Tree Predicting Non-Passenger's Mode Switch Decision



Yi Hou et al. (2014) proposed an approach to develop a real-time lane changing assistant system that assists and gives proper advice to drivers on whether to merge into a lane based on safe and unsafe gaps while making lane change decisions. This study differed from the previous works, where the focus was on blind spot identification, and the goal was to develop a simulation model instead of a real-time system. This paper suggested a methodology to obtain the best result by combining the Bayes and decision-tree classifiers to provide higher accuracy and give more importance to the accuracy of non-merge events rather than the merge events for safety purposes. The data used in this study was traffic data which is Next Generation Simulation (NGSIM) data provided by Federal Highway Administration. This dataset included vehicle trajectory data on two traffic segments; one is U.S. Highway 101 in Los Angeles, CA, and another one is Interstate 80 in San Francisco, U.S. data. The two datasets recorded the data on a different date for each dataset in 2005 for a period of 45 minutes. Each dataset had two traffic states known as a transition period and congestion condition. The quality of speed data in the NGSIM dataset was improved by using the moving average smoothing technique. In addition, only trajectory data of vehicles in auxiliary (merge) and adjacent (target) lanes were used for modeling (pp. 1-2).

The model-building approach used additional input variables which could affect a driver's decision-making capability: the merging vehicle's speed difference between the lead and the lag vehicle in the target lane, the merging vehicle's gap distance between the lead and the lag vehicle in the target lane and the distance to the beginning of the lane from the merging vehicle. The relative speed distance between the lead vehicle in the target lane and the merging vehicle was considered the most appropriate feature in the classification task of merge and nonmerge events, which were considered by the weight estimation of all the input variables using SVM. The Bayes classifier was developed using a risk of misclassification as one and a k value of three for

calculating the model's optimal kNN density estimation parameter. The prediction accuracy for the crucial nonmerge events provided by this classifier on the test data was 79.3%, and for merge, events were 92.3%. A decision tree with 18 terminal nodes was developed for the prediction task. As in the case of the Bayes classifier, in the decision tree, the effective node for making a root node was the relative speed distance between the lead vehicle in the target lane and the merging vehicle (pp. 3-8).

The accuracy generated by this model for both merge and nonmerge events was more than 80%. Figure 98 depicts the accuracy result achieved by both classifiers, which is retrieved from Yi Hou et al. (2014). The result provided by combining both classifiers using a majority voting rule improved the accuracy for the important nonmerge events to 94.3%. Figure 99 depicts the results achieved by the combined classifier with different misclassification weights as retrieved from Yi Hou et al. (2014). The result showed that the prediction accuracy of nonmerge events increased with the increase in the misclassification weights and decreased for the merge events. For a future scope, for the design of lane changing assistant system, other modeling techniques can be implemented and compared with the result achieved by the combined classifier (pp. 3-8).

Figure 98

Accuracy Rates of Different Models

Decision	Validation data			Test data			
		Bayes Classifier	Decision Tree		Bayes Classifier	Decision Tree	Combined Classifier
	Observations	Accuracy	Accuracy	Observations	Accuracy	Accuracy	Accuracy
Non-merge	73	82.2%	89.0%	459	79.5%	84.3%	94.3%
Merge	56	91.1%	85.7%	208	92.3%	80.8%	79.3%

Figure 99

Combined Classifier Sensitivity to Misclassification Weights

Decision	Observations	$\frac{\lambda_{21}}{\lambda_{12}} = 2$	$\frac{\lambda_{21}}{\lambda_{12}} = 5$
		Accuracy	Accuracy
Non-merge	459	95.4%	96.7%
Merge	208	73.6%	49.5%

Brunauer et al. (2013) proposed an approach to utilize only GPS data for detecting common travel modes. Unlike the previous works where the recognition model used GPS data along with some GIS or real-time information, this study focused on using only GPS data with a dense dataset of rural and urban areas having diverse physical and motorized travel modes. For the data collection approach, a wide variety of GPS devices were utilized where the sampling rate of the trajectories varies from one to ten seconds. Five users recorded trajectory data for over 17 months for data collection purposes, which involved 1000 business and private trips. This study used 400 single-mode trajectories collected from the collection process, and 54 classification features were calculated from GPS data points for further use. The methodology for developing and evaluating the algorithm for detecting travel modes was accomplished by MLP(Multilayer Perceptrons), Logistic Model Trees(LMT), and C4.5 Decision Trees. (p. 1).

The MLP had three layers where 54 features derived from the GPS data point were fed to the equivalent number of neurons of the input layer; the output layer consisted of one neuron for each travel mode. A C4.5 decision tree was used for linear classification, and to handle nonlinear classification, another type of tree, LMT, was used. For validation purposes, k-fold cross-validation was implemented, and k was selected based on a stratified sampling technique. For evaluation, the model's performance was checked using two approaches: one with the complete

feature set and another with the best feature subsets selected after a feature selection process. The result demonstrated that with a full feature set, the accuracy achieved by MLPs and LMTs was 92.24% and 92.09%, respectively. On the other hand, the decision tree had an accuracy of 84.48%. Figure 100 portrays a confusion matrix with a full feature set as retrieved from Brunauer et al. (2013). Concerning the feature subsets, with 38 features, the overall accuracy was 93.29%, but with the small feature set of nine, the accuracy of 90.27% was preferably good. Figure 101 portrays the accuracy along with the number of feature subsets which is retrieved from Brunauer et al. (2013). One limitation of the study was not achieving a unique feature subset that could be optimal for the classification task. In the future, this study can be extended by identifying and adding features that could enhance the accuracy (pp. 3-7).

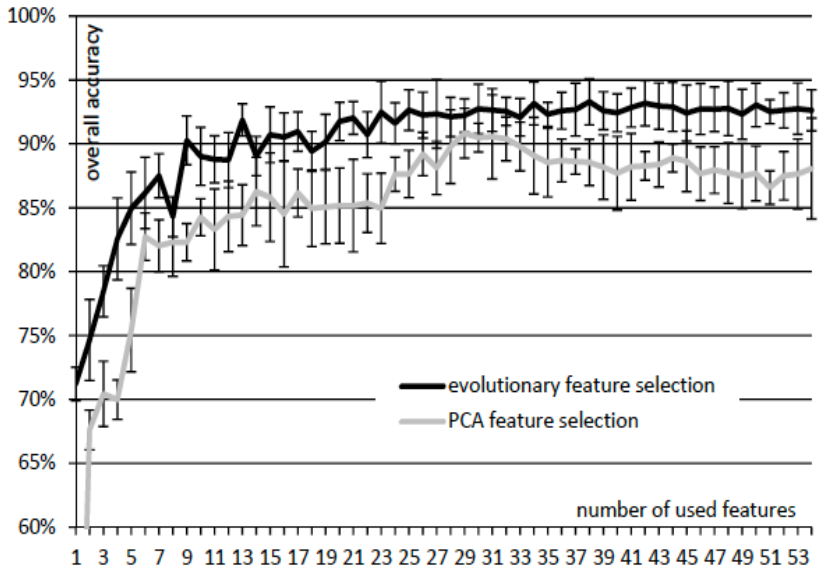
Figure 100

Confusion Matrix with Full Feature Set

			observed					
			car	*train*	*bus*	*bicycle*	*walk*	precision
predicted	*car*	MLP	5164	518	181	8	0	87.96%
		C4.5	4479	878	186	82	6	79.54%
		LMT	5148	465	149	11	0	89.17%
	train	MLP	193	2749	36	2	8	92.00%
		C4.5	828	2325	91	9	0	71.47%
		LMT	244	2796	42	18	1	90.16%
	bus	MLP	290	56	1707	42	31	80.29%
		C4.5	250	124	1539	165	50	72.32%
		LMT	254	73	1741	75	62	78.96%
	bicycle	MLP	3	0	23	4146	91	97.26%
		C4.5	65	3	140	3909	89	92.94%
		LMT	0	4	26	4068	66	97.69%
	walk	MLP	0	27	53	2	4670	98.27%
		C4.5	28	20	44	35	4655	97.34%
		LMT	4	12	42	28	4671	98.19%
	recall	MLP	91.40%	82.06%	85.35%	98.71%	97.29%	**92.24%**
		C4.5	79.27%	69.40%	76.95%	93.07%	96.98%	**84.48%**
		LMT	91.12%	83.46%	87.05%	96.86%	97.31%	**92.09%**

Figure 101

Accuracy with Feature Subsets



Rezaie et al. (2017) observed that most of the existing travel mode detection methods depended on fully validated data. Hence, a new approach was proposed using the semi-supervised method to use both the validated and un-validated data. There were three pieces of information used in this approach for data. The first was GPS data for user trips, the second was General Transit Feed Specification (GTFS) data, and the last was a city zone map dataset. The collected GPS data were preprocessed to derive the trip information. The feature vector proposed here had five components: total trip travel time, average trip speed, trip length, the minimum distance between the trip origin and the closest transit stop, and the minimum distance between the trip destination and the nearest transit stop. Two methods were proposed: supervised methods, DT and RF, and one semi-supervised method, Label Propagation (LP) (p. 1-2).

These models were estimated with different meta-parameters to identify the best one. DT mode detection used a recursive partitioning in which the partition of the training set was done based on the thresholds of the features. A reduced pruning approach is also considered to avoid

over splitting. In RF mode detection, each tree contains the most suitable split, derived from a subset of candidate features, which minimized the effect on variance. In LP mode detection, label propagation along with KNN was used. KNN approach used the Euclidean distance to measure the similarity. The experiment was performed with a different ratio of unlabeled data to ensure the stability of the result. Each suggested model is compared with the other with a meta-parameter setting. The meta-parameters for DT CP are 0.01, and the minimum number of observations for pruning was 20. The meta-parameter for RF was ten classifiers. For LP, the clamping factor was assigned to one, and the nearest neighbor k is set to be below five for labeled data and above five for most unlabeled data. Figure 17 shows that the number of unlabeled observations did not impact the average accuracy of the supervised models, which is retrieved from Rezaie et al. (2017). In contrast, the accuracy was improved for the semi-supervised algorithm as n increases (pp. 1-4, 6-7).

The proposed methodology used a decision tree, random forest, and KNN kernel. The main focus of the study was comparison instead of improvement. The results showed that when the dataset contained more than 70% of unlabeled data, the semi-supervised algorithm performed better and vice versa. Since this study is among the few initial studies involving semi-supervised data, there were a few errors in the estimations, which can be improved with a more detailed dataset. Future enhancements could be using a better dataset and feature selection (pp. 7-8).

F. de S. Soares et al. (2021) devised an approach to detect the travel mode using real-time data since most previous works rely on offline data. Since processing is done within smartphone devices, real-time detection could lower the cost and latency for ITS applications. Information will also be current, allowing for quicker action. The proposed approach involved developing an API called ActivityRecognitionAPI for travel mode detection, which used supervised machine

learning on real-time data from smartphone sensors. Rather than relying on noise removal techniques and expensive segmentation, travel mode was inferred from the features extracted from chunks of location traces every 90 seconds. The hierarchical classification was used based on their previous approaches, which showed they led to better performance, differentiating motorized and non-motorized using Sequential Minimal Optimization (SMO) with SVM pre-trained. The SVM model is used to identify motorized vehicles; similarly, a Decision tree or Bayesian Network identifies the mode, such as walking or biking. Implementing the classification as mentioned earlier, SVM achieved an average accuracy of approximately 62.2%.

Figure 18 summarizes each classifier's performance metrics, as shown by F. de S. Soares et al. (2021). The algorithm's performance is tested using a confusion matrix. The study results showed that the ActivityRecognitionAPI had higher precision in recognizing vehicles. However, it has a low recall, which was a limitation when the requirement is related to the quantity of the inference. The study can be enhanced further by including a vast range of users, modes including aquatic and airborne, and usage of different settings such as urban and rural (pp. 1, 3-4, 9-10).

Jahangiri and Rakha (2015) suggested a methodology for detecting multiple travel mode classifiers using several supervised learning approaches K-Nearest Neighbors(KNN), Support Vector Machine(SVM), Random Forest(RF), decision tree, and bagging. It differed from previous works concerning the consideration of data from various smartphone sensors, including the gyroscope, rotation vectors, and accelerometer, without using GPS data. The data collection approach involved a smartphone application that recorded data coming from various sensors, as mentioned above. As the data was collected from multiple sensors, thus, to ensure an identical sampling rate during the gathering process, data were resampled at a rate of 100 Hz after performing an initial sampling using a linear interpolation technique. A total of 25 hours of data

was collected. Coming to the model development, KNN was used, and a majority voting rule was applied to find the prediction. The SVM model was developed considering two primary factors: Gaussian kernel and implementing feature scaling. In the case of SVM modeling, two approaches were implemented and examined: a single SVM model with the full dataset and an ensemble SVM model with a smaller dataset. For tree-based classifiers, three models were identified. A decision tree model was developed using the Gini index to find the best split for each node. The second one was a bagging model where the variance of the tree was reduced by averaging or majority voting technique. A random forest model was developed to tackle the issue of highly correlated trees in bagging, where a random subset of features was selected for growing each tree. The top 80 features were chosen as part of the feature selection process for modeling using the minimum redundancy maximum relevance (mRMR) approach. The models were evaluated using 5-fold Cross-validation and Out-Of_bag error techniques. With respect to the result of all the models, KNN gave an accuracy of 91.2% with seven neighbors. The accuracy generated by the single and ensemble SVM models was 94.62% and 94.41%, respectively. The decision tree gave an accuracy of 87.27%, and it became 86.3% after pruning. Similarly, the overall accuracy generated by the bagging and random forest model was 95.1% and 94.4%, respectively. Figure 79 depicts the model comparison results using various performance measures such as F-score, Discriminant Power(DP), and Youden's index, which is retrieved from Jahangiri and Rakha (2015). In conclusion, the best performance was obtained by SVM and RF. The drawback of this paper was the misclassification of car mode to bus mode four to six percent of the time. As a future scope, this study can be extended further to enhance the detection performance in different ways (pp. 1, 3-11).

Similarities between the Existing Works. The papers discussed above share some similarities described in this section. Examining similarities aids in comprehending the commonly used existing approaches and methodologies. One such similarity is related to the algorithms used. Most of the papers implemented supervised algorithms. For instance, Safitri and Surjandari (2017) proposed a supervised algorithm, a decision tree, for achieving the objective. Similarly, Yi Hou et al. (2014) implemented two supervised algorithms, a decision tree and Bayes classifiers, to accomplish the goal. Another similarity is related to the validation method used for the suggested models. In the majority of the studies, k-fold cross-validation was followed for validation purposes. For example, F. de S. Soares et al. (2021) and Brunauer et al. (2013) used a k-fold cross-validation approach to validate and estimate the performance of the proposed models.

Contrasts between the Existing Works. Although the papers discussed have some similarities, there are some differences between them. Exploration of the identified dissimilarities aids in understanding the diverse set of approaches available, which could be explored further in the current paper. The first one was related to the type of dataset used in different studies. In some studies, it was found that real-time data source was utilized, whereas in others depended on the existing GPS datasets. For instance, Yi Hou et al. (2014) utilized a dataset provided by Federal Highway Administration, whereas F. de S. Soares et al. (2021) used real-time smartphone sensor data. Though it was mentioned that there is some similarity between the methodologies used, some studies still exist that experimented with a different method. For example, Rezaie et al. (2017) proposed a semi-supervised approach to accomplish the research objective. The observed contrasts were helpful in formulating the necessary steps to proceed with the execution of the project.

Working Methodology of Decision Tree Model. Following the technological review of existing works, proper guidance was obtained to implement the decision tree model in order to achieve the project's objectives. This section elaborates on the working methodology associated with a decision tree. In detail, it provides an idea about the algorithm and pseudocode logic behind the selected model. Several algorithms are available for analyzing a decision tree, such as Iterative Dichotomiser 3(ID3) and C4.5. The algorithm helps understand the internal working mechanisms of the model. Figure 94 represents a pseudocode description of the ID3 algorithm, retrieved from Kelleher et al. (2020) (pp. 172-173).

This algorithm follows an iterative and depth-first approach. The first two lines in the pseudocode depict that if there is only one target level for all the target instances, then a decision tree returns that target level as a prediction. Next, if there are no descriptive features left, then using a majority voting mechanism means that a decision tree with the majority target level becomes the model's prediction. Another case where this majority voting mechanism is applied to the immediate parent node of the majority target level leaf node is when their training instance is empty. If the above conditions do not match, then the best node to split is identified after calculating the information gain.

Suppose a descriptive feature exhibits maximum information gain. In that case, it gets selected for the dataset partitioning, and in the subsequent steps, this feature will not be considered further as a splitting candidate. This process iterates, and the tree grows until a pure set with the target levels are not generated. This splitting can go to a level such that the model cannot generalize better in the testing phase. Tree pruning or ensemble techniques are applied accordingly to mitigate such issues. Tree pruning is often helpful in also to reduce the complexity of the model by eliminating further classifications where unnecessary.

Optimization of Decision Tree

A successful implementation of an ML model can often be attributed to identifying optimal parameters that suit the scenario. Various parameters are made available intended for this purpose, and properly tuning the parameters increases the precision of the model. There are ways to further optimize the decision tree classifier model by using some significant parameters; some of those parameters are discussed in detail in this section.

Maximum Depth. A decision tree tends to overfit; thus, to mitigate this issue, a "max_depth" parameter can be used, which restricts the number depth of a tree. It accepts integers, and the default value is None. If it has the default value, the tree grows until a pure set is achieved or until all the terminating nodes or leaf nodes consist of less than the minimum number of samples for each internal node with a default value of 2.

Criterion. A decision tree's goal is to find a pure split and thus to measure the quality of a split "criterion" parameter is used, which has a default value of "gini," and it supports two other types: "entropy" and "log_loss."

Splitter. In addition, a split strategy can be applied to decide the split at each node by using the "splitter" parameter with appropriate input. It supports two strategies which are "best" and "random," based on the requirement of best or random splitting, respectively. The default one is "best."

Random State. It is essential to handle the randomness of splitting nodes. Therefore a "random_state" parameter can be utilized with an integer value like 0,42, or 233 to control the randomness.

Weight of Class. For calculating a split's purity while fitting the model, a parameter, "class_weight," becomes helpful, associating different weights with the target class labels. It

supports a dictionary, a list of a dictionary, or "balanced ."The default value is set to None, which means all the class labels have a weight of one.

Moreover, other than the parameters explained here, some other useful ones can significantly help optimize the model if used accordingly. "min_samples_split" and "min_samples_leaf" helps in identifying the minimum number of splits or the lowest number of samples at a terminating node. Similarly, other parameters assist in selecting the maximum number of features, leaf nodes, and minimum weight of a decision tree.

