COMPARING MACHINE LEARNING ALGORITHMS FOR VEHICLE ROUTE PREDICTION

by

PRITHVIDHAR PUDU

Presented to the Faculty of the Honors College of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

HONORS BACHELOR OF SCIENCE IN COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2023

ACKNOWLEDGMENTS

Many thanks to my mentor and professor for his patience and wisdom. I am also deeply indebted to the Honors College at The University of Texas at Arlington for their continued support and encouragement. Special thanks to Social Knowledge for providing the resources needed for this research project.

I could not have taken this journey without my fellow students for their advice on the topic and moral support. I would also like to recognize the librarians of the university library who provided invaluable knowledge.

Lastly, I am grateful for my family, especially my pet dog Oreo whose encouragement has given me the motivation to proceed with this project.

May 2, 2023

ABSTRACT

COMPARING MACHINE LEARNING ALGORITHMS FOR

ROUTE PREDICTION

Prithvidhar Pudu, B.S Computer Science

The University of Texas at Arlington, 2023

Faculty Mentor: Christopher McMurrough

Route prediction is an important tool in understanding the trajectories of vehicles. This information can be used to make intelligent decisions in city planning such as traffic avoidance and pedestrian safety. Machine Learning can predict these routes by learning common patterns from previous vehicle paths. These vehicle paths are plentiful due to the abundance of IoT devices and advancements in computer vision. The most suitable machine learning algorithm that could accurately predict destinations was determined by passing vehicle data to three popular machine learning algorithms: Decision Trees, Artificial Neural Networks, and Naïve-Bayes. The training data for these algorithms was converted into a suitable format using the Pandas python library. Moreover, the hyper-parameters for each algorithm were tuned to maximize the accuracy of the prediction. Respectively, decision trees were observed to provide the highest accuracy of 99.7% with the Naïve-Bayes and ANN showing 99.5% and 98%.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Application Context

Social Knowledge is the hub for Recreational Vehicle (RV) users all over the United States. Their website, RVlife, provides resources to help RVers purchase and maintain an RV. A vacation planner is also available to provide RV-specific pathing. However, all these services rely on data provided by a user. To help ease the vacation planning process, the company tasked our team to implement a functionality that allows their website to predict the next likely rest stop a user would take given their starting location. This would create a full vacation plan by showing users suggestions of popular rest stops close to their current location.

1.2 Project Value

The application consists of a front end that is responsible for receiving the user's position and displaying their vacation path. The suggested rest stops needed to create the vacation plan are obtained from the backend portion of the application. This section is made up of a Python Django webserver that houses the machine learning model that predicts the rest stops. This information is sent to the front end through a RESTful API. Moreover, the entire application is present in the Amazon Web Services (AWS) cloud platform as the service has more flexibility in terms of scaling and interactivity. This is due to the fact that most of the resources provided are virtual, removing the issue of compatibility.

Furthermore, this application can help RVers nationwide plan their vacation in a series of clicks and take advantage of RVLife's other RV-specific tools.

1.3 Honors Value

Many cities are becoming smarter every year in terms of the technology embedded in every day infrastructure ranging from buildings to household items and vehicles. This brings with it large amounts of data that can be processed for valuable information. When it comes to vehicle data, information such as position, speed, and driving habits can be received. Indeed, this is the basis for the honors contribution.

Trajectory prediction can provide information regarding a city's vehicle population, such as the path taken by certain vehicles like buses and trucks. In addition, it can provide valuable insight into the travel pattern of city denizens by observing taxi routes. This is done by making destination predictions based on a vehicle's position and other attributes. Furthermore, it can help improve road infrastructure by understanding vehicle-pedestrian relationships through vehicle trajectories and help reduce congestion as a result. Therefore, to make this prediction, a machine learning model is needed to help identify common patterns and classify them accordingly. However, there are several algorithms to achieve this result. To identify the most suitable model, I will compare three popular Machine Learning (ML) techniques and compare their prediction capability based on their accuracy and other ML evaluation metrics.