Long Short-Term Memory

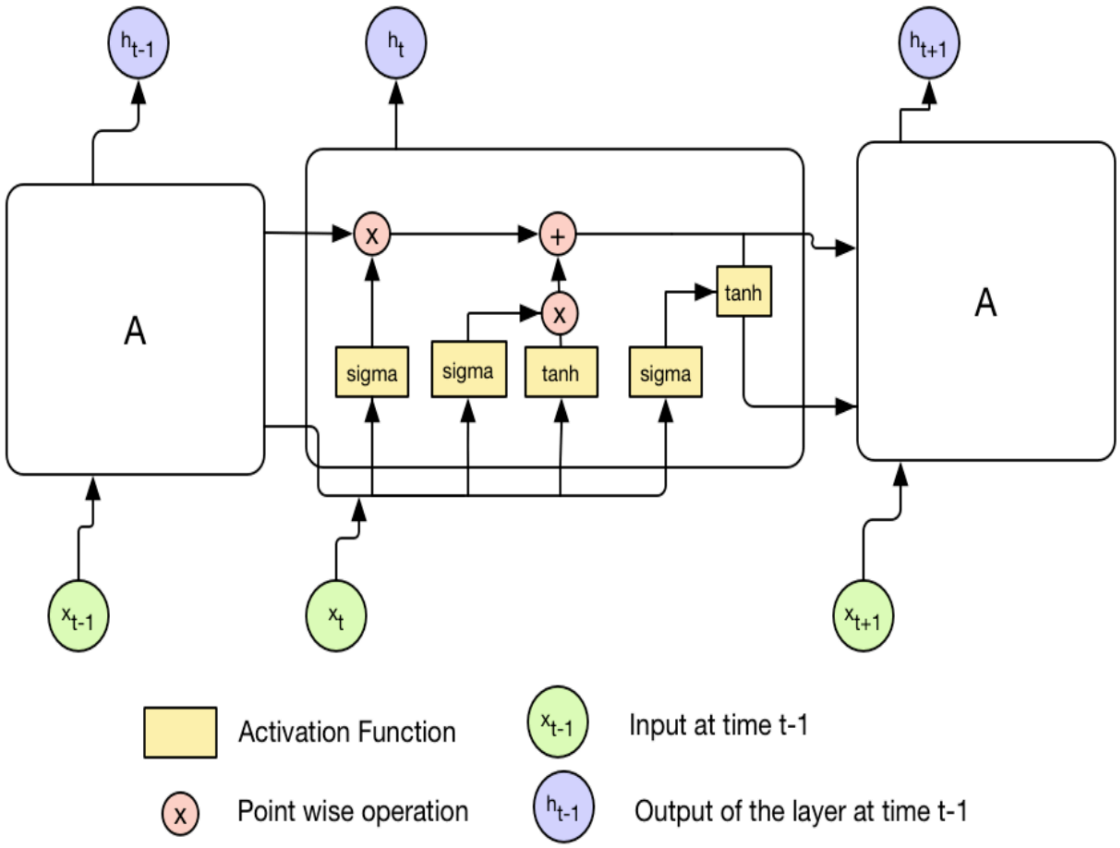
Traditional neural networks cannot persist information and thus have a disadvantage when predicting the next event based on the previous one. A Recurrent Neural Network(RNN) is used to solve this problem, and RNNs employ loops, which aid in the retention of information. An RNN is simply a collection of identical copies of standard neural networks, with each network passing information or messages to the succeeding network. Although it can persist information, it is not helpful in long-term dependencies. It means if there is a long gap between the point at which information is required and the point at which the relevant information is available, it becomes difficult for RNNs to connect such information. In this case, where a long-term dependency is required to learn and understand the context, a special type of RNN called an LSTM network is used. LSTMs have a chain-like structure where the repeating module has four neural network layers which interact in a specific way, unlike a single neural network layer in RNNs. Figure 102 depicts the structure of an LSTM retrieved from Dua and Ghotra (2018) (Generative Adversarial Networks, Recurrent Neural Network, LSTM networks).

In figure 102 , each line shows the path of an entire vector flowing from one node to the other as an input. Each circle marked in pink color depicts the pointwise operations such as

vector addition and multiplication. The yellow boxes represent the activation function or the neural network layers. The horizontal line through the top depicts a cell state where the LSTM adds or removes information controlled by gates, as shown in the yellow box, and the number of gates is three. The first gate represents a sigmoid layer known as forget gate, which generates an output of either zero or one for a cell state at time t-1. If the value is zero, a state is not retained; on the other hand, if it is one, then the state is retained. Post this; two layers decide which information to be stored in the cell state. The first sigmoid layer, known as an input gate, decides on the value to be updated, and the second layer, a Hyperbolic Tangent Function (Tanh) layer, creates the new values that need to be added.

Figure 102

Structure of LSTM Model



Technology and Solution Survey. Ji et al. (2020) proposed a Seq2Se1 model utilizing the LSTM prediction network to model the trajectories. The proposal also extends to detecting abnormal trajectories that are cosplaying as normal ones. The SL modeling utilizes the original features without needing to extract additional features, adapting the sequence length variations manually. Another unique concept introduced in this study was sequencing, which uses spatio-temporal and semantic information. The similarity between the trajectories is calculated to pick out the abnormal trajectories. The study's results showcase how the trajectories obtained were sufficiently descriptive and represented the trajectories appropriately. The accuracy attained by the models was higher than the existing methodologies. The proposed method's detection ability was sterner than the other models while being computationally less time-consuming. Figure 103, retrieved from Ji et al. (2020), depicts the performance comparison of the model's results while detecting the abnormalities based on the spatio-temporal methods and the trajectory outlier detection based on the entropy distribution method (pp. 1,10).

Figure 103

Performance Comparison of the Models

Performance index	Abnormal trajectory detection with spatio-temporal and semantic information	Abnormal trajectory detection based on TODCSS
Accuracy	91.351%	83.784%
Precision	85.294%	77.444%
ART(s)	111'74	176'86

Liu & Lee (2017) proposed a method to classify the transportation models by adding the semantics of human behavior to the recorded raw trajectories. The belief of the proposal was the utilization of the model for the sectors such as trip recommendation, transport planning, and traffic management. Unlike the previous works, which primarily relied on extracting features assisted by a time-consuming detailed pre-processing phase, the study devised a method utilizing

Recurrent Neural Networks. The study utilized a bidirectional LSTM classifier, specifically, end-to-end for travel mode detection. The framework was further improved by embedding time interval features externally while modeling. The model's performance was evaluated using AUC curves, and the results show that the model outperforms the existing methods. The highest accuracy attained by the model was recorded to be around 95%. The future scope of the project can be attributed to the focus on more diverse transportation labels. The other implementation that can be added to this model in the foreseeable future is incorporating the supervised and unsupervised models. Figure 104, retrieved from Liu & Lee (2017), depicts the results of the study (pp. 1,4).

Figure 104

Experiment Results

Model	AUC
SVM	0.86
Random Forest	0.88
Bi-LSTM+3L+64H	0.88
Bi-LSTM+3L+100H	0.89
Bi-LSTM+5L+100H	0.94
Bi-LSTM+5L+300H	0.94
Bi-LSTM+3L+64H+Embedding	0.89
Bi-LSTM+3L+100H+Embedding	0.90
Bi-LSTM+5L+100H+Embedding	0.945
Bi-LSTM+5L+300H+Embedding	**0.946**

Xu et al. (2019) devised a method to detect travel modes and believed that the generated information could be helpful in various scenarios involving urban design, monitoring activities, and planning journies in real time. The proposed approach is an in-grained detection utilizing a kinetic energy harvester (KEH). The information received from the output signal of a KEH was used as the input to the transportation mode detection model. Each transportation mode has its own unique signature patterns, and the nature of its being distinctive could be helpful for detection. An LSTM framework was considered to achieve higher precision while detecting the

transportation modes. The data set chosen was collected over three months by volunteer users carrying the developed prototype, and the resultant model attained an overall accuracy of over 97%. Additionally, the measures demonstrated that the power consumption was only 460uW and significantly outperformed the existing models. The limitation of the work was that the prototype design was on the heavier side, being more extensive. The future scope can rely on optimizing the prototype and reducing its size. Figure 105, retrieved from Xu et al. (2019), depicts the results observed during the evaluation of the proposed model (pp. 1,9,10).

Figure 105

Results Attained by the Study for Individual Travel Modes

	Bus	Car	Ferry	Light rail	Train	Walk	Run
Bus	96.5%	1.2%	0.8%	0.5%	0.5%	0.2%	0.3%
Car	0.9%	95.5%	0.85%	1%	1.05%	0.41%	0.29%
Ferry	0.25%	0.14%	98.5%	0.5%	0.26%	0.15%	0.2%
Light rail	1%	0.23%	0.21%	97.8%	0.46%	0.18%	0.12%
Train	0.25%	0.5%	0.2%	0.15%	98.5%	0.1%	0.3%
Walk	0.4%	0.6%	1.1%	0.7%	0.2%	95.8%	1.2%
Run	0.5%	0.3%	0.4%	1.1%	0.3%	0.6%	96.8%

Dai et al. (2019) proposed utilizing the LSTM model for trajectory analysis and travel mode detection while also addressing the existing issues with the model. The primary identified issue was the existing LSTM model's inability to describe the interaction between different vehicles spatially. Also, there exists an inability to address the temporal relationship of the time-series data. The second issue addressed in this paper is the vanishing gradient issue suffered by the LSTM models, which makes it harder to train the time-series data. The LSTM model proposed in this study sported two modifications. A shortcut connection was introduced between two consecutive LSTM layers, which helped handle the vanishing gradient issue. The model was evaluated using an I-80 and US-101 datasets and demonstrated a higher accuracy than the existing models suggesting the model can be state-of-art. The limitation of the model was observed during the scenarios involving limited-scale training, and the experiments showed that

a more robust generalization capability is needed. This can be addressed as a part of the future scope. Figure 106 and 107, retrieved from Dai et al. (2019), depicts the loss variation observed by LSTM with and without shortcuts (pp. 1,9)

Figure 106

Loss Variation Observed by LSTM with and without Shortcut

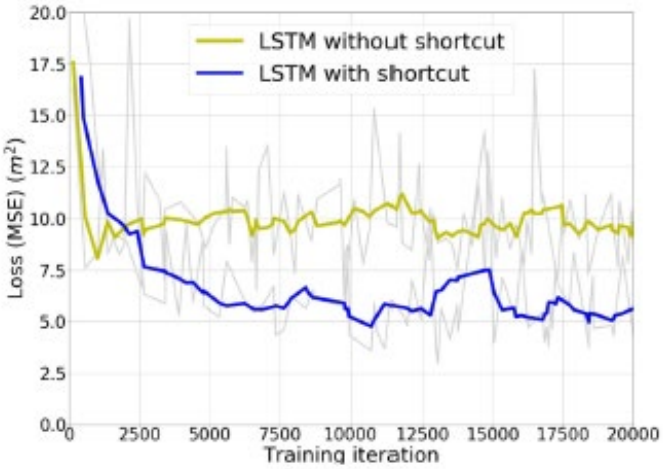
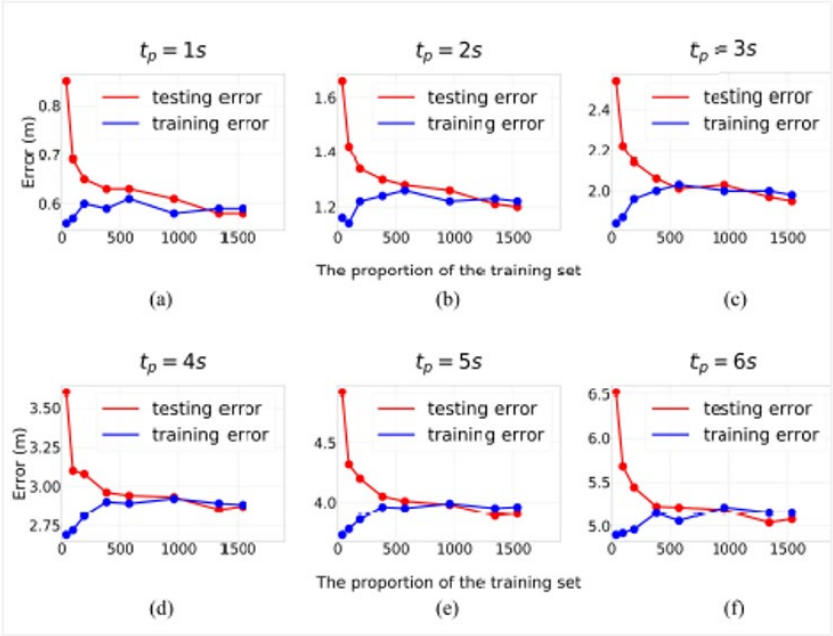


Figure 107

Comparison of Training and Testing Errors Across Various Training and Testing Sizes



Zhao et al. (2022) proposed a trajectory analysis model, a mobility predictor based on the LSTM model. A reinforced learning technique was used to accelerate the model convergence rate, which in turn helps automate selecting the best neural network architecture. In order to further substantially enhance the learning environmental search process, the architectural knowledge learned from a teacher LSTM has been transferred to a student LSTM using a learning framework. The proposed study also showcases improved networks enabled by edge-computing using a predictive handover algorithm. The algorithm helps apply the prediction information, thus reducing the handover-failure rate while migrating the edge-computing network layer. The study results showed that the proposal is efficient and impacts the ping-pong handover by improving that along with the service migration. Figure 108 and 109 depict the algorithm employed for monitoring and assigning the Reinforced Learning Based Service Migration (RLSM). Figure 110 depicts the number of service migration failures by various approaches. The greedy algorithm recorded the highest failure rate. The results of the study showcase that the proposed model works better in comparison with the existing models.

Figure 108

Algorithm of RSLM Monitoring

Algorithm 1: RLSM Monitor.

input: Mobility prediction model for each user in the network.
output: Migration decision for each user.

```
while user is connected do
    Perform mobility prediction;
    Estimate when HOs will be triggered based on the predicted
     user trajectory;
    if HO is eminent then
        Perform migration decision;
    Measure QoS;
    if QoS is below the threshold then
        Perform migration decision;
```

Figure 109

Algorithm of RLSM Assignment

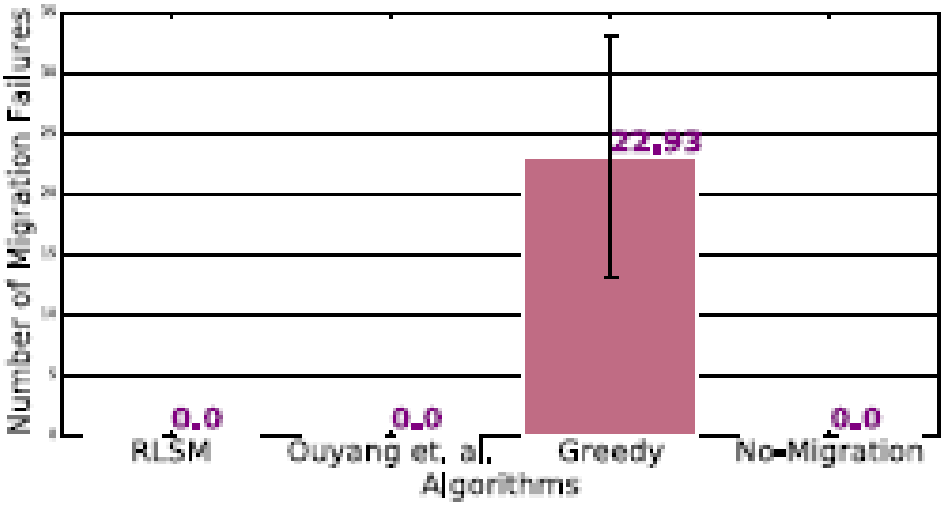
Algorithm 2: RLSM Assignment.

input: User not provided with minimum requirements, list of available servers.
output: ID of the best server for the user, migration operations.
Data: Minimum requirements of the service

```
1: List available servers;
2: Remove servers lacking resources for the UE application from
   list;
3: for Each available edge server do
4:    Get QoS for the server;
5: while Server has not been chosen do
6:    Get the closest server to the UE's future location;
7:    Estimate migration time;
8:    if Migration can be done before the UE's arrival and Server
      QoS is above threshold then
9:       Choose this server as target;
10:   else if Current connected server QoS is below threshold and
      Server QoS is above threshold then
11:      Choose this server as target;
12:   else
13:      Remove this server from list;
14: Perform migration;
```

Figure 110

Number of Service Migration Failures



Similarities between the Existing Works. The existing works helped in realizing the optimal approaches that would result in the execution of the model with higher accuracy and precision. Diving into the existing works helped attain insights into how the various kinds of data were processed and handled in those works. There were quite a few similarities observed throughout the works that were analyzed, and this section will summarize some of the observed similarities. The first observed similarity was in the kind of data source chosen. The works by Zhao et al. (2022) and Liu & Lee (2017) both preferred utilizing the GPS-recorded trajectory information. The second similarity observed was the involvement of spatio-temporal information for improving the prediction results. The studies by Dai et al. (2019) and Ji et al. (2020) used the mentioned technique to enhance the results. The third similarity was that all the papers mentioned above in the technological and solutions survey section use the LSTM model as the baseline with further enhancements. The studies by Ji et al. (2020) and Liu & Lee (2017) preferred not to extract additional features and relied on the original set. These similarities portrayed the common traits of enforcing LSTM Modeling.

Contrasts between the Existing Works. A few contrasting opinions were observed between the papers during the background research process. The contrasting opinions shed light on how multiple angles can approach a problem with various techniques. Most of the contrasting opinions focused on data selection and methodologies. Some limitations of the paper were solved with the goal of another work. Some of the contrasting opinions that were observed are listed in this section. The first observed discrepancy was related to the specifications of the LSTM models. For instance, the study by Liu & Lee (2017) used a bi-directional classification LSTM model. In contrast, the work by Dai et al. (2019) does not involve the utilization of the bi-directional classification LSTM. The second observed contrasting factor was how the data was

sourced for implementation. For instance, the study by Xu et al. (2019) opted to use a self-designed prototype for data collection. In contrast, the study by Liu & Lee (2017) used a readily available GPS-recorded dataset. The third observed contrasting opinion was the involvement of reinforced learning. The work by Zhao et al. (2022) preferred utilizing reinforced learning, whereas the study by Dai et al. (2019) did not opt for it. The contrasts helped identify the varying approaches in solving travel mode detection and helped formulate the optimal approach.

Working Methodology of LSTM. This section will cover the working of the algorithm after a detailed study of the existing paper to understand how this model will help achieve the project's goal. Figure 111, retrieved from Kumar et al. (2021) (p. 6), shows the step involved in the LSTM algorithm. The model avoids issues related to long-term dependencies by storing the information for longer durations. This model has a less complex structure, such as a continuous chain-like structure, but the modules that occur again will have a different structure.

The first step is to create a set of sequential input data from the collected historical data. The model normalizes the input data to avoid any bias from features that could have values in the higher range. Following this, add layers in the order in which they need to relate to each other. These layers initially consist of a first layer where the samples, timesteps, and features are defined. A sequential model can be created by stacking the layer on top of each other. When stacking, the output of the layers should be in sequence instead of having a single value.

The data points obtained needed to generate weights for them; also, the model will set recurrent data weight. The additional parameters that will be set are peephole weight and offset. The algorithm work on a given time 't,' and the input and output data are trained. The input is passed through all the layers present in the networks in a sequential order to obtain an outcome. The formulas used to obtain the prediction are discussed in detail in the following section.

Figure 111

LSTM Algorithm

Algorithm: Long Short-Term Memory (LSTM)
Input: Normalized data
Output: Closed stock price

Begin
Step 1: Set input data weight: W_i, W_f, W_c, W_o
Step 2: Set Recurrent data Weight: R_i, R_f, R_c, R_o
Step 3: Set peephole weight: $V \in R^N$
Step 4: Set Offset: b_i, b_f, b_c, $b_o \in R^N$
Step 5: At time t, x_t is the input and y_t is the output of the node
Step 6: $f_t = \sigma(W_f x_t + R_f h_{t-1} + b_f)$ is the output of the forget gate at time t
Step 7: $i_t = \sigma(W_i x_t + R_i h_{t-1} + b_i)$ is the output of the input gate at time t
Step 8: $\acute{C}_t$, C_t is the input and cell structure of the node at time t, respectively, which are expressed as:
$\acute{C} = \tanh(W_c x_t + R_c h_{t-1} + b_c)$
$C = i_t \odot \acute{C}_t + f_t \odot c_{t-1}$
Step 9: $O_t = \sigma(W_o x_t + R_o h_{t-1} + b_o)$ is the output of the output gate
Step 10: The final output h_t of the node is expressed: $h = O_t \odot \tanh(C_t)$
End

LSTM has been chosen as the first model implemented for the stated classification problem based on a background literature review. Deep learning networks (DNN) have been utilized for transportation mode detection (TMD) to increase performance over machine learning (ML) approaches since the work of Fang et al. (2016). Their approach utilized a backpropagating network with three hidden layers and outperformed all other ML classifiers in the experiment. Song et al. (2016) developed and compared several LSTM-based architectures that demonstrated high accuracy performance on a large dataset TMD task. Soares et al. (2019) implemented LSTM to perform TMD classification that demonstrated comparable performance to ML classifiers in several metrics with much less computational and memory resource costs. Asci and Guvensan (2019) utilized LSTM and observed the importance of window and frame size in the time series data for improving accuracy. LSTM can perform TMD classification for datasets with varying time domain length between transportation classes with high accuracy as demonstrated by Iskanderov and Guvensan (2020).

LSTM was proposed by Hochreiter and Schmidhuber (1997) as a means of storing long-term information during recurrent backpropagation. LSTM is a variant of recurrent neural networks (RNN). RNN are networks which can utilize sequential information to optimize weights. This is important for sequential and time series data where prior information is important relative to future predictions.

In addition to this, RNN can handle input and output with varying size, which is not possible in standard DNN. However, there are two prominent issues with standard RNN architecture which are learning dependencies and vanishing gradients. Equation (7) is the only learn short term dependencies. Equation (8) is vanishing gradient during backpropagation. LSTM was developed to address these two short comings. Information can be remembered or forgotten through memory cells by input and output gate.

$$y^{out_j}(t) = fout_j\left(net_{out_j}(t)\right); y^{in_j}(t) = fin_j\left(net_{in_j}(t)\right) \tag{7}$$

An activation function determines the memory cell's state

$$S_{c_j}(0) = 0, S_{c_j}(t) = S_{c_j}(t-1) + y^{in_j}(t) g\left(net_{c_j}(t)\right) \; for \; t > 0 \tag{8}$$

The output of the memory cell (9) is computed as

$$y^{c_j}(t) = y^{out_j}(t)\, h(S_{c_j}(t)) \tag{9}$$

The gates are key to what information is remembered as weights are optimized for minimizing loss during training. The purpose of forgetting information is to control error due to input weight conflicts. This assists with preventing easily learned short-term memories from disrupting learning of long-term memories.

The LSTM model can be optimized by controlling the network learning rate, number of units in network layers aside from output, weight initialization, dropout, weight decay rate, and number of epochs. Learning rate will help with locating global optima while avoiding vanishing and exploding gradient during backpropagation. Controlling for number of units in the LSTM layer and hidden layers regulates complexity in the model through parameter count and assist with generalized learning. Weight initialization can improve model learning by assigning optimal starting weights in the training process. Dropout regularizes the network by randomly dropping units by a set probability, preventing a single unit from having outsized weight which hinders generalized learning. Weight decay regularizes the network by preventing a single stale unit form accumulating weight. The number of epochs offers a balancing point between under and overfitting the model.

BInfer-LSTM. The BInfer-LSTM to be compared with the baseline LSTM model is the same model but with an added Bayesian dense variational layer directly after the LSTM layer. The purpose of this variational layer is to build epistemic uncertainty into the model. Complex sequential data that contains high degrees of uncertainty can be difficult to train with RNNs due to vanishing gradient. Gulshad et al. (2017) developed a solution to this problem by replacing training output mean square error with mean and variance negative log likelihood in the network backpropagation. Fortunato et al. (2017) demonstrated an approach for incorporating variational Bayes scheme into RNN backpropagation to reduce variance during training. They explain the additional information provided by the posterior probabilities increases accuracy of gradient estimates. Another approach to incorporating uncertainty was proposed by Lam et al. (2019) where the activation functions in various parts of the LSTM the memory cell were modified to account for parameter uncertainty and observed better preservation of long-term dependencies.

Nghiem et al. (2022) implemented Bayesian inference after the LSTM layer to update LSTM hyperparameter weights during model training. Their experiment demonstrated significant predictive performance improvement for time series data that produced underfitting with LSTM modeling.

The specifics of the implementation of the Bayesian dense variational layer are described in Fortunato et al. (2017) as Bayes by Backprop with Posterior Sharpening. The first step in algorithm is to sample a minibatch of time sequence inputs and targets. Second, the latent posterior distribution is created and sampled. Third, loss is calculated between the minibatch sample and the posterior sample. The fourth step computes the gradients of the loss function with respect to the posterior mean, variance, and per-parameter learning rate. The final step is the update of the posterior mean, variance, and per-parameter learning rate. Figure 112 presents the formulations involved in this learning process (p. 5).

Figure 112

Bayes by Backprop with Posterior Sharpening Algorithm

Posterior Sharpening Algorithm - Bayes by Backprop

Sample a minibatch (x, y) of truncated sequences.
Sample $\varphi \sim q(\varphi) = \mathcal{N}(\varphi|\mu, \sigma)$.
Let $g_\varphi = -\nabla_\varphi \log p(y|\varphi, x)$.
Sample $\theta \sim q(\theta|\varphi, (x, y)) = \mathcal{N}(\theta|\varphi - \eta * g_\varphi, \sigma_0^2 I)$.
Compute the gradients of eq. (8) w.r.t. (μ, σ, η).
Update (μ, σ, η).

Optimization of LSTM

Optimizations for this model are the same as the base LSTM model, with the addition of the dense variational layer activation function and scale of the posterior distribution function.

The activation function controls the transformation of minibatch data into normal distribution parameters, and the posterior distribution scale assists with convergence with a prior distribution. Optimizing the parameters can often help enhance the performance of the algorithm there by incrementing the possibility of resultant predictions to be more precise. Often various parameter tuning combinations are experimented to find the optimal permutation. This section will explain in detail some of such hyperparameters of LSTM that can be tuned.

Number of Hidden Layers and Nodes. The layers in the LSTM model are either input or output; these are called hidden layers. The number of nodes present can be set, and there is no fixed number on how many layers or nodes can be present in the model. The number is decided on multiple experiments that can be carried out to determine the optimal number. The approach uses a single layer in cases of simple problems.

Dense Layer Units. Hyperparameter controls the input obtained from the previous layer when the layers in the LSTM are stacked on each other. These multiple layers connected are known as a dense layer. The parameters, such as additional layers or nodes, will improve the model's overall performance by creating a good base on the model to build on. Specifying certain units will impact how the output is obtained from the layer.

Dropout. A dropout layer is added for every LSTM layer present in the model. These layers are added only to the input as they help avoid the overfitting issues that the data can cause. The functionality of the layer is to reduce the input's weight so that the modes' sensitivity is still intact. These nodes that are worked on are selected at random by adding these additional layers; the model will become more complex during computation. The value assigned to this parameter should always be less in number as the increase in value can not solve the issues that arise due the sensitivity and overfitting.

Decay Rate. This parameter is used to add weights to the data. If the model does not have any weight update planned, this is to ensure a specific rule to added to the nodes so that the weights will be reduced so that it comes close to zero. Following this, when weight is implemented, the value of this parameter is updated by a factor of 1. This is done to avoid the value of the weight becoming large.

Model Supports

A successful and smooth implementation can be partially attributed to satisfying the minimum required system configurations. The model support section expands on the hardware, software, and tool requirements that must be acquired for the successful implementation of the project.

Package and Library Requirements

Various packages and libraries were imported to facilitate different operations during the multitude of phases involved in the project implementation. In order to traverse through various data processing stages such as cleaning, pre-processing, transforming, and bias elimination, various libraries such as pandas and NumPy were utilized. Since the project utilizes geographic information, the geopy library was imported. Packages related to handling time features were also required since the calculations involved were instructed to process various time-related features. The package also aided while recording the time taken by various phases of execution.

The project also included exploratory data analysis, which required visualization packages to generate charts and other visualizations. After data processing, the condensed data was passed on to the modeling phase, which required its own set of packages and libraries. A significant amount of imported packages involved in the modeling phase come from the scikit-learn library, which can help invoke the various functions associated with model development, such as splitting the train and test data, data standardization, model selection, hyperparameter tuning, evaluation and validation of the attained results. Table 11 depicts the list of libraries and packages imported and installed as a mandate for the project's implementation.

Table 11

Package and Library Requirements Table

Library		Method	Usage
Scikit-Learn	sklearn.model_selection	train_test_split	Splitting the training and testing data
		StratifiedKFold	Provides indices to split the data into sets.
		learning_curve	Determines cross-validated training and test scores
		cross_validation	Implementation of Cross-validation during modeling
		cross_val_score	Model Evaluation
		cross_val_predict	Model Evaluation
	sklearn.metrics	classification_report, confusion_matrix, f1_score, precision_score, multilabel_confusion_matrix, ConfusionMatrixDisplay, precision_recall_fscore_support	Model Evaluation
	sklearn.preprocessing	StandardScaler, MinMaxScaler	Standardization of data
	sklearn.svm	SVC	Implementation and hyperparameter tuning of SVM Classification Model
	sklearn.ensemble	RandomForestClassifier	Implementation and hyperparameter tuning of RF Classification Model
	sklearn.tree	DecisionTreeClassifier	Implementation and hyperparameter tuning of DT Classification Model
Keras	keras	initializers	Used for assigned weights to the Keras layers
	keras	Sequential	Equips the model with training and inference features
	keras	losses	Used for calculating the quantity that a model must minimize during training.
	keras	optimizers	Used for compliation of Keras Model
	keras	layers	Basic foundation for building neural networks in Keras
	keras	keras_tuner	For optimizing the hyperparameters.
	preprocessing.sequence	TimeseriesGenerator	Generates timeseries data
Tensor Flow	tensorflow_probability		For probablistic reason
Geopy	geopy.distance	geodesic	Handling geographical data
Numpy	Numpy	array, percentile, mean, std	Data manipulation and processing
Matplotlib	pyplot	figure, rc, show	Creation of Visualizations
Seaborn	Seaborn	heatmap	Creation of Visualizations
Time	time	time	Additional feature extraction and time based calculations
Pandas	DataFrame	read_excel, read_csv, describe, set_option,shape, drop, head, isNull, sum, duplicated, drop_duplicates	Reading the data and performing various cleaning and transformations before modeling
	Profiling	profile_report, to_file	Analyze the features by profiling

Support Vector Machine - Model Architecture and Data Flow

During the earlier stages, the data-cleaning process, which included the elimination of outliers, nulls, duplicates, and any other abnormalities, helped condense the data and bring quantifiable information to the forefront. That cleaned data was subsequently split into training and testing sets, with the split ratio being 70:30. The optimal split was identified and performed randomly to eliminate any bias within the classification. The split data was then subsequently passed into the chosen SVM machine-learning model. This section details how this data flows through the SVM model and discusses the working mechanism employed by SVM.

In the simplest form of classification involving just two class separations for the data, which is linearly separable, the SVM model tries to find the hyperplane that could maximize the separation between the classes. This is possible when the dataset consists of two sets of classifications in a two-dimensional space point. Basically, in binary classification scenarios involving only two distinct classifications, the hyperplane is just a line that separates the two classes. The data passed on as a training set was plotted by SVM in an N-Dimensional space, where N implies the total number of features present in the data. SVM attempts to identify the optimal hyperplane which separates the data. This is straightforward in the case of linearly separable and binary classifiable data. Figure 15, retrieved from Bin Altaf and Yoo (2016), demonstrates a sample of how linear separation happens.

However, in cases where the data is not linearly separable, the SVM uses kernel tricks and mandates the usage of an algorithm that can generate non-linear decision boundaries. The kernel trick behind separating linearly non-separable binary classified data is projecting it into a higher dimensional space which can facilitate the linear separation by making it compatible with the process. Some of the available approaches for classifying linearly non-separable data involve the

utilization of kernel tricks such as polynomial-NLSVM or Gaussian Basis Function NLSVM. It is crucial to select a suitable approach to establish boundary separation on linearly non-separable data since it can directly influence the trade-off between classification and accuracy. The left side image shown in figure 113, retrieved from Bin Altaf and Yoo (2016), depicts a sample of linearly non-separable data. The right-side image in figure 114, retrieved from Bin Altaf and Yoo (2016), showcases how the data is projected and the classification boundary is established (p.3).

Figure 113

Sample of Linearly Separable Binary Classified Data

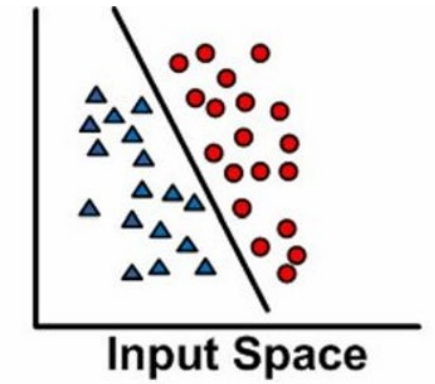
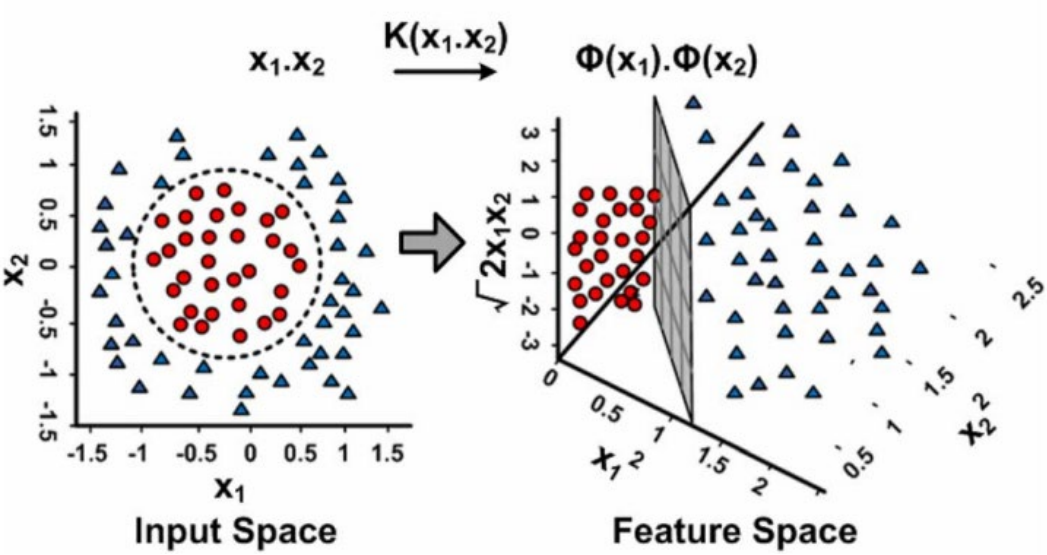


Figure 114

Sample Demonstration of Classifying Linearly Non-Separable Data



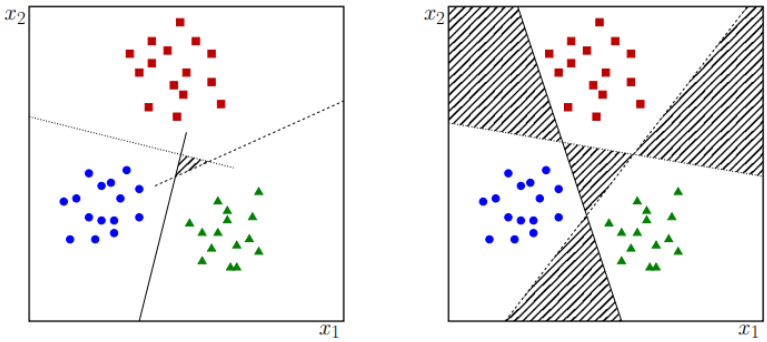
Unlike the above-presented approaches, the data passed into the SVM model in this specific approach has a unique characteristic: the data is not binary classified, but it contains multilevel classifiers. Hence, to establish the classification boundaries in a multi-class problem, various approaches are available in the SVM. One such approach is establishing a decision boundary between each class, considering each class and binary level. For instance, if the classification of the data contains apples, oranges, and mangoes. A binary classification boundary will be established, separating all the mangoes and the records that are not. A similar procedure is carried out between each of the presented classes. Simply put, the data can be split as one belongs to the class and the others do not.

In the case of the classification of various travel modes that are present in the processed data, a Kernelized SVM approach is better suited. Suppose the non-linearly separable data is present in a single dimension; it can be made separable by projecting into a higher dimensional space. This conversion can be a compelling but generalized transformation. A kernel function within a kernelized SVM indicates how the similarity lies between given points when projecting into a different dimensional plane. There are various kernel functions available that are helpful with this separation. The most popular ones are a Radial Basis Function Kernel (RBF) and a polynomial Kernel. According to the functionality of an RBF kernel, the similarity between two points presented in a higher dimension space can be equated to the distance between the vectors and foundational input space in an exponentially decaying function. Whereas the polynomial kernel adds a degree parameter that helps control the complexity of the model and utilizes a polynomial equation for separation. This also directly affects the transformation cost. Various approaches were devised for specifically handling multi-label separations. Some prominent ones are one-to-rest or one vs. all, one-to-one or one vs. one, and non-heuristic approaches. In a one-

to-one approach, a hyperplane was generated between two classes neglecting the other classes in the play, which means the separation happens only between the two chosen classes. Likewise, it is repeated between each pairing. With the one-to-rest approach, the hyperplane separates one class from all the rest of the classes, which divides the selected classes from the rest of all classes by grouping them together and creating a hyperplane. Figure 115 retrieved from Gerrit J. J. van den Burg and Patrick J. F. Groenen (2015) demonstrates the various approaches of linearly non-separable multilevel classification using an SVM.

Figure 115

Demonstration of Multi-Label Classification Approaches



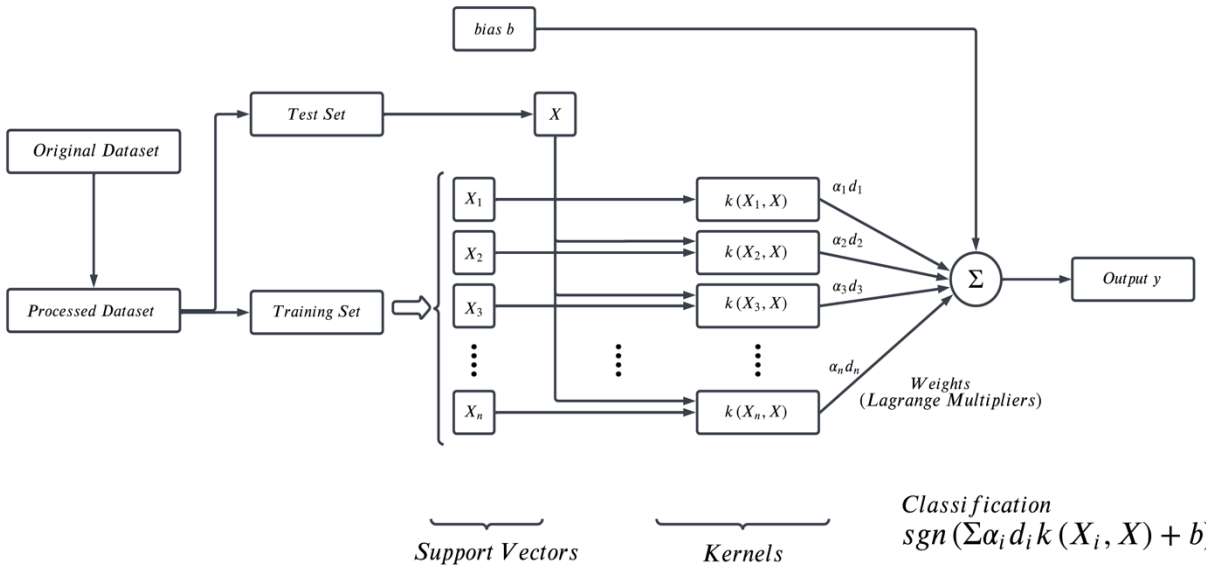
(a) One vs. One (b) One vs. All

Once the boundaries are established, the testing data that falls in the respective boundary lane will be assigned the classification depicted by the data within the boundary range. A detailed depiction of the process and data flow architecture of an SVM model is depicted in figure 18. The original dataset, which passed through various processing phases, was split into training and testing sets. The diagram depicted in figure 116 enhanced from Gu et al. (2009) shows the support vectors drawn from the training set and the input vector X derived from the testing set. The structure depicts the addition of kernels representing the utilization of the kernelized SVM approach since the data chosen for this project involves multi-class

classification. Kernelized SVM works optimally with multilevel classification. The bias and weights part of the SVM working mechanism is also depicted as needed. The weights can help minimize the error. The classification function, which is nominal in the case of the SVM baseline, is also depicted in the pictorial representation. Simply put, the generated support vectors and the established boundaries through kernelized SVM can facilitate the righteous class prediction for the test data that is passed through.