CHAPTER 2

BACKGROUND

2.1 Recent Studies

Trajectory prediction is a common topic of research and with the computational ability of the latest technology and the plethora of vehicle data available, it is clear that Machine Learning has some applications in this field. This has encouraged many to create diverse architectures consisting of several layers of mathematical computations to identify common patterns in vehicle traffic. This chapter will go over two such studies and will review common ML algorithms that are used in this area of research.

2.1.1 Classification Learning Neural Network

Deep Learning is a subset of ML that consists of software architectures made of several layers of mathematical computations. Data is passed through these layers to identify common patterns within it. This feature extraction property is helpful when determining the next position of a vehicle as previous data can be processed to make such a prediction feasible.

Tang et al. (2021) takes advantage of this property using their Classification Learning Neural Network (CLNN) model to better understand travel behaviors in cities. The data used by this model takes three forms: partial, historical, and external. Partial data consists of GPS coordinates with historical data referring to prior vehicle information, like location and speed. Next, external data consists of information not related to the vehicle directly, like driver characteristics and the day of the week. CLNN uses this information to greatly increase its effectiveness as a deep learning model. This is because when more context of the vehicle's current state is available and is in large quantity, the model will have enough data to properly optimize a function to make accurate predictions. Compared to other models that use only partial data to determine a vehicle trajectory, CLNN considers the surrounding of a vehicle, which provides better insight into its behavior and encourages pedestrian-vehicle relationships to be woven into the computations of the model. As a result, CLNN better represents a vehicle in an urban setting. Below is an illustration of the model's architecture which displays the computational layers that extract the information mentioned previously.