Figure 116

SVM Architecture



Random Forest - Model Architecture and Data Flow

This project aimed to look deep into the algorithm of Random Forest and how this model will help classify the travel mode. The data is obtained after all the processes, such as data cleaning and transformation. The data is next passed to the preparation phase, where it will be split into ratio of 70:30 for training and testing. This will then be fed into the model. This section will cover the model's architecture, the components present in the model, and how the data flows among them to understand better.

The architecture of the Random Forest model will give us an idea of the present layers and the flow of data in them. As covered earlier, the RF model has multiple decision trees, which will create based on a different set of descriptive features. The data from the previous phases are passed to the model; this is the initial step to perform before the modeling phase.

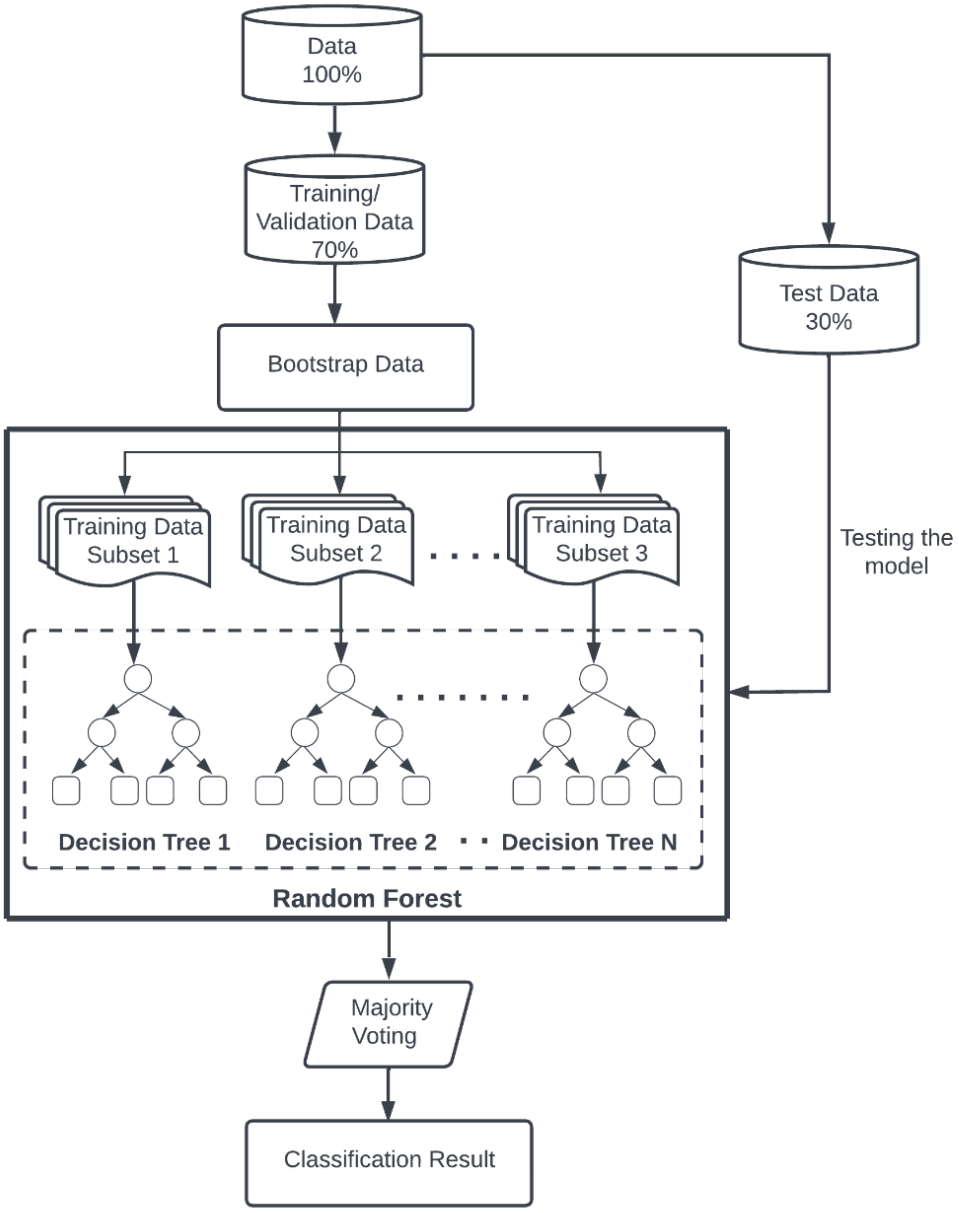
Next step, the data is split into training which has 70% of the data and is also used for validation purposes. The other 30% of the data was used to test the model's performance. Figure 117 was improvised from the paper by Chen et al. (2019) (p.6), which shows the model's architecture and data flow.

The model's training data was passed to a bootstrap function; the data passed to each tree will be a subset of the dataset called a bag. Each tree is trained on a different set of the bag, which is done to ensure the model will be trained on a diverse group so that the model will be generalized better. The majority outcome obtained will finally be optimal. The bags passed to each tree will have a different subset of data and descriptive features.

Each decision tree performs a rule-based approach on these data subsets. The number of the decision tree that can be used can be set in the Random Forest model. This can be used to ensure the model is generalized and avoid overfitting. The tree also has an impurity measure attached, which will help split the convenient feature to gain the best information. The different measures of impurity were covered previously. The Random Forest model will calculate the information gain on the subset of features that were passed to the tree, the feature with the highest information gain will be chosen as the root node. Following this step, the data is further split into two parts the left node and the right node. When moving on to perform the next split, the data will the root node feature will be removed when performing the next split. All these steps are repeated until the leaf node is obtained.

Figure 117

Random Forest Architecture and Data Flow



The hyperparameters can be set to ensure the model is not overfitted so that a new test sample can still be predicted for the outcome. The test data is passed to the trained model built with a set parameter. The testing will predict the multiclass result of the data. Each test record will be passed to multiple decision trees, where the data will be traversed until it reaches a leaf

node. The leaf node it reaches will be taken as the prediction of that test record. This same process will be applied to the other decision tree, and the outcome will be collected. The majority will be taken from each outcome of the decision tree. Figure 120 shows the top few prediction samples obtained for the test record. The prediction results are then passed into evaluation, where these predictions are evaluated to see the model's performance.

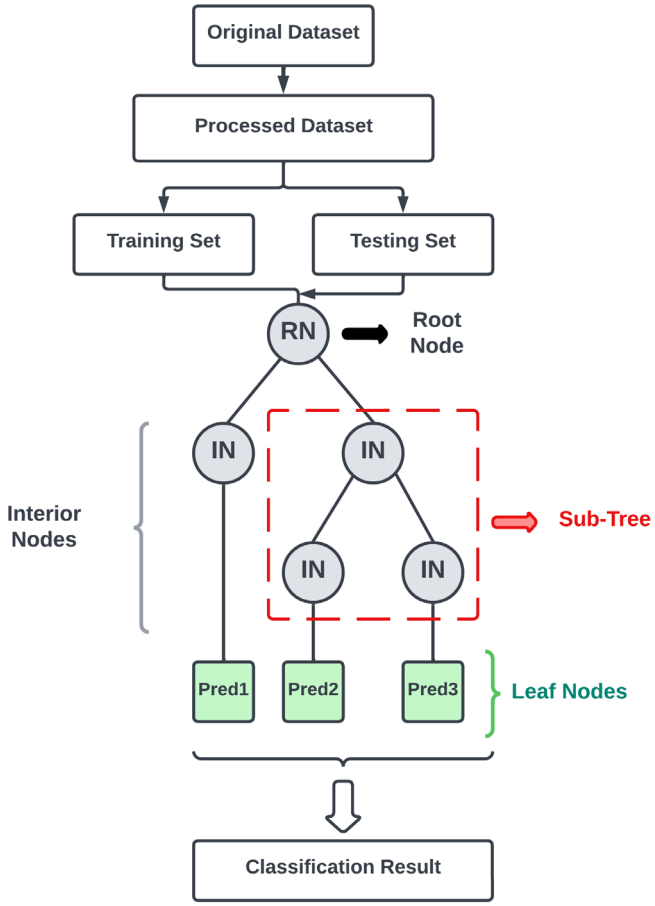
The data flow among these steps, shown in Figure 120, gives an idea of how the data moves from one stage to another. The processes the data goes through before being passed to modeling; the cleaner the data, the more optimal the result obtained from the model. Understanding how the data flows in the model and the architecture can help better understand how the model can be enhanced. The upcoming section will cover the different characteristics of the model and how it can be implemented and further enhanced to obtain optimal prediction results.

Decision Tree - Model Architecture and Data Flow

It is critical to design an efficient architectural plan for the model and associated data flow tasks after analyzing the project's resource requirements in terms of the minimum configuration of hardware, software, tools, and libraries. This section elaborates on the model architecture, describing the components involved as well as how data flows through these components to yield the model's prediction result. Figure 118 illustrates the model architecture of the chosen model in the model proposal section. Figure 118 shows that the first stage was generating appropriate training and test dataset for passing it through the model. As explained in the data processing section, the original dataset was passed through various stages of data cleaning and transformation steps. Then it was prepared to be utilized in the modeling phase. The data was prepared in such a way that the model learning performance became better.

Figure 118

Decision Tree Architecture



The dataset was divided into a 70:30 ratio where 70% was used for training the model and the rest 30% was for testing. The higher proportion of training data ensured that the model was trained better and provided accurate predictions in the testing phase. The train and test data were standardized to avoid any issues due to the mismatch in the scales of different features Using the Scikit-learn StandardScaler function. A cross-validation step was performed to train the model effectively and achieve a generalized model which is free from bias. Additionally, there might have been some mismatch in the proportion of the number of observations or instances assigned to the respective sets. Thus, a stratified k-fold validation step was conducted

where k was five, ensuring that the model would not overfit. Once the data was prepared by following the step mentioned above, it was ready to be passed through the model.

When the data was passed through the decision tree classifier, it predicted the end result in a rule-based decision-making approach, explained in detail in this section. A decision tree structure is depicted in figure 1, and a basic idea about each associated element, such as root node, branches, interior nodes, and leaf nodes, is explained in detail in the same section. This section covers how the chosen decision tree classifier works internally. A decision tree classifier can work for both binary class and multilevel classification. The approach followed for this project was a multilevel classification, as the dataset has different transportation modes in the target feature. A decision tree decides which feature can be selected for splitting the dataset at each node and how the tree grows. The selected features can be either continuous or categorical. If continuous, the dataset was further divided based on a specific threshold value. Now it begs the question of selecting an appropriate node from the feature set. For instance, from the feature set for this project, either maximum speed can be the right candidate for splitting at the root level or maximum acceleration. This decision can be taken by calculating the impurity of the split datasets based on the features. For identifying an appropriate node for splitting, an information gain metric is used. It decides the informativeness of the generated split datasets based on impurity measures. Impurity means how homogenous the split set is. For instance, if the dataset was split based on maximum speed and the impurity of a set generated based on this feature is attributed to the proportion of target class levels it holds. It was considered an impure set if it contained more diverse target levels such as bike, walk, car, airplane, or subway. The impurity can be measured using two different methods: the Gini index and entropy. Table10 depicts the formula for calculating entropy, and also represents the Gini index calculation formula,

respectively, as retrieved from Kelleher et al. (2020). The entropy calculation shows the summation of probabilities of each feature based on the target level. The Gini index shows the misclassification rate of a target feature level; thus, the probability of the feature based on the target level is subtracted from one, as depicted in table 10. The feature which generates a high information gain for the split datasets becomes the viable candidate for being selected as a split node. As shown in table 10 the remainder calculation, and also the information gain formula, respectively, which is retrieved from Kelleher et al. (2020). The information gain is calculated by deducting the remainder entropy from the whole dataset entropy. If a Gini index is used as an impurity measure, then the same information gain calculation formula also applies in that case. This process of calculating the information gain and splitting the dataset by simultaneously growing the tree goes on until it reaches the leaf node with a pure set which depicts the final prediction. However, if the tree grows to select all the possible features in the dataset, it leads to overfitting. Therefore, different settings or approaches can be used to prune the tree and avoid overfitting issues, such as restricting the number of samples, number of leaves, or depth of a tree.

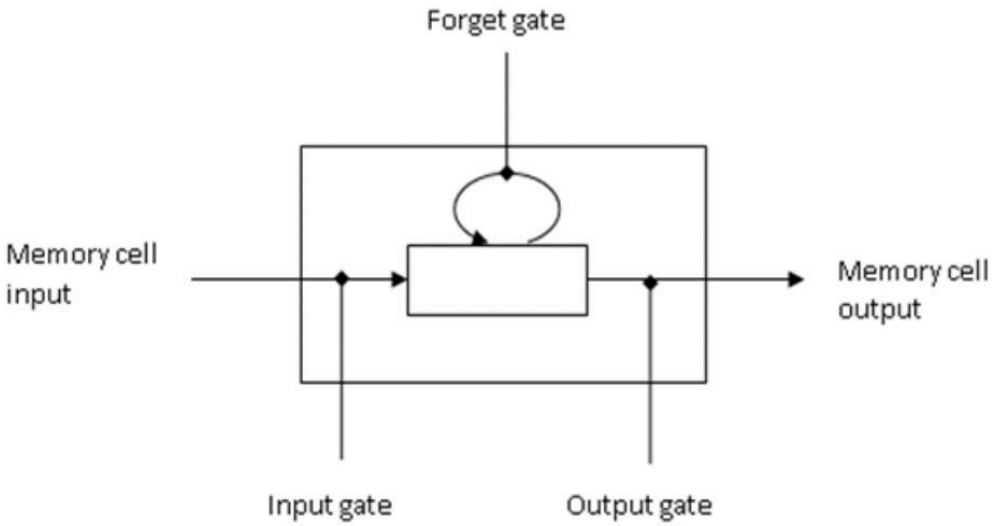
LSTM - ***Model Architecture and Data Flow*** see Appendix

LSTM is made up of LSTM blocks that are linked together. Each block consists of three kinds of analogical gates: the input gate implementing the writing function, the output gate implementing the reading function, and the forget gate related to resetting a cell state or memory. A sigmoid activation function manages these gates, ranging from zero to one. If the value is zero, then the cell state is fully inhibited, and on the other hand, one means a total activation happens. The input gate decides on the state to be retained; the forget gate is readjusted to the current cell state if the cell state value is zero, and the output gate determines whether the cell value should be carried.

Figure 119 portrays the block diagram of an LSTM cell, retrieved from Zaccone et al. (2017) (Optimizing Tensorflow AutoEncoders, Recurrent Neural Network, LSTM networks).

Figure 119

Block Diagram of LSTM Cell

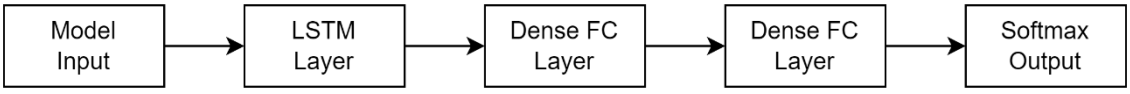


Baseline LSTM. The architecture of this model is a RNN deep learning network consisting of an input LSTM layer, two hidden fully connected layers, and an output Softmax layer. The LSTM, dense variational, and hidden dense fully connected layers have the same number of units to be optimized as model hyper-parameter. The output Softmax layer has a fixed number of nodes based on the number of classes in the data. The data flow in the network depends on whether the LSTM layer implements dropout regularization, a model hyper-parameter, as this will randomly change which nodes participate in network propagation in a particular epoch during training. All nodes in the hidden and output layers will be present as they do not incorporate dropout. In general, the data forward feeds through the network each epoch in the order the layers were built into the model. Each layer performs a nonlinear transformation of the data, with the output layer transforming the data into probabilities. During training, after the data goes through the output layer, loss between the input and output is calculated.

Backpropagation is then performed in which the layers are traversed in reverse order and node weights are updated. Figure 120 illustrates the architecture of the network.

Figure 120

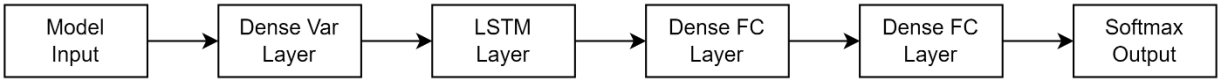
Baseline LSTM Architecture



BInfer-LSTM. The architecture for this model is similar to the baseline LSTM model with the addition of the dense variational layer after the LSTM layer and before the first fully connected layer. This model implements the same LSTM dropout regularization hyper-parameter as the base LSTM. A difference in the data flow is transformation of the input data in the dense variational layer. A latent posterior distribution is created based on the input data and stochastically sampled, creating new data that is forward fed through the network. Loss calculation and backpropagation are performed the same way as the baseline model. Figure 121 illustrates the data flow in BInfer-LSTM model.

Figure 121

BInfer-LSTM Architecture



Model Comparison and Justification

Once a clear understanding regarding the working mechanism of the model is understood, the way the model can be utilized, keeping the requirement in mind, was devised. The model chosen should be put through a detailed building phase, following which the test data will be processed for prediction. The SVM model chosen for this study has to be tested with the set of

isolated testing data, and the model's performance needs to be monitored. It is always recommended to experiment with how the model works with the default setting and track the performance. Following this step, model customization happens by experimenting with a diverse set of permutations for hyperparameters that can be tuned. This section details the comparison between various experimental attempts to find the optimal setup.

Support Vector Machine (SVM)

The first stage of implementation proceeded with the default settings that came pre-tuned. The curiosity to explore and identify the suitable set of parameters that can tune the algorithm's performance even more efficiently laid the groundwork for this second stage. The training and testing phases were carried out one after another. Hyperparameters that were tuned during this phase were the C and gamma parameters. The kernel chosen for this set of execution is RBF, and since RBF does not necessarily use the degree parameter, it was kept out of tuning. After tuning, the values assigned to tuned hyperparameters were 0.5 for gamma and five for C. The gamma parameter is the feature of SVM that designates the distance of influence of a training parameter—assigning a lower value facilitated grouping more parameters together, which helped eliminate a few misclassifications due to the boundaries. The following parameter tuned was C, which added a penalty to each misclassified point. A smaller value of C implies a minor penalty. Increasing the penalty help eliminate the misclassification due to the margin-based boundaries can be minimized. These changes help influence the margins to the optimal level, reducing misclassifications. The hyperparameter tuning does not influence the model's ability to handle diverse data types. The model can still handle various data types, such as text, audio, and image. The compatibility of handling geographical data was also the same as during the default settings. The model is capable of handling both sparse and larger volume datasets. The changes made to

the parameters do not adversely affect the basic capabilities of the model. The possibility of data overfitting or underfitting is always prevalent with any SVM model. The overfitting or underfitting issues identified while examining the model with default parameters were fixed by tuning the C and gamma parameters. An optimal penalty was identified, determining the trade-off between minimizing the training error and model complexity. The model's complexity does not change much from before and stays the same. The multilevel classification and the total number of features and distinct classes still influence the complexity of the model. The complexity of the model does not impact much and stays within the optimal range (see Appendix B for the code of this process).