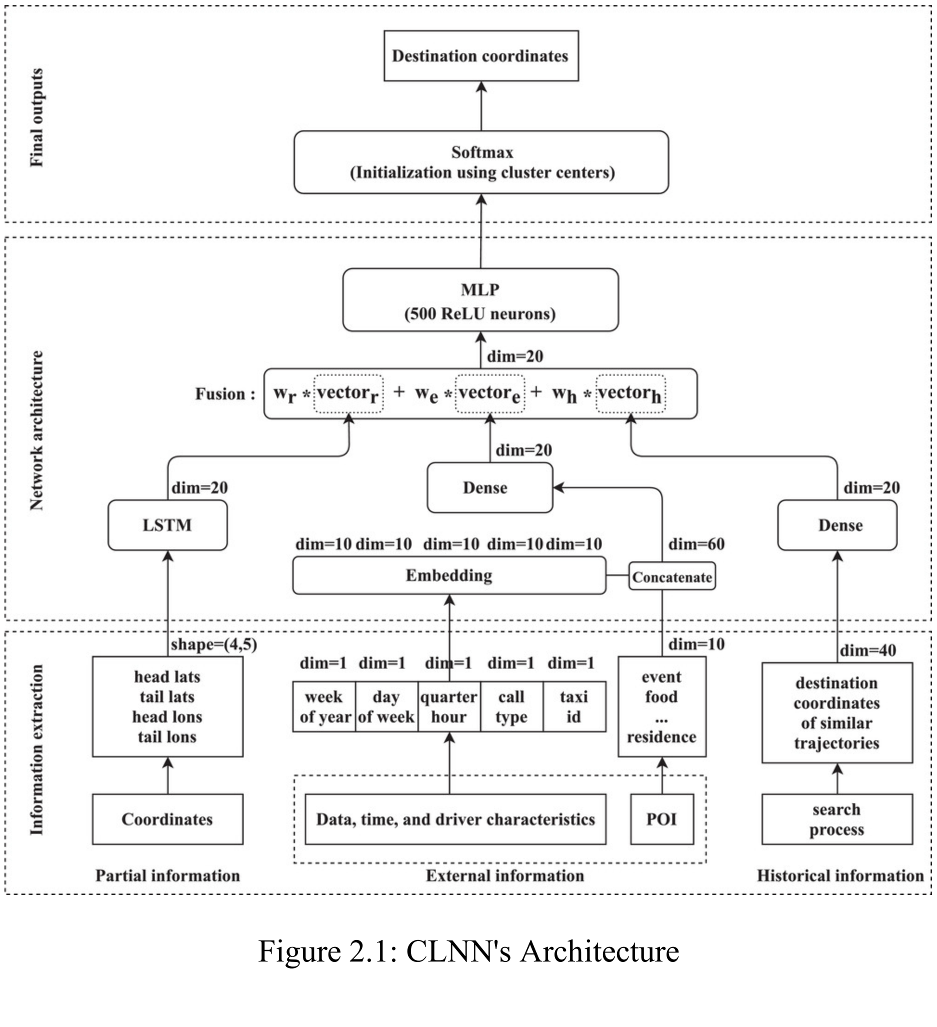


Figure 2.1: CLNN's Architecture

The model's architecture consists of layers for each of the three datatypes. Once the computation at each layer is applied to the data the results are fused to another series of computations. Finally, a destination prediction is provided. CLNN was compared to other common algorithms and provided more accurate results comparatively. To summarize, CLNN is a deep learning architecture that extracts large amounts of features from seemingly unrelated data and makes accurate predictions using them by repeatedly optimizing the data mining capabilities of Deep Learning.

2.1.2 Gaussian Mixture Model

The previous study required large amounts of data processing to be usable. Jin-jun Tang et al.'s (2019) approach, on the other hand, offers a more lightweight method to calculate vehicle density in city streets. To accomplish this, a clustering algorithm is first used to determine areas of high vehicle congestion. Next, these areas are labeled into clusters to differentiate among them. This reduces the effort of prediction as every vehicle location need not be identified and only requires it to be part of a group. To classify the clusters based on vehicle traffic, a Gaussian Mixture Model (GMM) is used. This algorithm is used to determine the parameters of the probability distribution to which a vehicle belongs, i.e., it helps to determine which cluster a vehicle resides. This approach does not need large amounts of computations and is ideal to determine high areas of traffic by repeatedly using data to form probability distribution estimations. In brief, Tang et al. (2019) takes advantage of similar data to predict future hotspots using repeated loops of prediction until convergence is achieved.

2.2 Common Algorithms

The studies above display the assortment of techniques that are possible with Machine learning. This section aims to provide a better understanding of the more commonly used algorithms. These are also used in this study and should prove some insight into their implementation.

2.2.1 Artificial Neural Networks

Artificial Neural Networks, or ANNs for short, are networks used to process data in a way similar to the human brain. Similar to a human brain, an ANN contains large amounts of tiny data units called neurons. Each neuron contains a function that takes processes input and returns an output. These artificial neurons are organized in layers. The input layer is responsible for receiving external data. Following this, the output of this layer is passed to the hidden layer, which consists of a large number of neurons. All these units are interconnected and influence each other's outputs. Finally, the hidden layer passes its output to the output layer, which represents a desired quantity or label. Each neuron has a weight that is adjusted to produce the desired outcome. Below is a simple network illustrating the concept discussed.

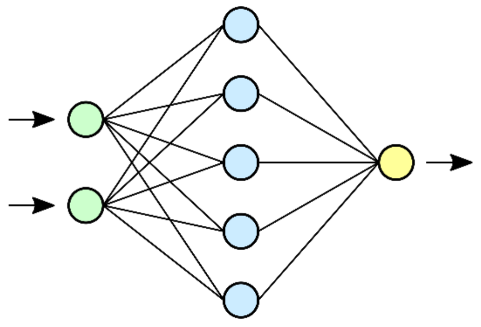


Figure 2.2: A Simple Neural Network

Neural networks have the ability to find non-linear features in almost any dataset. This is due to their ability to form complex functions by simply adding more neurons to the network. The strength of a neural network is limited to the computational ability of its hardware. Due to their highly flexible nature, ANNs are used in several applications from medical diagnosis to speech recognition.

2.2.2 Decision Trees (DT)

Decision Trees (DT) are a more intuitive form of supervised machine learning. It is intuitive because it uses previous data to determine what a new piece of data might be classified as. To elaborate, DTs consist of nodes (data attributes) that have tests to help classify a data point. They also contain leaves that sort the point into a particular category. A root node is the first and highest node in the tree and is split into decision nodes based on various metrics. These nodes are further split into more nodes until a leaf is formed. In many cases, a split is made based on how much information is provided when splitting an attribute. This information is calculated by using either Gini or entropy, two metrics that help measure the information an attribute provides. Decision trees take fewer resources to create as they are solely based on the data itself and do not need any features to be extracted. The inference is also fast as traveling down a tree is quick and efficient. However, if the data is not properly pre-processed a tree can grow too large and increase training time considerably. See Figure 2.3 for an example of this structure.

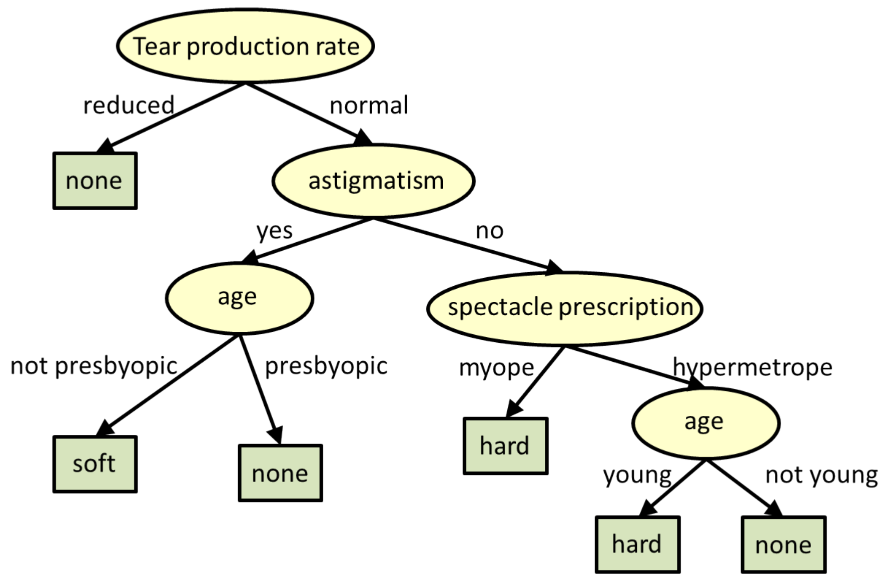


Figure 2.3: A Simple Decision Tree

2.2.3 Naïve-Bayes Classifiers

Naïve-Bayes classifiers are a probabilistic machine learning algorithm that creates probability distributions to determine what category a data point most likely belongs to. This is done using the two theories in its namesake. One of them is the Bayes theorem which proposes that evidence can affect the probability of something happening. It is mathematically represented in the following formula with "A" and "B" being two events:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Figure 2.4: Bayes' Theorem

The second theory is an assumption that all the evidence events that affect the outcome of a situation are independent of each other. This eases the calculations as, under this assumption, all event probabilities can be multiplied to avoid complex calculations. Naïve-Bayes classifiers are easy to build and can be accurate so long as the Naïve assumption holds. These classifiers see use in spam detection and recommendation systems as independent events are plentiful in those areas.

CHAPTER 3

METHODOLOGY

This chapter will go over the implementation of the experiment from the data formatting to the hyper tuning of the individual models. The first section will cover how the environment for the project was set up on an EC2 instance on the AWS cloud platform. It will go over the installation of the Jupyter Notebook server along with the Anaconda environment. The next section will go over the clustering process used to group the GPS coordinates together for easy classification. Finally, the last section will discuss the creation of my ANN and the hyperparameter tuning of the other machine learning model.