Random Forest (RF)

Providing a preface on how the random forest mechanism works, Multiple decision tree classifiers are fed with data subsets per the random forest mechanism. The data were then classified with the help of an estimator, an in-built functionality promoting the majority rule and helping increase the accuracy while simultaneously decreasing the overfitting. The training data was initially supplied to the sklearn.ensemble.RandomForestClassifier method of the sklearn library without changing the original parameter to understand how the model worked. The random state was set to 233, which helped in deciding how to keep the training testing set split constantly throughout the execution phase and not fluctuate during different runs. As the number of trees rises, Random Forests' testing performance does not suffer from overfitting. The model had specific parameters set to default to play a role in better tuning the RF model. The process of exploring different permutations of parameter values and identifying the optimal fit is called hyperparameter tuning. The tuned parameter was n_jobs, which was set to two differing from the default setting of none. This helps configure the maximum number of jobs the model performs

parallelly to the maximum of two and ensures that the model complexity is reduced to a certain extent. Setting the number of jobs parameter to two prompts that the work will be divided among the two processors, and the model will take more time due to work happening in parallel.
The next tuned parameter was the max_depth parameter which helped control the tree depth and was set to 10. The max_depth parameter ensured the model did not traverse down to the leaf node crossing the tenth division. RF model tends to spend considerable time training, which can be attributed to the construction of numeral decision trees parallelly. Since the n_jobs parameter was set to limit the total parallel run jobs, this helps improve the model's performance while reducing complexity. The model's strengths and limitations needed to be understood to make a better analysis of the model's efficiency. The primary advance of the model was the preciseness in predicting the travel modes involving multi-label classifiers. Handling the overfitting issue and reducing the parallel jobs and tree depth helped combat the performance issue while reducing the model's complexity. The limitation of the model can be attested due to the varying bias in the chosen dataset since specific modes were more prominently occurring compared to the others. This can be due to the expense factor (see Appendix B for the code of this process).

Decision Tree (DT)

The decision tree algorithm was implemented by exploring the various hyperparameters that can be tuned with the hopes of the resultant model's capability to be better with the tuned setting than the default. This section describes the modified version of the model with a detailed explanation of the various tuned parameters. The first parameter chosen to be tuned from the mix of multiple available ones was the criterion but assigned with the default settings of gini since it is computationally faster than entropy. However, specific hyperparameters were tuned to get a better-performing model. The maximum depth of the tree was changed to four which ensured that the

tree did not grow so much that it could lead to an overfitting issue. Next was the control of the randomness of the model. As the splitting of the features happened randomly, it was essential to comprehend the deterministic behavior of the split. Thus, the default "random_state" value of None was updated to 233. The model still supported data with several formats and types, such as image, audio, video, and tabular. In addition, a large dataset impacts the training time complexity of the model, and thus the selected model performed well with a small dataset size. In general, a decision tree tends to overfit because it recursively iterates over all the features until a pure set is achieved with a homogenous set of class labels. The training time of the model was proportional to the number of dimensions. Moreover, the model's space complexity for training and testing depended on the number of nodes. The model was a supervised machine learning model; computational efficiency was achieved using a CPU with the minimum configuration described in the hardware configuration section instead of any GPU or TPU configuration. The strength of this model was that it was easier to understand and interpret even though additional hyperparameters increased the complexity. Furthermore, the overfitting issue was mitigated to some extent by restricting the maximum depth of the tree. The imbalanced dataset was still a limiting factor as it could cause to create a biased prediction (see Appendix B for the code of this process).

Long-Short Term Memory (LSTM)

The basic architecture of the model remains almost the same, as they differ only in the presence of a dense variational layer. The model takes the same input data, which was transformed differently before the model Softmax output. The baseline LSTM was purely deterministic in design, while the BInfer-LSTM performs a stochastic data transformation by randomly sampling a latent distribution. Theoretically, the BInfer-LSTM model is better equipped for epistemic uncertainty and should perform comparatively better with relatively more minor and sparse data

than the baseline LSTM model. Both models were evaluated for overfitting, which likely had a similar solution to reducing training iterations. Underfitting was addressed in both models by increasing training iterations. However, the BInfer-LSTM model had the advantage of adjusting the hyper-parameter scale of the posterior distribution in the variational layer. Both models were executed with a CPU but showed shorter training time when executed on a GPU in Google Colab. Execution of a single epoch during training took an exponentially shorter duration for the baseline LSTM model and the Execution time of BInfer-LSTM spiked by one second with a 20% increase in CPU time. Both models had similar strengths in leveraging an LSTM layer. The additional strength of the dense variational layer in the BInfer-LSTM presents a weakness that was not in the baseline LSTM model. Additional training parameters, similar in the count to the LSTM layer, adds additional model complexity by increasing the overall parameter count. Finding the proper scale of the posterior probability distribution was not easily deduced and could be challenging to optimize. An improper scale could impede convergence in the model and reduce comparative performance to the baseline model. Another weakness was reproducibility. The model included a stochastic process, and optimized values for hyper-parameters were prompter to be within a tolerable difference metric-wise for two independent model training instantiations. Otherwise, results may be anomalous and decrease model performance and interpretation understanding. (see Appendix B for the code of this process).

Comparison of Model Characteristics

Comparing characteristics between the various models discussed in the previous section clarified how the model could be optimized to attain the desired goal. The description of various characteristics observed during modeling helped recognize how specific models behave under certain circumstances. However, it is always on the menu to look for ways to optimize the results

and make the model perform to its utmost capacity. This was achieved by tuning various parameters, which improved the performing efficiency while not compromising the quality and minimized the misclassification rate. The results of these tuned models were more optimized than how the model performed in default settings. The previous sections showcased these in a detailed manner, and also the passage describes the nuances in the various customizations made, but it is better represented in a tabular format. This section reviews the previously discussed model and differentiates them based on characteristics prominently evident in each. Table 12 compares both models based on these differentiating characteristics, such as the values of the tuned parameters, the evaluation methods used, and the kind of data passed to the model.

Table 12

Comparison of Model Characteristics

Characteristics	SVM	RF	DT	LSTM
Parameter Tuning	Yes	Yes	Yes	Yes
Architecture	Supervised ML	Supervised ML	Supervised ML	RNN +Bayesian Inference
Output	Deterministic	Deterministic	Deterministic	Stochastic
DataType	Text , Geographic Data	Text , Geographic Data	Text , Geographic Data	Text , Geographic Data
Train-Test Ratio	70:30	70:30	70:30	70:30
Standardization	Applied	Applied	Applied	Applied
Random State	233	233	233	NA
Gamma	0.5	NA	NA	NA
C	5	NA	NA	NA
Kernel	rbf	NA	NA	NA
Class Weight	Balanced	NA	NA	NA
N fold	5	5	5	NA
Stratify	Yes	Yes	Yes	NA
Shuffle	True	True	True	NA
Learning Curve Scoring	F1 Weighted	F1 Weighted	F1 Weighted	NA
Cross-Validation	Yes	Yes	Yes	NA
cv	Stratified K Fold	Stratified K Fold	Stratified K Fold	NA
N job	NA	2	NA	NA
Criterion	NA	Entropy	gini	NA
max depth	NA	10	4	NA
CrossVal Scoring	Accuracy	Accuracy	Accuracy	NA
Units	NA	NA	NA	9, 32, 64 and 128
Dropout	NA	NA	NA	0.1, 0.2, 0.3, 0.4 and 0.5
Activation	NA	NA	NA	tanh, softmax
Variational Layer	NA	NA	NA	TRUE
Epistemic Error	NA	NA	NA	Handled
Epoch CPU Time	NA	NA	NA	6 sec
Confusion Matrix	Implemented for Evaluation	Implemented for Evaluation	Implemented for Evaluation	Implemented for Evaluation
Classification Report	Implemented for Evaluation	Implemented for Evaluation	Implemented for Evaluation	Implemented for Evaluation

Model Evaluation Methods

The built models were initially trained and following which the testing phase began. The model trained using the generated training dataset while tested using the test data yield various measures that can help evaluate the model. Different designated approaches exist to assess the model's performance by the time taken and the preciseness while classifying. Some of such evaluation methods will be discussed in this section.

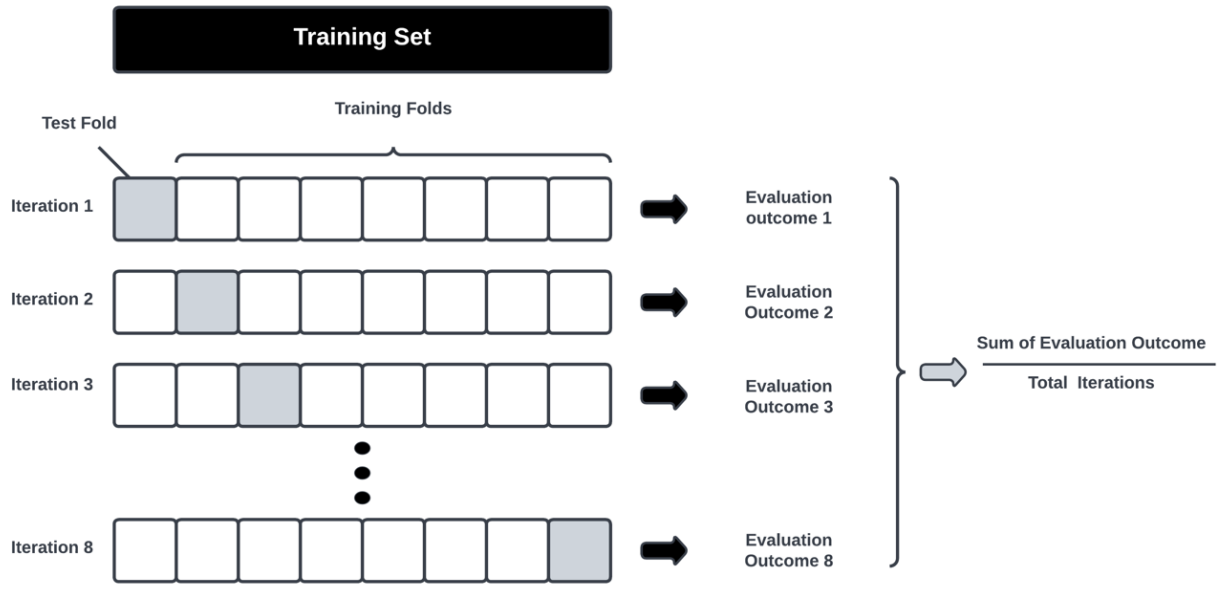
Cross Validation

The primary approach that was chosen for evaluating this model is Cross-Validation. In a broader spectrum, model validation can be explained as the process performed to ensure the model's acceptability in real-world scenarios by simulating it to a possible extent. It helps predict the model's performance under stressful scenarios involving data not used while training. One such approach is Cross-Validation.

Cross-Validation is the process where the training dataset is split into multiple subsets. One of the split subsets will be chosen and held back to act as the testing set while the remaining subsets are used for training. The subset chosen for testing will then be passed through the model trained with the remaining subsets, and the prediction outcome measure is recorded during each iteration. The most commonly used measure is accuracy. Upon completion of all the iterations, the recorded outcomes are aggregated, and the average value is taken. This average value, also called the Cross Validation Score, is displayed as the end result of the Cross-Validation approach. There are many specific techniques in Cross-Validation, such as Hold-Out, k-Fold Cross-Validation, and Stratified k-Fold Cross-Validation. The approach chosen for this project from the various available ones will be explained in the following section. Figure 122 depicts the working methodology of Cross-Validation.

Figure 122

Cross-Validation Methodology



Stratified k-Fold Cross Validation

Cross-Validation explained above gives an introduction to the overall approach itself. This section explains the specific technique chosen, which is Stratified k-Fold Cross-Validation. A short introduction to the k-Fold Cross-Validation is necessary to understand this better. While performing a k-Fold Cross-Validation, the training set is divided into K subsets, and the process is iterated K times. During each attempt, one subset is chosen as a training set while the others are used to train the model. The slice chosen during one iteration was not chosen again in another iteration. This way, the model can be trained using the entire data during the iterative approach, and there is no doubt that any data was getting missed out.

Stratified k-Fold Cross-Validation is an extension to the normal k-Fold Cross-Validation. In this approach, the dataset will first be randomly shuffled, and the training and testing subsets will be isolated with the imbalance of output variables in consideration. For instance, if the target

class labels of the dataset are 80:20 in ratio, there exists an imbalance. A Stratified k-Fold validation ensures that each subset perfectly represents this ratio which is achieved by ensuring that each subset contains the same 80:20 ratio of target class labels. The dataset chosen for this project contains an imbalance between the multiple target class labels. Hence Stratified k-Fold is chosen as the optimal Cross-Validation Approach. Figure 123 depicts the Stratified k-Fold methodology.

Figure 123

Stratified k-Fold Methodology

Measures of Evaluation

Various metrics can be produced as the result of the testing phase using several evaluation methods and help evaluate the model's performance. Accuracy, precision, recall, and F1 measure are some of those metrics. This passage of the model evaluation methods section explains in-depth how those measures are calculated.

Accuracy. The primary metric that helps evaluate the model's performance is accuracy. Accuracy can be defined as the probabilistic measure of the total number of correct classifications divided by the total records classified, ranging from zero to one or 0% to 100%. There is also

another commonly used way to calculate accuracy, which is by using the true positive (TP), true negative (TN), false positive (FP), and false negative (FN), which is depicted in (9) retrieved from Kelleher et al. (2020).

$$Classification\ Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{9}$$

Precision. Precision is a metric used to measure how often the prediction returned by the model is the right one. A higher precision score helps build confidence in the model itself, and precision helps retain the trust that the optimistic prediction made by the model will turn out positive. Often the precision is measured between the range of zero and one. The higher the precision, the better the scope of the model's future implications. The formula to calculate the precision is depicted in (10), retrieved from Kelleher et al. (2020).

$$Precision = \frac{TP}{TP+FP} \tag{10}$$

Recall. The recall is a measure similar to calculating the true positive rate. Calculating the recall helps measure proportionally how many actual positive labels were correctly predicted by the model. Similar to precision, recall often ranges from zero to one—higher recall implies better model performance. The formula to calculate the recall is depicted in (11), retrieved from Kelleher et al. (2020).

$$Recall = \frac{TP}{TP+FN} \tag{11}$$

FI Score. F1-Score is a metric that depicts the accuracy of a model specifically while evaluating using a binary classification system and classifying the values into a positive or negative category. F-Score is an amalgamation of precision and recall, often referred to as the harmonic mean of both. The prominent sectors of its usage are natural language processing and systems like search engines. The F1-Score can also be adjusted, giving prominence to either of the two. The formula to calculate the F1-Score is depicted in (12), retrieved from Kelleher et al. (2020).

$$F1\ Measure = 2\ X\ \frac{(Precision\ X\ Recall)}{(Precision+Recall)} \quad (12)$$

Model Validation and Evaluation

The modeling phase of CRISP-DM was followed by the Evaluation phase, where the built models were validated using the various evaluation methods and metrics discussed in the previous section. Exploring and identifying the reasoning behind the results attained could not only help understand the model better but also help tackle a similar issue that arises in future implementations. This section concentrates on the results of each model separately, followed by a comparison of the results achieved.

Validation of SVM Model

The first step of evaluating the SVM model was to verify the cross-validation score attained by the stratified k-Fold Cross-Validation process. Compared to the previous model, the results after tuning parameters like gamma and c had a minimal positive change, increasing to 80%. Evaluating the other metrics helped attain a clear picture of whether this tuning was optimal. Figure 124 depicts the Cross-Validation score attained by the SVM model with tuned hyperparameters.

Figure 124

Cross Validation Score Attained by SVM Mode

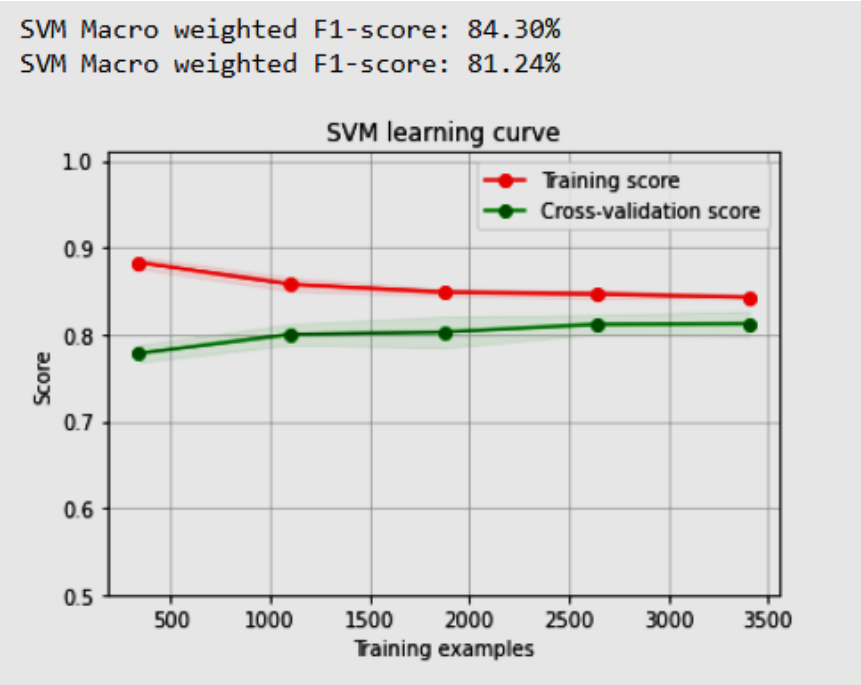
```
Training SVC ...
Average CV performance for SVC : 0.801202 (in 2.14918 seconds)
```

Since the cross-validation did not showcase a very evident spike, the next validated metric was the weighted f1-score. Since the travel mode detection is a multi-label classifier, weight f1-score is the optimal evaluation method which can be described as the mean of the f1-scores attained by individual class labels. The generated scores were plotted. The training score

attained was 84.30%, and the cross-validation score was 81.24 %. Figure 125 depicts the macro-weighted F1-Score attained with the hyperparameter-tuned SVM model during the training and cross-validation.

Figure 125

Learning Curve of SVM Model



The hyperparameter-tuned SVM model was also evaluated using additional measures such as accuracy, precision, recall, and F1-Score. Since the learning curve generation process already covered F1-Score, the other three measures were evaluated by generating a classification report. The classification report generated each class label's precision, recall, and accuracy with the tuned SVM model. The imbalance in the data was the reasoning behind the class label boat returning zero for all the measures. The highest values of the measures were recorded by the travel mode walk followed by bike, which was more commonly found in the dataset. The average accuracy attained by the SVM model was 84% after tuning. Figure 126 depicts the classification report generated by the SVM model with tuned hyperparameters.