3.1 Installation

Many data science projects require several types of software packages to perform different functions in the data science pipeline. Some of these functions include pre-processing, feature extraction, data mining, and visualization. It becomes a hassle to manage the installation of these several packages as one needs to take into account their compatibility, availability, and location on the system. This is where Anaconda comes into play by offering a package distribution service that can create virtual environments. An environment is a folder/directory in one's system that houses all the required packages. Anaconda helps maintain these packages by ensuring that they are of the latest version and are compatible with each other

3.1.1 Anaconda Installation

To begin the experiment, the Anaconda distribution service first had to be installed onto the EC2 instance. This was accomplished by first accessing the instance, through a shell session created using the Windows Command Prompt. Once the session was active, the Anaconda installation file was downloaded by using the following command in the Linux terminal: *curl -O https://repo.anaconda.com/archive/Anaconda3-2021.04-Linux-aarch64.sh.* After downloading the file, the service was installed by running the installation program. With the service installed, a virtual environment was created using the command: "*conda create*." Using this environment, the necessary packages can be installed and managed in a single folder in the EC2 instance.

3.1.2 Jupyter Notebook Installation

One of the packages primarily used in this study is Jupyter Notebook. Jupyter Notebooks are applications that enable data scientists to visualize their data mining processes. It enables them to execute different data engineering pipelines separately by running code in distinct code blocks. The program also supports the markdown language allowing users to provide explanations for their implementation.

After installing the software, using Anaconda, to create a notebook, a configuration file was generated to configure the password and network setting for the Jupyter Server. A hash of a password was created using IPython and the server was set to be hosted on the machine itself on port number 8888 and address "0.0.0.0". This meant that the notebook could be reached from a browser using the URL: "http:localhost/8888". Then, using a browser to connect to the Jupyter Server, a Jupyter Notebook was created to house the project and the importation of the data

3.2 Data Loading and Pre-Processing

Python is one of the primary programming languages used in data science thanks to its rich set of libraries that support the field. One of these is called Pandas which is a python library that offers data structures and operations for manipulating data tables. This library was installed via Anaconda and used to import data from the CSV files. These files contain the GPS data needed for the ML models. Pandas stores these values in a data structure called a DataFrame. These structures consist of rows and columns and support operations to easily access the data stored within them.

Following this storage, the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm is used to cluster GPS coordinates that are within a certain distance from one another. The algorithm works by grouping points together based on density. To elaborate, it works under the assumption that dense points are separated by areas of sparse points. This works for the study as coordinates that are closer together in a cluster are places that a user would likely travel to if starting in that cluster. Sklearn's implementation of the algorithm was used with the appropriate parameters provided. The radius size to be part of a cluster was set to 10 kilometers with the Haversine distance metric being used. This is because Haversine distance calculates the distance between two points on a sphere which is ideal for the study's use case of GPS coordinates.

3.3 Machine Learning Model Training and Tuning

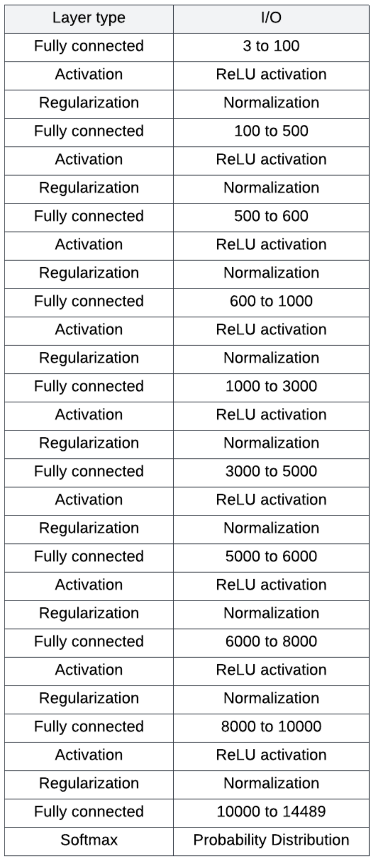
This section will cover how each of the models was tuned and tested to obtain optimum cluster prediction accuracy. The Artificial Neural Network will be covered first followed by Decision Trees and ending with Naïve- Bayes classifiers.

3.3.1 Artificial Neural Network

Pytorch is a deep-learning Python library that provides the tools and functions that enable users to create complex machine learning models. The ANN was created using this library by taking advantage of the library's primary data structure: Tensors. Tensors utilize the GPU of a system to perform complex mathematical calculations such as convolutions and differentiations in a computationally efficient way. They also store the gradient of each calculation which will become important later.

To begin training, the data first has to be split into three sections: Train, Validation, and Test. The training portion of the dataset will be used to teach the Neural Network the common pattern in the data while the validation set is used to check the accuracy of the system. If the accuracy is not at a preferred level, the model is tuned to slowly increase the validation accuracy. The test set is saved for last to record the final accuracy of the model. This is done to ensure that the model is evaluated on unseen data and best represents its capabilities. The data set was split using the random_split() function from PyTorch to divide the data set into the three mentioned sets in the ratio 60:20:20 (Train, Validate, Test). Following this step, the data is loaded into data loaders, another PyTorch data structure that groups data in batches and randomly shuffles them. It is also good to note that the input consists of GPS coordinates with cluster labels attached to them.

This makes the model a supervised learning algorithm. With the data ready, the model is constructed using the architecture below:



Layer type	I/O
Fully connected	3 to 100
Activation	ReLU activation
Regularization	Normalization
Fully connected	100 to 500
Activation	ReLU activation
Regularization	Normalization
Fully connected	500 to 600
Activation	ReLU activation
Regularization	Normalization
Fully connected	600 to 1000
Activation	ReLU activation
Regularization	Normalization
Fully connected	1000 to 3000
Activation	ReLU activation
Regularization	Normalization
Fully connected	3000 to 5000
Activation	ReLU activation
Regularization	Normalization
Fully connected	5000 to 6000
Activation	ReLU activation
Regularization	Normalization
Fully connected	6000 to 8000
Activation	ReLU activation
Regularization	Normalization
Fully connected	8000 to 10000
Activation	ReLU activation
Regularization	Normalization
Fully connected	10000 to 14489
Softmax	Probability Distribution

Figure 3.1: ANN Architecture

The fully connected layers consist of a dense network of neurons that contain linear functions with weights and biases. These weights are normalized in the normalization layer to make the model more generalized. The ReLU activation function is also used to gather non-linear features from the data. The output of the model is a large probability distribution determining the most likely cluster a given GPS coordinate would go to. The data loops through the model repeatedly with the optimizer changing the weights of the neuron to make the input match their labels.

The closeness to a label is determined using a loss function. CrossEntropy is used here for this model as it is known for its reliability in various datasets. The Standard Gradient Descent optimizer was chosen for this model as it eventually finds weight that minimizes the amount of loss.

Tuning this model involved adding and removing layers with the final version being the one shown before. The hardware constraints prevented the model from training for long periods which negatively affected its accuracy. In response to this, Sklearn's state-of-the-art Multi-Layer Perceptron (MLP) was used to predict clusters. The MLP resembles the Pytorch architecture but differs by using state-of-the-art implementations to counteract the hardware constraints mentioned before which means utilizing advanced algorithms to reduce the time and space complexity of the model. This algorithm contained several parameters ranging from the number of neurons to the learning rate. The best parameters were determined using GridSearchCV which allows users to go through combinations of parameter values to determine one that provides the best accuracy. This provided better results compared to the earlier version.

3.3.2 Decision Tree

The Decision Tree (DT) model involved creating a recursive function that repeatedly generated tree-like data structures to create the entire model. This creation of trees involved determining which of the input attributes contained the most information. In other datasets, different attributes contain different amounts of useful information that can help classify data accordingly. Determining the best attribute involves creating child trees of attributes and comparing their entropy to their parent. Entropy refers to the amount of chaos in the child node representing an attribute. Less child entropy is desired as it tells us that splitting the dataset based on those attributes helps categorize the data more clearly. This splitting is done till a node only has one value; this makes it a leaf. Once this recursive process is done a DT is produced. The initial implementation resulted in low accuracy due to the space complexity of the function. To overcome this Sklearn's state-of-the-art implementation was used.

Sklearn's decision tree classifier has parameters ranging from tree depth to the number of leaves, all of which were hyper-tuned using GridSearchCV which like in the previous model tries all combinations of hyperparameters to determine one that gives the highest accuracy.