Figure 126

Classification Report of SVM Model

	precision	recall	f1-score	support
bike	0.82	0.89	0.85	100
boat	0.00	0.00	0.00	1
bus	0.75	0.82	0.78	169
car	0.78	0.78	0.78	88
subway	0.81	0.77	0.79	57
taxi	0.43	0.11	0.18	27
train	0.75	0.38	0.50	8
walk	0.94	0.96	0.95	298
accuracy			0.84	751
macro avg	0.53	0.47	0.48	751
weighted avg	0.83	0.84	0.83	751

Validation of RF Model

The chosen Random Forest model was hyper-tuned in the modeling phase to see if it would yield better results. The model used five folds of cross-validation and a grading mechanism for accuracy. The model was initially evaluated using this method. The results of cross-validation are displayed in Figure 127. This model had an accuracy of 0.86 and took 3.22822 seconds to train. This increase in the train time was because of running multiple CPUs simultaneously. This indicates how well the machine learning model works with new data. The following evaluation tool was the model learning curve that displayed the results of each cross-validation iteration's score. The model learning curve over the five folds is shown in Figure 128. For training and cross-validation, the macro-weighted F1 score was calculated; the results were 96.42% and 85.15%, respectively. The classification report created to compare the other metric that can be used to assess the model is shown in Figure 129 below. The metrics, including the accuracy attained for the test set, are shown in Figure 129. The reported accuracy was 87%, and precision and weighted F1 scores received 86%. The recall was 87%, which indicates that the model could classify the target feature's classes more accurately.

Figure 127

Cross Validation Result of RF Model

```
Training RandomForestClassifier ...
Average CV performance for RandomForestClassifier : 0.865386 (in 3.22822 seconds)
```

Figure 128

Model Learning Curve of RF Model

```
RF Macro weighted F1-score: 96.42%
RF Macro weighted F1-score: 85.15%
```

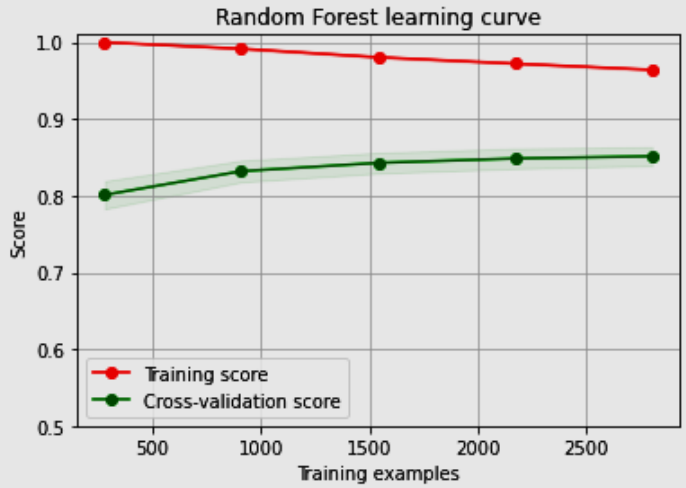


Figure 129

Classification Report of RF Model

	precision	recall	f1-score	support
bike	0.86	0.88	0.87	200
boat	0.00	0.00	0.00	1
bus	0.82	0.86	0.84	339
car	0.77	0.85	0.81	176
motorcycle	0.00	0.00	0.00	1
subway	0.85	0.75	0.80	114
taxi	0.60	0.17	0.26	53
train	0.86	0.38	0.52	16
walk	0.94	0.97	0.96	597
accuracy			0.87	1502
macro avg	0.61	0.49	0.52	1502
weighted avg	0.86	0.87	0.86	1502

Validation of DT Model

This section describes the DT model's statistics with specific hyperparameter tuning. The hyperparameters random_state and max_depth have been added to the default model. The model comparison and justification section go into great detail about the significance of these parameters. The first method to evaluate the model was stratified k-fold cross-validation with the stratification value set to five. Figure 130 depicts the model's average stratified k-fold cross-validation score and the corresponding execution time. It shows that the average score is nearly 0.78, and the execution time is 0.048 seconds. Figure 131 represents the model's learning curve and the macro weighted average f1-score. The macro-weighted f1-score for training was 79.68%, whereas cross-validation was 78.13%.

Figure 130

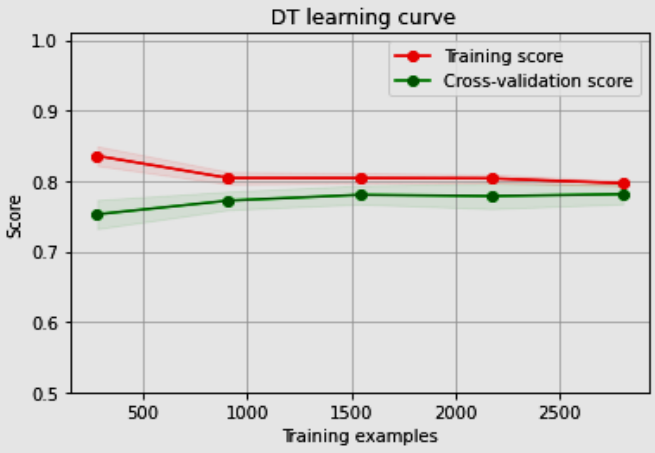
Cross Validation Result of DT Model

```
Training DecisionTreeClassifier ...
Average CV performance for DecisionTreeClassifier : 0.784216 (in 0.0483582 seconds)
```

Figure 131

Model Learning Curve of DT Model

```
DT Macro weighted F1-score: 79.68%
DT Macro weighted F1-score: 78.13%
```



A classification report was used to list the precision, accuracy, and f1-score along with the support for each mode, macro average, and weighted average statistics as it was implemented for the default model. Figure 132 portrays a classification report with detailed statistical results. It shows that the model's accuracy with hyperparameter tuning was 79%. The validation time in the cross-validation shows a significantly less execution time. The classification report shows that the precision and recall value has been decent. After performing these evaluations, a detailed comparison between the models will be provided.

Figure 132

Classification Report of DT Model

	precision	recall	f1-score	support
bike	0.78	0.81	0.79	200
boat	0.00	0.00	0.00	1
bus	0.83	0.76	0.79	339
car	0.51	0.90	0.65	176
subway	0.73	0.45	0.55	114
walk	0.93	0.94	0.94	597
accuracy			0.79	1502
macro avg	0.34	0.35	0.34	1502
weighted avg	0.78	0.79	0.78	1502

Validation of Baseline LSTM Model

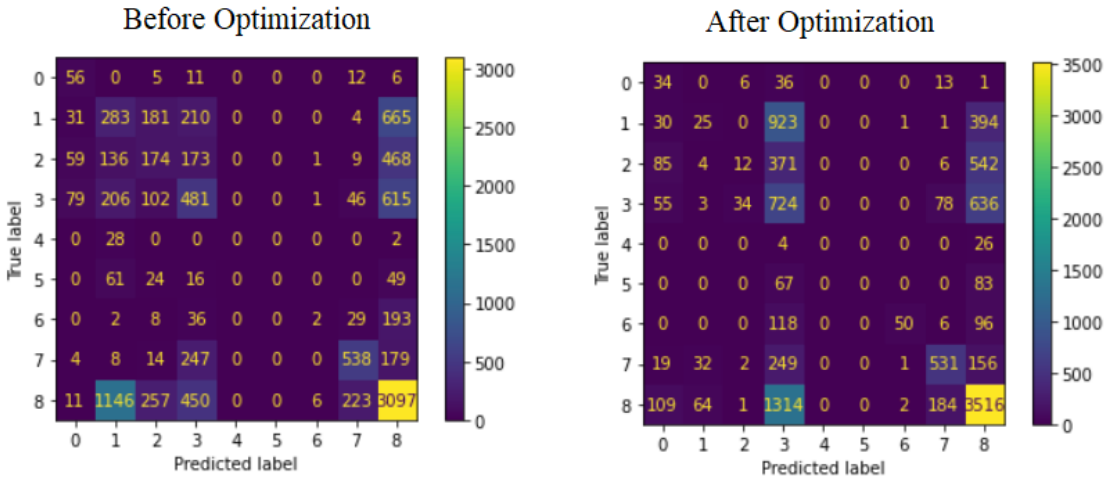
After training the model, predictions were made on the test set containing nine imbalanced classes. Before optimization, the model achieved an accuracy of 0.44 and a weighted average score of 0.43. In seven of the nine classes, scores were less than 0.5, with the remaining two around 0.6. Most classes had similar precision and recall. This model without hyper-parameter tuning exhibits underfitting of the input data with low precision and recall in most classes.

Model hyper-parameters were optimized using random search and resulted in improved accuracy of 0.46, and the weighted score remained the same, 0.43. Weighted average precision and recall improved, as well. The model still exhibited underfitting with two classes with high false positive incidents.

Comparison of classification performance before and after has been visually summarized in figure 133. Before optimization, the model exhibited more uncertainty in predictions with a greater spread in occurrences of misclassifications. The model became more deliberate after optimization, but it needs to be generalized better. These results were interpreted as inferring epistemic uncertainty within the data and were not significantly reduced with model hyper-parameter optimization.

Figure 133

Baseline LSTM Confusion Matrix Comparison



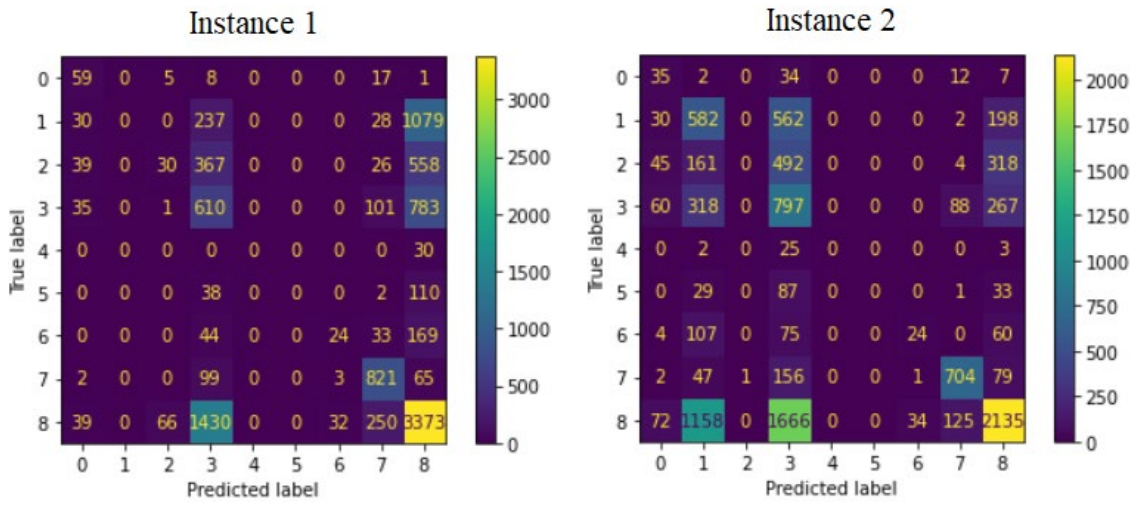
Validation of BInfer-LSTM Model

The BInfer-LSTM model was introduced to address model difficulties with data epistemic uncertainty. The model was executed multiple times with identical hyper-parameters due to stochasticity. Two training instantiations are presented to compare and demonstrate variability in the model predictions. Instance one resulted in an accuracy of 0.4 and a score of 0.41. Instance two resulted in an accuracy of 0.46 and a score of 0.41. the score is a more robust metric for comparing these instances because of the imbalance in class size. Comparing weighted average precision and recall between instances could better explain these scores. The first instance demonstrated higher weighted precision, and the second was a higher weighted recall. An explanation of this behavior was the first instance under-fitted predictions to three classes and the second to two. This behavior should be investigated as improved hyper-parameter optimizations may reduce this behavior and produce more stability in overall prediction performance.

This would improve the reproducibility of model predictions, increasing the validity of the results. Figure 134visualizes the difference in predictions between the two training instantiations.

Figure 134

BInfer-LSTM Confusion Matrix Comparison



Comparison of Results

A comparison of the models will help identify which suited the purpose of this project. Though the previous passages explain the results attained by each model in a detailed manner, it will be easier if represented in a tabular format. Table 13 depicts the comparison of the results attained by both models by evaluating using the various methods.

Table 13

Comparison of Model Characteristics

Model	Weighted F1-Score	Weighted Precision	Weighted Recall	Accuracy
Support Vector Machine (SVM)	0.83	0.83	0.84	0.84
Random Forest (RF)	0.86	0.86	0.87	0.87
Decision Tree (DT)	0.78	0.78	0.79	0.79
Long-Short Term Memory (LSTM)	0.43	0.47	0.46	0.46

Conclusion

Detecting the travel modes can help tackle various ecological and sociological problems. As the initial stages of the project commenced, the chosen dataset was passed through various wrangling, pre-processing, transforming, and preparation stages. These stages' outcomes were a dataset pruned of inconsistencies, which was then passed on to the built machine learning model. The approach proposed to detect the travel modes utilized various models such as RF, DT, SVM, and LSTM with tuned parameters. The results were evaluated using various measures such as accuracy, precision, recall, and f1-score. The evaluation also included the Stratified k-Fold Cross-Validation approach. The hyperparameter tuning results showcase a positive spike portraying that the implementation has worked optimistically. The best-suited model was identified as Random forest with an accuracy of 87%, followed by SVM with an accuracy of 84%. LSTM was least suited since it requires properly streamlined time-series data, and the extraction of additional features compromised this. Certain realizations gave insight into where the concentration for future works lies. Some of the limitations and the future scope of the project will be discussed in this section.

Limitations

Every project or research work has its own set of benefits and shortcomings. Understanding the limitation of the project can help envision where the future scope of the project implementation lies. Some of the limitations of this research work are discussed in this section. One of the primary limitations can be attributed to the imbalance in the number of records collected for various travel modes. The users preferred some travel modes during data collection, which resulted in a bias. The second limitation was the cost since some of the travel modes in the multilabel classifier dataset were expensive. The third limitation of the project was

the need for differentiation between modes with similar characteristics. For instance, buses and subways have similar speed and acceleration characteristics, due to which the model struggles to differentiate. Addressing these limitations can help increase the efficiency of the model.

Future Scope

Understanding the limitations can be advantageous to realize the project's future direction—some of the sectors where the future scope of the project lies are discussed in this section. First and foremost, the project can be expanded to utilize real-time GPS data since the work here mainly concentrated on travel mode prediction using already collected data. Predicting using real-time data would be both interesting and challenging. In the previous section, the limitation of the project was identified as struggling to classify modes with similar characteristics; the future scope of the project can also include the possibility of finding a new approach to tackle this. The project can also be expanded to the research areas such as city planning and targeted marketing, as identification of the most preferred modes of transportation in a specific area can help find ways to shift the population from using private transport to public transport. Understanding the region's travel patterns can also be helpful in targeted marketing since it can help plan which region to target for a specific product. The future scope of the project remains optimal in various research areas.

References

Ahmad, I. (n.d.). *40 Algorithms Every Programmer Should Know*. O'Reilly Online Learning. https://www.oreilly.com/library/view/40-algorithms-every/9781789801217/0f45c87a-818c-4e7c-9250-527fc3f7a250.xhtml

Asad, S. M., Dashtipour, K., Rais, R. N. B., Hussain, S., Abbasi, Q. H., & Imran, M. A. (2021, November 4). Employing Machine Learning for Predicting Transportation Modes Under the COVID-19 Pandemic: A Mobility-Trends Analysis. *2021 International Conference on UK-China Emerging Technologies (UCET)*. https://doi.org/10.1109/ucet54125.2021.9674960

Asci, G., Guvensan, M. A. (2019). A novel input set for LSTM-based transport mode selection. *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, 107-112. doi: 10.1109/PERCOMW.2019.8730799

Asgari, F., & Clemencon, S. (2018). Transport Mode Detection when Fine-grained and Coarse-grained Data Meet. *2018 3rd IEEE International Conference on Intelligent Transportation Engineering (ICITE)*. https://doi.org/10.1109/icite.2018.8492673

Bednjanec, A., & Tretinjak, M. F. (2013, May). Application of Gantt charts in the educational process. *2013 36th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO).* https://ieeexplore.ieee.org/abstract/document/6596335

Bin Altaf, M. A., & Yoo, J. (2016). A 1.83 J/Classification, 8-Channel, Patient-Specific Epileptic Seizure Classification SoC Using a Non-Linear Support Vector Machine. *IEEE Transactions on Biomedical Circuits and Systems, 10*(1), 49–60. https://doi.org/10.1109/tbcas.2014.2386891

Brunauer, R., Hufnagl, M., Rehrl, K., & Wagner, A. (2013). Motion pattern analysis enabling accurate travel mode detection from GPS data only. *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. https://doi.org/10.1109/itsc.2013.6728265

Bonaccorso, G. (2020). *Mastering Machine Learning Algorithms: Expert techniques for implementing popular machine learning algorithms, fine-tuning your models, and understanding how they work, 2nd Edition*. Packt Publishing.

Chen, Yu, Wen, Zhang, Yin, Wu, & Yao. (2019). Pre-evacuation Time Estimation Based Emergency Evacuation Simulation in Urban Residential Communities. *International Journal of Environmental Research and Public Health*, *16*(23), 4599. https://doi.org/10.3390/ijerph16234599

Cortes, C., & Vapnik, V. (1995, September). Support-vector networks. *Machine Learning, 20*(3), 273–297. https://doi.org/10.1007/BF00994018

Dai, S., Li, L., & Li, Z. (2019). Modeling Vehicle Interactions via Modified LSTM Models for Trajectory Prediction. *IEEE Access*, *7*, 38287–38296. https://doi.org/10.1109/access.2019.2907000

Dua, R., & Ghotra, M. S. (2018). *Keras Deep Learning Cookbook: Over 30 recipes for implementing deep neural networks in Python*. Packt Publishing.

F. de S. Soares, E., Revoredo, K., Baião, F., A. de M. S. Quintella, C., & V. Campos, C. A. (2019, December). A Combined Solution for Real-Time Travel Mode Detection and Trip Purpose Prediction. *IEEE Transactions on Intelligent Transportation Systems*, *20*(12), 4655–4664. https://doi.org/10.1109/tits.2019.2905601

F. de S. Soares, E., Salehinejad, H., Campos, C. A. V., & Valaee, S. (2019, December). Recurrent Neural Networks for Online Travel Mode Detection. *2019 IEEE Global Communications Conference (GLOBECOM)*, *22*(9), 5473–5485. https://doi.org/10.1109/globecom38437.2019.9013316

F. de S. Soares, E., Quintella, C. A. D. M. S., & Campos, C. A. V. (2021, February). Smartphone-Based Real-Time Travel Mode Detection for Intelligent Transportation Systems. *IEEE Transactions on Vehicular Technology*, *70*(2), 1179–1189. https://doi.org/10.1109/tvt.2021.3055413

Fang, S. H., Liao, H. H., Fei, Y. X., Chen, K. H., Huang, J. W., Lu, Y. D., & Tsao, Y. (2016). Transportation modes classification using sensors on smartphones. *Sensors*, 16(8), 1324. doi: 10.1109/TITS.2015.2405759

Fortunato, M., Blundell, C., & Vinyals, O. (2017). *Bayesian Recurrent Neural Networks*, arXiv, https://doi.org/10.48550/arxiv.1704.02798.

Gao, L., Chen, X., Zhu, Z., & Chang, T. H. (2020, September). Effectiveness of Public Transport Networks in Motorized Mode Detection: A Case Study of a Planning Survey in Nanjing. *2020 IEEE 5th International Conference on Intelligent Transportation Engineering (ICITE)*, *22*(9), 5473–5485. https://doi.org/10.1109/icite50838.2020.9231462

Gerrit J. J. van den Burg, & Patrick J. F. Groenen. (2015). GenSVM: a generalized multiclass support vector machine. *Journal of Machine Learning Research*, *17*(1), 7964–8005.