3.3.3 Naïve-Bayes Classifier

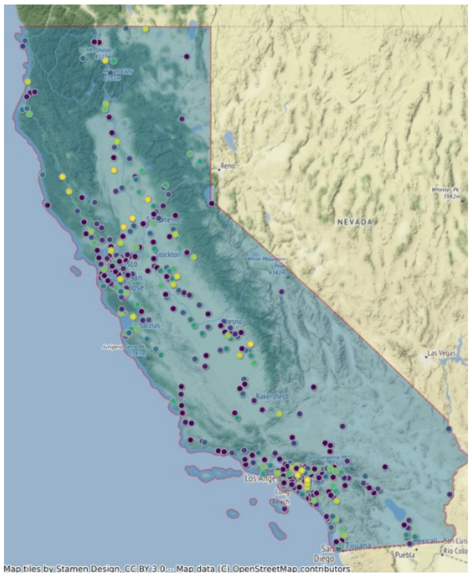
Sklearn's implementation was used for this study because collecting the evidence to determine the likelihood of an event occurring requires state-of-the-art implementation. A Gaussian Naïve-Bayes classifier was chosen instead of the default one as the GPS data worked well when made into a Gaussian distribution which consisted of calculating the mean and median of the dataset. The hyperparameters of the model were once again determined using GridSearchCV.

CHAPTER 4

RESULTS

This section will go over the results when applying the models to three datasets. The datasets include cities in India, California, and RV stops in the United States. The models' evaluation metrics will be provided along with a possible reason for the results.

4.1 California



NEVADA
Reno
Las Vegas
Stockton
Los Angeles
San Diego
Tijuana
Map tiles by Stamen Design, CC BY 3.0 -- Map data (C) OpenStreetMap contributors

Figure 4.1: Clusters of Cities in California

Table 4.1: Evaluation Metrics from the California Dataset

Algorithm	Accuracy	F1 Score	Training Time (secs)
Decision Tree	57%	0.24	0.003
Naive Bayes	48%	0.21	0.01
Artificial Neural Network	27%	0.007	0.12

The California dataset has relatively few coordinates compared to the other datasets. Table 4.1 shows how the models' ability to learn new and diverse patterns was hindered. Among the models, the decision tree performed the best in all categories thanks to its ability to extract features efficiently and in an intuitive manner that is relevant to this task. On the other hand, the Naïve Bayes and ANN models struggle to form a proper mapping between locations and clusters. This is due to most coordinates being independent of each other and the lack of powerful hardware.

4.2 India

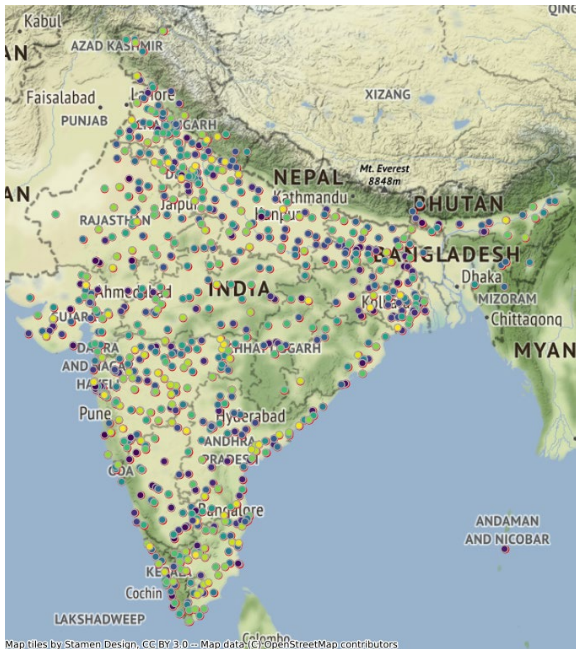


Figure 4.2: Clusters of Cities in India

Table 4.2: Evaluation Metrics from the India Dataset

Algorithm	Accuracy	F1 Score	Training Time (secs)
Decision Tree	61%	0.31	0.004
Naive Bayes	56%	0.24	0.01
Artificial Neural Network	18%	0.007	0.001

The results in Table 4.2 are similar to the previous dataset but have improved for the Decision Tree and Naïve Bayes models. This is because the India dataset contains more coordinates compared to the previous test which allows the models to extract more distinguishing features. For instance, the Decision tree can now form more meaningful nodes from large amounts of data (higher accuracy score of 61%). The Naïve-Bayes classifier had a large pool of evidence events to make more accurate predictions (accuracy score of 56%). The hardware limitations still negatively impacted the Neural Network's performance.