Ghosh, S., Dasgupta, A., & Swetapadma, A. (2019). A Study on Support Vector Machine based Linear and Non-Linear Pattern Classification. *2019 International Conference on Intelligent Sustainable Systems (ICISS)*. https://doi.org/10.1109/iss1.2019.8908018

Global Public Transportation Market Size Report, 2028. (n.d.). Retrieved September 24, 2022, from https://www.grandviewresearch.com/industry-analysis/public-transportation-market-report#:%7E:text=The%20global%20public%20transportation%20market%20is%20expected%20to%20grow%20at,USD%20309.1%20billion%20by%202028.

Gu, I. Y. H., Andersson, H., & Vicen, R. (2009c). Wood defect classification based on image analysis and support vector machines. *Wood Science and Technology, 44*(4), 693–704. https://doi.org/10.1007/s00226-009-0287-9

Gulshad, S., Sigmund, D., & Kim, J. H. (2017). Learning to reproduce stochastic time series using stochastic LSTM. *2017 International Joint Conference on Neural Networks (IJCNN)*, 859-866. doi: 10.1109/IJCNN.2017.7965942.

Hochreiter, S., & Schmidhuber, J. (1997, November). Long Short-Term Memory. *Neural Computation*, 9(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Ho, T. K. (1995, August). Random Decision Forests. *Proceedings of the 3rd International Conference on Document Analysis and Recognition*. https://doi.org/10.1109/ICDAR.1995.598994

Iskanderov, J. & Guvensan, M. A. (2020). Breaking the limits of transportation mode detection: Applying deep learning approach with knowledge-based features. *IEEE Sensors Journal, 20*(21), 12871-12884. doi: 10.1109/JSEN.2020.3001803

Jahangiri, A., & Rakha, H. A. (2015). Applying Machine Learning Techniques to Transportation Mode Recognition Using Mobile Phone Sensor Data. *IEEE Transactions on Intelligent Transportation Systems, 16(5),* 2406–2417. https://doi.org/10.1109/tits.2015.2405759

Ji, Y., Wang, L., Wu, W., Shao, H., & Feng, Y. (2020). A Method for LSTM-Based Trajectory Modeling and Abnormal Trajectory Detection. *IEEE Access*, *8*, 104063–104073. https://doi.org/10.1109/access.2020.2997967

Kelleher, J. D., Namee, B. M., & D'Arcy, A. (2015). *Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies (The MIT Press)* (1st ed.). The MIT Press.

Kumar, A., Alsadoon, A., Prasad, P. W. C., Abdullah, S., Rashid, T. A., Pham, D. T. H., & Nguyen, T. Q. V. (2021). Generative adversarial network (GAN) and enhanced root mean square error (ERMSE): deep learning for stock price movement prediction. *Multimedia Tools and Applications*, *81*(3), 3995–4013. https://doi.org/10.1007/s11042-021-11670-w

Lazar, A., Ballow, A., Jin, L., Spurlock, C. A., Sim, A., & Wu, K. (2019). Machine Learning for Prediction of Mid to Long Term Habitual Transportation Mode Use. *2019 IEEE International Conference on Big Data (Big Data)*. https://doi.org/10.1109/bigdata47090.2019.9006411

Lam, M. W. Y., Chen, X., Hu, S., Yu, J., Liu, X., & Meng, H. (2019). Gaussian process LSTM recurrent neural network language models for speech recognition. *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 7235-7239. doi: 10.1109/ICASSP.2019.8683660.

Li, J., Pei, X., Wang, X., Yao, D., Zhang, Y., & Yue, Y. (2021). Transportation mode identification with GPS trajectory data and GIS information. *Tsinghua Science and Technology*, *26*(4), 403–416. https://doi.org/10.26599/tst.2020.9010014

Lin, L., Borlaug, B., Lin, L., Holden, J., & Gonder, J. (2021, April). Identifying Light-Duty Vehicle Travel from Large-Scale Multimodal Wearable GPS Data with Novelty Detection Algorithms. *2021 IEEE Green Technologies Conference (GreenTech)*. https://doi.org/10.1109/greentech48523.2021.00034

Liu, H., & Lee, I. (2017). End-to-end trajectory transportation mode classification using Bi-LSTM recurrent neural network. *2017 12th International Conference on Intelligent Systems and Knowledge Engineering (ISKE)*. https://doi.org/10.1109/iske.2017.8258799

Lu, S., & Xia, Y. (2020, September 23). Dual Supervised Autoencoder Based Trajectory Classification Using Enhanced Spatio-Temporal Information. *IEEE Journals & Magazine | IEEE Xplore*. Retrieved September 25, 2022, from https://ieeexplore.ieee.org/document/9204708/

Microsoft. (n.d.). GeoLife GPS Trajectories. *Microsoft Downloads*. Retrieved September 27, 2022, from https://www.microsoft.com/en-us/download/details.aspx?id=52367&from=https%3A%2F%2Fresearch.microsoft.com%2Fen-us%2Fdownloads%2Fb16d359d-d164-469e-9fd4-daa38f2b2e13%2F

Nghiem, T. L., Le, V. D., Le, T. L., Maréchal, P., Delahaye, D., & Vidosavljevic, A. (2022). Applying Bayesian inference in a hybrid CNN-LSTM model for time-series prediction. *2022 International Conference on Multimedia Analysis and Pattern Recognition (MAPR)*, 1-6. doi: 10.1109/MAPR56351.2022.9924783.

NHTSA Media. (2022, September 19). NHTSA Early Estimates Show Overall Increase in Roadway Deaths in First Half of 2022, Second Quarter 2022 Projects First Decline Since 2020. *NHTSA*. Retrieved September 24, 2022, from https://www.nhtsa.gov/press-releases/early-estimates-traffic-fatalities-first-half-

2022#:%7E:text=The%20preliminary%20data%20from%20the,same%20time%20period%20last%20year.

Patil, V., Shivam, B. P., & Pradeep, K. A. (2019, September 1). Geosecure-O: A Method for Secure Distance Calculation for Travel Mode Detection using Outsourced GPS Trajectory Data. *IEEE Conference Publication | IEEE Xplore*. Retrieved September 25, 2022, from https://ieeexplore.ieee.org/document/8919474/

Raktrakulthum, P., & Netramai, C. (2017). Vehicle classification in congested traffic based on 3D point cloud using SVM and KNN. *2017 9th International Conference on Information Technology and Electrical Engineering (ICITEE)*. https://doi.org/10.1109/iciteed.2017.8250451

Rasmus Pedersen. (2006). An Embedded Support Vector Machine. *2006 International Workshop on Intelligent Solutions in Embedded Systems.* https://doi.org/10.1109/wises.2006.237155

Rezaie, M., Patterson, Z., Yu, J., & Yazdizadeh, A. (2017, September). Semi-Supervised travel mode detection from smartphone data. *2017 International Smart Cities Conference (ISC2)*, *19*(5), 1547–1558. https://doi.org/10.1109/isc2.2017.8090800

Saini, R., Kumar, P., Roy, P. P., & Dogra, D. P. (2017). An efficient approach for trajectory classification using FCM and SVM. *2017 IEEE Region 10 Symposium (TENSYMP)*. https://doi.org/10.1109/tenconspring.2017.8070076

Safitri, D. M., & Surjandari, I. (2017). Travel mode switching prediction using decision tree in Jakarta greater area. *2017 International Conference on Information Technology Systems and Innovation (ICITSI).* https://doi.org/10.1109/icitsi.2017.8267951

Servello, M., & Evans, M. W. (2002, January 15). Work Breakdown Structure. *Encyclopedia of Software Engineering*. https://doi.org/10.1002/0471028959.sof380

Shalev-Shwartz, S. & Ben-David, S. (2014). *Understanding machine learning: From theory to algorithms*. Cambridge University Press. 244.

Singh, S. K. (2009, August). Database Systems: Concepts, Design and Applications. *Pearson India.* https://learning.oreilly.com/library/view/database-systems-concepts/9788177585674/

Song, X., Kanasugi, H., & Shibasaki, R. (2016). Deeptransport: Prediction and simulation of human mobility and transportation mode at a citywide level. *IJCAI*, 2618–2624. doi: 10.5555/ 3060832.3060987.

Su, X., Caceres, H., Tong, H., & He, Q. (2016, October). Online Travel Mode Identification Using Smartphones With Battery Saving Considerations. *IEEE Transactions on Intelligent Transportation Systems*, *17*(10), 2921–2934. https://doi.org/10.1109/tits.2016.2530999

Tišljari´, L., Cvetek, D., Vareskic, V., & Greguric, M. (2021). Classification of Travel Modes from Cellular Network Data Using Machine Learning Algorithms. *2021 International Symposium ELMAR*. https://doi.org/10.1109/elmar52657.2021.9550817

Wang, B., Gao, L., & Juan, Z. (2018, May). Travel Mode Detection Using GPS Data and Socioeconomic Attributes Based on a Random Forest Classifier. *IEEE Transactions on Intelligent Transportation Systems*, *19*(5), 1547–1558. https://doi.org/10.1109/tits.2017.2723523

Wang, C., Luo, H., Zhao, F., & Qin, Y. (2021, September). Combining Residual and LSTM Recurrent Networks for Transportation Mode Detection Using Multimodal Sensors Integrated in Smartphones. *IEEE Transactions on Intelligent Transportation Systems*, *22*(9), 5473–5485. https://doi.org/10.1109/tits.2020.2987598

Wang, Z., Yang, Y., Yao, E., Zhuang, H., & Li, Y. (2020, September 1). Tourism Travel Mode Identification Based on Cell Phone Signaling Data. *IEEE Conference Publication | IEEE Xplore.* Retrieved September 26, 2022, from https://ieeexplore.ieee.org/document/9231333/

Wang, J., Yuan, Y., Ni, T., Ma, Y., Liu, M., Xu, G., & Shen, W. (2020, March 1). Anomalous Trajectory Detection and Classification Based on Difference and Intersection Set Distance. *IEEE Journals & Magazine | IEEE Xplore*. Retrieved September 25, 2022, from https://ieeexplore.ieee.org/document/8963673/

Xiao, G., Cheng, Q., & Zhang, C. (2019, August). Detecting Travel Modes Using Rule-Based Classification System and Gaussian Process Classifier. *IEEE Access*, *7*(10), 116741–116752. https://doi.org/10.1109/access.2019.2936443

Xu, W., Feng, X., Wang, J., Luo, C., Li, J., & Ming, Z. (2019). Energy Harvesting-Based Smart Transportation Mode Detection System via Attention-Based LSTM. *IEEE Access*, *7*, 66423–66434. https://doi.org/10.1109/access.2019.2918555

Yi Hou, Edara, P., & Sun, C. (2014). Modeling Mandatory Lane Changing Using Bayes Classifier and Decision Trees. *IEEE Transactions on Intelligent Transportation Systems,* 15(2), 647–655. https://doi.org/10.1109/tits.2013.2285337

Zaccone, G., Karim, M. R., & Menshawy, A. (2017). *Deep Learning with TensorFlow: Explore neural networks with Python*. Packt Publishing.

Zhao, Z., Emami, N., Santos, H., Pacheco, L., Karimzadeh, M., Braun, T., Braud, A., Radier, B., & Tamagnan, P. (2022). Reinforced-LSTM Trajectory Prediction-Driven Dynamic Service Migration: A Case Study. *IEEE Transactions on Network Science and Engineering*, *9*(4), 2786–2802. https://doi.org/10.1109/tnse.2022.3169786

Zhu, L., Borlaug, B., Lin, L., Holden, J., & Gonder, J. (2021, April). Identifying Light-Duty Vehicle Travel from Large-Scale Multimodal Wearable GPS Data with Novelty Detection Algorithms. *2021 IEEE Green Technologies Conference (GreenTech)*. https://doi.org/10.1109/greentech48523.2021.00034

Zhu, M., Li, X., & Li, J. (2010, December 1). Travel Mode Choice of Residents in Spring Festival Based on Neural Network. *IEEE Conference Publication | IEEE Xplore*. Retrieved September 26, 2022, from https://ieeexplore.ieee.org/document/5709476/

Appendix A

Data Processing

```python
import os
import pandas as pd
import csv
from os import listdir
import time

import os
# traverse whole directory
Data_files = []
for root, dirs, files in os.walk(r'Data'):
    # select file name
    for file in files:
        # check the extension of files
        if file.endswith('.txt'):
            # print whole path of files
            root = root.split("/")[1]
            Data_files.append(os.path.join(root))

f = open('data.csv', 'w', encoding='utf-8', newline="")
headers = ['People_Num', 'Time', 'Travel Start Time',
           'Travel End Time', 'Lat', 'Lon', 'Alt', 'Transportation Mode']
csv_writer = csv.writer(f)
csv_writer.writerow(headers)

pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

start_time = time.time()

for m in range(len(Data_files)):
    #print(Data_files[m])
    if(Data_files[m] not in [".DS_Store",".ipynb_checkpoints"]):
        trajectory_path = r"Data/"+Data_files[m]+"/Trajectory"
        Trajectory_files = listdir(trajectory_path)
        #print(Trajectory_files)
        for n in range(len(Trajectory_files)):
            path_traj = r"Data/"+Data_files[m]+"/Trajectory/"+Trajectory_files[n]
            path_label = r"Data/"+Data_files[m]+"/labels.txt"
            labels = pd.read_table(path_label, sep='\t')
            trajectory = pd.read_table(path_traj, header=None, sep=',', skiprows=6)
            size_labels = len(labels)
            size_trajectory = len(trajectory)

            labels['Start Time'] = pd.to_datetime(labels['Start Time'])
            labels['End Time'] = pd.to_datetime(labels['End Time'])

            count = 0
            i = 0
            while i < size_labels - 1:
                if (labels['Start Time'][i + 1] - labels['End Time'][i]).seconds == 1 \
                    and labels['Transportation Mode'][i] == labels['Transportation Mode'][i + 1]:
                    labels['End Time'].at[i] = labels['End Time'].at[i + 1]
                    labels = labels.drop(labels.index[i + 1]).reset_index(drop=True)
                    count = count + 1
                    i = i - 1
                i = i + 1
                if i == size_labels - count - 1:
                    break

            size_labels = len(labels)

            trajectory['Time'] = trajectory[5] + ' ' + trajectory[6]
            trajectory = trajectory.drop(columns=[2, 4, 5, 6], axis=1)
            trajectory.columns = ['Lat', 'Lon', 'Alt', 'Time']
            trajectory['Time'] = pd.to_datetime(trajectory['Time'])
            origin_j = 0
            origin_i = 0

            for i in range(origin_i, size_labels):
                for j in range(origin_j, size_trajectory):
                    if labels['Start Time'][i] <= trajectory['Time'][j] <= labels['End Time'][i]:
                        csv_writer.writerow([Data_files[m], trajectory['Time'][j], labels['Start Time'][i],
                                             labels['End Time'][i], trajectory['Lat'][j], trajectory['Lon'][j],
                                             trajectory['Alt'][j], labels['Transportation Mode'][i]])
                        origin_j = j
                        origin_i = i
f.close()

end_time = time.time()
time_taken = end_time - start_time
print(end='\n')
print(f'Total time taken: {time_taken}')
```

Figure A1. The code depicted above showcases how the chosen .plt files were merged and converted into a .csv file.

```
start_time = time.time()

pd.set_option('display.max_columns', None)

pd.set_option('display.max_rows', None)

travel_count = 1

data['Travel Count'] = 1

data['Time'] = pd.to_datetime(data['Time'])

count = []
```

Remove records which suggest Different modes during same time period

```
# For each time point of track data, if it corresponds to multiple modes of transportation,
#all of them will be deleted

for i in range(len(data)-1):

    if (data['Time'][i + 1] - data['Time'][i]).seconds == 0:

        count.append(i)

        count.append(i+1)

data = data.drop(count).reset_index(drop=True)

count.clear()
```

Remove abnormal latitude and longitude

```
# Delete data with abnormal latitude and longitude

for i in range(len(data)):

    if data['Lat'][i] > 90 or data['Lat'][i] < 0 or data['Lon'][i] > 180 or data['Lon'][i] < -180:

        count.append(i)

data = data.drop(count).reset_index(drop=True)
```

Splitting travel segments

```
for i in range(len(data)):
    data['Travel Count'].at[i] = travel_count
    if (i != len(data)-1):
        if ((data['Travel Start Time'][i] != data['Travel Start Time'][i+1]) or \
            ((data['Time'][i + 1] - data['Time'][i]).seconds > 1800)):

                travel_count = travel_count + 1

result = pd.value_counts(data['Travel Count'])
```

```
end_time = time.time()

print(f'Time taken for processing the data:\t {end_time - start_time}')
```

Figure A2. The code depicted above showcases the preprocessing steps performed to eliminate inconsistencies at the foundational level. Some of the operations performed are elimination of records with different travel modes starting simultaneously and abnormal latitude and longitude values.

```
time_gap = []

distance_gap = []

speed = []

acceleration = []

total_time = []

total_time_count = 0

total_distance = []

total_distance_count = 0
```

```
i = 0
count = []
for i in range(0,len(data)-1):
    if (data['Travel Count'][i] == data['Travel Count'][i+1])  and \
    ((data['Time'][i+1] - data['Time'][i]).seconds < 50):
        count.append(i+1)
        data['Time'].at[i+1] = data['Time'].at[i]
data = data.drop(count).reset_index(drop=True)
```

```
for i in range(len(data)-1):

    if data['Travel Count'][i] == data['Travel Count'][i+1]:
        time_gap.append((data['Time'][i + 1] - data['Time'][i]).seconds)
        distance_gap.append(geodesic((data['Lat'][i], data['Lon'][i]), (data['Lat'][i + 1], data['Lon'][i + 1])).m)
    else:
        time_gap.append('N.A')
        distance_gap.append('N.A')

time_gap.append('NA')
distance_gap.append('N.A')
```

```
for i in range(len(data)-1):
    if time_gap[i] != 'N.A':
        speed.append(round(distance_gap[i]/time_gap[i], 2))
    else:
        speed.append('N.A')
speed.append('N.A')
```

```
for i in range(len(data)-1):
    if speed[i] != 'N.A' and speed[i+1] != 'N.A':
        acceleration.append(round(((speed[i+1]-speed[i])/time_gap[i]), 2))
    else:
        acceleration.append('N.A')
acceleration.append('N.A')
```

```
for i in range(len(data)-1):
    if time_gap[i] != 'N.A':
        total_time_count = total_time_count + time_gap[i]
        total_time.append('N.A')
        total_distance_count = total_distance_count + distance_gap[i]
        total_distance.append('N.A')
    else:
        total_time.append(total_time_count)
        total_distance.append(total_distance_count)
        total_time_count = 0
        total_distance_count = 0
total_time.append('N.A')
total_distance.append('N.A')
```

```
data['Time Gap(s)'] = time_gap
data['Distance(m)'] = distance_gap
data['Speed(m/s)'] = speed
data['Acceleration(m/s^2)'] = acceleration
data['Total Time(s)'] = total_time
data['Total Distance(m)'] = total_distance
```

Figure A3. The code depicted above showcases process of extracting additional features representing quantifiable information. Some of the extracted features are speed, distance and time gap.

```
data_feature = pd.DataFrame(columns=['Travel Count', 'Transportation Mode', 'Max Speed(m/s)', '95% Speed(m/s)',
                                     '75% Speed(m/s)', 'Mean Speed(m/s)', 'Speed Std', 'Max Acceleration(m/s^2)',
                                     '95% Acceleration(m/s^2)', '75% Acceleration(m/s^2)',
                                     'Mean Acceleration(m/s^2)', 'Acceleration Std', 'Non 0 Mean Speed(m/s)',
                                     'Non 0 Mean Acceleration(m/s^2)', 'Total Time(s)', 'Total Distance(m)'])
```

Fetaure extraction criterias for speed

```
count_speed = []

for i in range(len(data)):
    if data['Transportation Mode'][i] == 'train' or data['Transportation Mode'][i] == 'subway':
        if data['Speed(m/s)'][i] != 'N.A':
            if float(data['Speed(m/s)'][i]) > 180:
                count_speed.append(i)
    if data['Transportation Mode'][i] == 'taxi' or data['Transportation Mode'][i] == 'bus' \
            or data['Transportation Mode'][i] == 'car':
        if data['Speed(m/s)'][i] != 'N.A':
            if float(data['Speed(m/s)'][i]) > 45:
                count_speed.append(i)
    if data['Transportation Mode'][i] == 'walk':
        if data['Speed(m/s)'][i] != 'N.A':
            if float(data['Speed(m/s)'][i]) > 5:
                count_speed.append(i)
    if data['Transportation Mode'][i] == 'bike':
        if data['Speed(m/s)'][i] != 'N.A':
            if float(data['Speed(m/s)'][i]) > 10:
                count_speed.append(i)

data = data.drop(count_speed).reset_index(drop=True)
```

Fetaure extraction criterias for acceleration

```
count_speed_a = []

for i in range(len(data)):
    if data['Transportation Mode'][i] == 'train' or data['Transportation Mode'][i] == 'subway':
        if data['Acceleration(m/s^2)'][i] != 'N.A':
            if float(data['Acceleration(m/s^2)'][i]) > 8:
                count_speed_a.append(i)
    if data['Transportation Mode'][i] == 'taxi' or data['Transportation Mode'][i] == 'bus' \
            or data['Transportation Mode'][i] == 'car':
        if data['Acceleration(m/s^2)'][i] != 'N.A':
            if float(data['Acceleration(m/s^2)'][i]) > 12:
                count_speed_a.append(i)
    if data['Transportation Mode'][i] == 'walk':
        if data['Acceleration(m/s^2)'][i] != 'N.A':
            if float(data['Acceleration(m/s^2)'][i]) > 3:
                count_speed_a.append(i)
    if data['Transportation Mode'][i] == 'bike':
        if data['Acceleration(m/s^2)'][i] != 'N.A':
            if float(data['Acceleration(m/s^2)'][i]) > 5:
                count_speed_a.append(i)

data = data.drop(count_speed_a).reset_index(drop=True)
```

```
count = 1
count_list = []
location = []

for i in range(len(data)-1):
    if data['Travel Count'][i] == data['Travel Count'][i+1]:
        count = count + 1
    else:
        count_list.append(count)
        count = 1
count_list.append(count)

for i in range(len(count_list)):
    if count_list[i] < 4:
        location.append(i+1)

data = data[-data['Travel Count'].isin(location)]
data = data.reset_index(drop=True)
```

Figure A4. The code depicted above showcases the extraction of additional features such as acceleration.

```
for i in range(len(data)):
    if i != len(data)-1:
        if data['Travel Count'][i] == data['Travel Count'][i+1]:
            if float(data['Speed(m/s)'][i]) != 0:
                non_0_speed.append(float(data['Speed(m/s)'][i]))
                non_0_speed_acceleration.append(data['Acceleration(m/s^2)'][i])
            speed.append(float(data['Speed(m/s)'][i]))
            acceleration.append(data['Acceleration(m/s^2)'][i])
        else:
            if 'N.A' in acceleration:
                acceleration.remove('N.A')
            acceleration = list(map(float, acceleration))
            acceleration = list(map(abs, acceleration))
            speed_np = np.array(speed)
            acceleration_np = np.array(acceleration)
            travel_count_feature.append(data['Travel Count'][i])
            mode_feature.append(data['Transportation Mode'][i])

            speed_max_feature.append(max(speed))
            speed_95_feature.append(round(np.percentile(speed_np, 95), 2))
            speed_75_feature.append(round(np.percentile(speed_np, 75), 2))
            speed_mean_feature.append(round(float(np.mean(speed_np)), 2))
            speed_std_feature.append(round(float(np.std(speed_np)), 2))

            acceleration_max_feature.append(max(acceleration))
            acceleration_95_feature.append(round(np.percentile(acceleration_np, 95), 2))
            acceleration_75_feature.append(round(np.percentile(acceleration_np, 75), 2))
            acceleration_mean_feature.append(round(float(np.mean(acceleration_np)), 2))
            acceleration_std_feature.append(round(float(np.std(acceleration_np)), 2))

            total_time_feature.append(float(data['Total Time(s)'][i]))
            total_distance_feature.append(round(float(data['Total Distance(m)'][i]), 2))

            speed = []
            acceleration = []
            if 'N.A' in non_0_speed_acceleration:
                non_0_speed_acceleration.remove('N.A')
            non_0_speed_acceleration = list(map(float, non_0_speed_acceleration))
            non_0_speed_acceleration = list(map(abs, non_0_speed_acceleration))
            non_0_speed_np = np.array(non_0_speed)
            non_0_acceleration_np = np.array(non_0_speed_acceleration)
            non_0_speed_mean_feature.append(round(float(np.mean(non_0_speed_np)), 2))
            non_0_speed_acceleration_mean_feature.append(round(float(np.mean(non_0_acceleration_np)), 2))

            non_0_speed = []
            non_0_speed_acceleration = []

    else:
        if 'N.A' in acceleration:
            acceleration.remove('N.A')
        acceleration = list(map(float, acceleration))
        acceleration = list(map(abs, acceleration))
        speed_np = np.array(speed)
        acceleration_np = np.array(acceleration)
        travel_count_feature.append(data['Travel Count'][i])
        mode_feature.append(data['Transportation Mode'][i])

        speed_max_feature.append(max(speed))
        speed_95_feature.append(round(np.percentile(speed_np, 95), 2))
        speed_75_feature.append(round(np.percentile(speed_np, 75), 2))
        speed_mean_feature.append(round(float(np.mean(speed_np)), 2))
        speed_std_feature.append(round(float(np.std(speed_np)), 2))

        acceleration_max_feature.append(max(acceleration))
        acceleration_95_feature.append(round(np.percentile(acceleration_np, 95), 2))
        acceleration_75_feature.append(round(np.percentile(acceleration_np, 75), 2))
        acceleration_mean_feature.append(round(float(np.mean(acceleration_np)), 2))
        acceleration_std_feature.append(round(float(np.std(acceleration_np)), 2))

        total_time_feature.append(float(data['Total Time(s)'][i]))
        total_distance_feature.append(round(float(data['Total Distance(m)'][i]), 2))

        if 'N.A' in non_0_speed_acceleration:
            non_0_speed_acceleration.remove('N.A')
        non_0_speed_acceleration = list(map(float, non_0_speed_acceleration))
        non_0_speed_acceleration = list(map(abs, non_0_speed_acceleration))
        non_0_speed_np = np.array(non_0_speed)
        non_0_acceleration_np = np.array(non_0_speed_acceleration)
        non_0_speed_mean_feature.append(round(float(np.mean(non_0_speed_np)), 2))
        non_0_speed_acceleration_mean_feature.append(round(float(np.mean(non_0_acceleration_np)), 2))
```

Figure A5. The code depicted above showcases continuation of feature extraction.

Appendix B

Modeling

Cross-Validation

```
test_size = .20
random_state = 233
X_train_clus, X_test_clus, y_train_clus, y_test_clus = cross_validation.train_test_split(
    X, y, test_size = test_size, random_state = random_state, stratify=y)
clf_dict_clus = {}
clf_report_clus = []
clf_feature_relevance_clus = []
clf_clus = SVC(kernel='rbf', gamma=0.5, C=5)
clf_name = clf_clus.__class__.__name__
print ("Training", clf_name, "...")
# Fit model on training data
clf_dict_clus[clf_name] = clf_clus.fit(X_train_clus, y_train_clus)
# Predict based on it
# y_pred = clf.predict(X_train)
# Perform cross validation
start = time()
scores = cross_validation.cross_val_score(clf_clus, X_train_clus, y_train_clus, cv=skf, scoring='accuracy')
end = time()
duration = end-start
print ("Average CV performance for {} : {:.6} (in {:.6} seconds)".format(clf_name, scores.mean(), duration))
clf_report_clus.append([clf_name, scores.mean(), duration])
# Store information in list for better visibility
clf_report_clus = pd.DataFrame(clf_report_clus, columns=['classifier', 'Cluster_train_accuracy',
                                                         'Cluster_train_time'])
```

Model learning curve

```
def plot_learning_curve(estimator, title, X, y, ylim=(0.5, 1.01), cv=skf, train_sizes=np.linspace(.1, 1.0, 5)):
    train_sizes, train_scores, test_scores = learning_curve(
        estimator, X, y, cv=cv, train_sizes=train_sizes, scoring='f1_weighted')
    plt.figure()
    plt.title(title)
    if ylim is not None:
        plt.ylim(*ylim)
    plt.xlabel("Training examples")
    plt.ylabel("Score")
    train_sizes, train_scores, test_scores = learning_curve(
        estimator, X, y, cv=cv, train_sizes=train_sizes, scoring='f1_weighted')
    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)
    print("SVM Macro weighted F1-score: %.2f%%" % (train_scores_mean[-1] * 100.0))
    print("SVM Macro weighted F1-score: %.2f%%" % (test_scores_mean[-1] * 100.0))
    plt.grid()

    plt.fill_between(train_sizes, train_scores_mean - train_scores_std,
                     train_scores_mean + train_scores_std, alpha=0.1,
                     color="r")
    plt.fill_between(train_sizes, test_scores_mean - test_scores_std,
                     test_scores_mean + test_scores_std, alpha=0.1, color="g")
    plt.plot(train_sizes, train_scores_mean, 'o-', color="r",
             label="Training score")
    plt.plot(train_sizes, test_scores_mean, 'o-', color="g",
             label="Cross-validation score")

    plt.legend(loc="best")
    return plt
```

Model Learning Curve

```
clf = SVC(kernel='rbf', gamma=0.5, C=5)
plot_learning_curve(clf, u"SVM learning curve", X_train, y_train)
plt.show()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
test_accuracy = precision_score(y_test, y_pred, average='weighted')
```

Classification Report and Confusion Matrix

```
print(classification_report(y_test, y_pred))
cf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize = (15,10))
ax= sns.heatmap(cf_matrix, cmap='coolwarm',annot= True, linewidth= 1, fmt = "d" )
# labels, title and ticks
ax.set_xlabel('Predicted labels');
ax.set_ylabel('True labels');
ax.set_title('Confusion Matrix');
ax.xaxis.set_ticklabels(['airplane','bike','boat','bus','car','motorcycle','run','subway','taxi','train','walk'
                         ]);
ax.yaxis.set_ticklabels(['walk','train','taxi','subway','run','motorcycle','car','bus','boat','bike','airplane'
                         ]);
```

Figure B1. The code depicted above showcases Support Vector Machine modelling

Cross-Validation

```

test_size = .20
random_state = 233
X_train_clus, X_test_clus, y_train_clus, y_test_clus = cross_validation.train_test_split(
    X, y, test_size = test_size, random_state = random_state, stratify=y)
clf_dict_clus = {}
clf_report_clus = []
clf_feature_relevance_clus = []
clf_clus = RandomForestClassifier(random_state = random_state, max_depth=10, criterion ='entropy',n_jobs=2)
clf_name = clf_clus.__class__.__name__
print ("Training", clf_name, "...")
# Fit model on training data
clf_dict_clus[clf_name] = clf_clus.fit(X_train_clus, y_train_clus)
# Predict based on it
# y_pred = clf.predict(X_train)
# Perform cross validation
start = time()
scores = cross_validation.cross_val_score(clf_clus, X_train_clus, y_train_clus, cv=skf, scoring='accuracy')
end = time()
duration = end-start
print ("Average CV performance for {} : {:.6} (in {:.6} seconds)".format(clf_name, scores.mean(), duration))
clf_report_clus.append([clf_name, scores.mean(), duration])
# Store information in list for better visibility
clf_report_clus = pd.DataFrame(clf_report_clus, columns=['classifier', 'Cluster_train_accuracy',
                                                         'Cluster_train_time'])

```

K-fold validation

```
n_folds = 5
skf = StratifiedKFold(n_splits=n_folds, shuffle=True, random_state=233)
```

Model learning curve

```
def plot_learning_curve(estimator, title, X, y, ylim=(0.5, 1.01), cv=skf, train_sizes=np.linspace(.1, 1.0, 5)):
    train_sizes, train_scores, test_scores = learning_curve(
        estimator, X, y, cv=cv, train_sizes=train_sizes, scoring='f1_weighted')
    plt.figure()
    plt.title(title)
    if ylim is not None:
        plt.ylim(*ylim)
    plt.xlabel("Training examples")
    plt.ylabel("Score")
    train_sizes, train_scores, test_scores = learning_curve(
        estimator, X, y, cv=cv, train_sizes=train_sizes, scoring='f1_weighted')
    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)
    print("RF Macro weighted F1-score: %.2f%%" % (train_scores_mean[-1] * 100.0))
    print("RF Macro weighted F1-score: %.2f%%" % (test_scores_mean[-1] * 100.0))
    plt.grid()

    plt.fill_between(train_sizes, train_scores_mean - train_scores_std,
                     train_scores_mean + train_scores_std, alpha=0.1,
                     color="r")
    plt.fill_between(train_sizes, test_scores_mean - test_scores_std,
                     test_scores_mean + test_scores_std, alpha=0.1, color="g")
    plt.plot(train_sizes, train_scores_mean, 'o-', color="r",
             label="Training score")
    plt.plot(train_sizes, test_scores_mean, 'o-', color="g",
             label="Cross-validation score")

    plt.legend(loc="best")
    return plt
```

```
clf = RandomForestClassifier(random_state = random_state, max_depth=10, criterion ='entropy', n_jobs=2)
plot_learning_curve(clf, u"Random Forest learning curve", X_train, y_train)
plt.show()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
test_accuracy = precision_score(y_test, y_pred, average='weighted')
```

Figure B2. The code depicted above showcases Random Forest modelling

Cross-Validation

```
test_size = .20
random_state = 233
X_train_clus, X_test_clus, y_train_clus, y_test_clus = cross_validation.train_test_split(
    X, y, test_size = test_size, random_state = random_state, stratify=y)
clf_dict_clus = {}
clf_report_clus = []
clf_feature_relevance_clus = []
clf_clus = DecisionTreeClassifier(random_state = random_state,max_depth=4)
clf_name = clf_clus.__class__.__name__
print ("Training", clf_name, "...")
# Fit model on training data
clf_dict_clus[clf_name] = clf_clus.fit(X_train_clus, y_train_clus)
# Predict based on it
# y_pred = clf.predict(X_train)
# Perform cross validation
start = time()
scores = cross_validation.cross_val_score(clf_clus, X_train_clus, y_train_clus, cv=5, scoring='accuracy')
end = time()
duration = end-start
print ("Average CV performance for {} : {:.6} (in {:.6} seconds)".format(clf_name, scores.mean(), duration))
clf_report_clus.append([clf_name, scores.mean(), duration])
# Store information in list for better visibility
clf_report_clus = pd.DataFrame(clf_report_clus, columns=['classifier', 'Cluster_train_accuracy',
                                                         'Cluster_train_time'])
```

K-fold validation

```
n_folds = 5
skf = StratifiedKFold(n_splits=n_folds, shuffle=True, random_state=233)
```

Model learning curve

```
def plot_learning_curve(estimator, title, X, y, ylim=(0.5, 1.01), cv=skf, train_sizes=np.linspace(.1, 1.0, 5)):
    train_sizes, train_scores, test_scores = learning_curve(
        estimator, X, y, cv=cv, train_sizes=train_sizes, scoring='f1_weighted')
    plt.figure()
    plt.title(title)
    if ylim is not None:
        plt.ylim(*ylim)
    plt.xlabel("Training examples")
    plt.ylabel("Score")
    train_sizes, train_scores, test_scores = learning_curve(
        estimator, X, y, cv=cv, train_sizes=train_sizes, scoring='f1_weighted')
    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)
    print("DT Macro weighted F1-score: %.2f%%" % (train_scores_mean[-1] * 100.0))
    print("DT Macro weighted F1-score: %.2f%%" % (test_scores_mean[-1] * 100.0))
    plt.grid()

    plt.fill_between(train_sizes, train_scores_mean - train_scores_std,
                     train_scores_mean + train_scores_std, alpha=0.1,
                     color="r")
    plt.fill_between(train_sizes, test_scores_mean - test_scores_std,
                     test_scores_mean + test_scores_std, alpha=0.1, color="g")
    plt.plot(train_sizes, train_scores_mean, 'o-', color="r",
             label="Training score")
    plt.plot(train_sizes, test_scores_mean, 'o-', color="g",
             label="Cross-validation score")

    plt.legend(loc="best")
    return plt
```

```
clf = DecisionTreeClassifier(random_state = random_state,max_depth=4)
plot_learning_curve(clf, u" DT learning curve", X_train, y_train)
plt.show()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
test_accuracy = precision_score(y_test, y_pred, average='weighted')
```

Figure B3. The code depicted above showcases Decision Tree modelling

```
Xscaler = MinMaxScaler(feature_range=(-1, 1))
Xscaler.fit(X_train)
X_train = Xscaler.transform(X_train)
X_test = Xscaler.transform(X_test)

generator_train = TimeseriesGenerator(X_train,y_train.values.reshape(-1),length=32)
generator_test = TimeseriesGenerator(X_test,y_test.values.reshape(-1),length=6)
```

```
def build_model(hp):
    model = Sequential()
    model.add(layers.LSTM(hp.Choice('units', [9, 32, 64, 128, 256]),
                          dropout=hp.Choice('dropout', [0.0, 0.1, 0.2, .3, .4, .5]),
                          input_shape=(1,X_train.shape[1])))
    model.add(layers.Dense(hp.Choice('units', [9, 32, 64, 128, 256]),
                           activation='tanh'))
    model.add(layers.Dense(hp.Choice('units', [9, 32, 64, 128, 256]),
                           activation='tanh'))
    model.add(layers.Dense(9, activation='softmax'))
    learning_rate = hp.Float("lr", min_value=1e-4, max_value=1e-2, sampling="log")
    beta_1 = hp.Float("b1", min_value=0.9, max_value=1, sampling="log")
    model.compile(
        optimizer=optimizers.Adam(learning_rate=learning_rate,beta_1=beta_1),
        loss=losses.SparseCategoricalCrossentropy(from_logits=False),
        metrics=["accuracy"],
    )
    return model

build_model(keras_tuner.HyperParameters())
```

```
tuner = keras_tuner.RandomSearch(
    hypermodel=build_model,
    objective="accuracy",
    max_trials=50,
    seed=1
)
```

```
tuner.search_space_summary()
tuner.search(generator_train, epochs=10)
```

```
best_hps = tuner.get_best_hyperparameters()
model = build_model(best_hps[0])
model.summary()
model.fit(generator_train, epochs=300)
```

```
loss_per_epoch = model.history.history['loss']
plt.plot(range(len(loss_per_epoch)),loss_per_epoch);
model.evaluate(generator_test)
```

```
predictions = model.predict(generator_test)
pred_list = []
for i in predictions:
  pred_list.append(i.argmax())
```

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(list(generator_test.targets[6:]),pred_list)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```

```
from sklearn.metrics import classification_report
print(classification_report(list(generator_test.targets[6:]),pred_list))
```

```
best_hps[0].values
```

Figure B4. The code depicted above showcases LSTM modelling