4.3 United States of America

Table 4.3: Evaluation Metrics from the U.S. Dataset

Algorithm	Accuracy	F1 Score	Training Time (secs)
Decision Tree	99.8%	0.69	0.496
Naive Bayes	99.5%	0.64	0.01
Artificial Neural Network	97.68%	0.05	150

The U.S. dataset consists of over 100,000 RV rest stop locations all over the United States. This allows the models to extract helpful features from the sparse coordinates. Table 4.3 shows the three models performing exceptionally well with the ANN having the longest training time in the entire test period. This is because forming a prediction function for such a large dataset is computationally very expensive. Decision trees and Naïve-Bayes classifiers have comparatively simpler computations ranging from counting to sorting. They do not have to deal with optimizing the weights of thousands of individual neurons, thus, giving them accuracy scores of 99.8% and 99.5% respectively.

CHAPTER 5

CONCLUSION

The goal of this experiment was to determine which Machine Learning algorithm, among the three picked, performed the best with GPS coordinates. This was determined by first setting the proper environment. Next, the proper tools were installed to visualize and summarize the performance of the model after training. The training of the models involved three datasets with varying types of GPS coordinates. The California coordinates were low in number and spread apart while the India coordinates were densely packed. The US dataset contained a very large number of coordinates compared to the other two locations. Training these models involves tuning them to achieve the highest accuracy they can offer which results in the previously discussed results.

The results indicated that decision trees work well when categorizing coordinates. Decision trees are known to be useful when the dataset has distinguishing features as nodes for each of these features can be made easily. The GPS coordinates can be made into buckets to categorize them and make accurate predictions. Future research in this area involves comparing and using more complex models based on the ones used in this study. A more complex version of decision trees involves running multiple smaller trees in parallel and averaging the result of each tree to make a prediction. This model is known as a Random Tree Classifier and is used in many machine learning applications dealing with categorical data. More attributes such as speed and time of day can make ANNs produce more accurate and valuable information if provided with powerful hardware.

REFERENCES

Bhiksha, B. (2023). Decision tree to determine type of contact lens to be worn by a person. The known attributes of the person are tear production rate, whether he/she has astigmatism, their age (categorized into two values) and their spectacle prescription. [Computer Graphic]. Carnegie Mellon University School of Computer Science. https://www.cs.cmu.edu/~bhiksha/courses/10-601/decisiontrees/

Maliah, M. (2023). Brain Cartoon [Computer Graphic]. CleanPNG. https://www.cleanpng.com/png-artificial-neural-network-deep-learning-biological-739402/

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011, February 1). Scikit-Learn: Machine learning in Python. The Journal of Machine Learning Research. Retrieved April 24, 2023, from https://dl.acm.org/doi/10.5555/1953048.2078195

Tang, J., Liang, J., Yu, T., Xiong, Y., & Zeng, G. (2021). Trip destination prediction based on a deep integration network by fusing multiple features from taxi trajectories. IET Intelligent Transport Systems, 15(9), 1131–1141. https://doi.org/10.1049/itr2.12075

Tang, J.-jun, Hu, J., Wang, Y.-wei, Huang, H.-lai, & Wang, Y.-hai. (2019). Estimating hotspots using a gaussian mixture model from large-scale taxi GPS trace data. Transportation Safety and Environment, 1(2), 145–153. https://doi.org/10.1093/tse/tdz006

Tang, T. (2021). The network architecture of CLNN [Computer Graphic]. The Institute of Engineering and Technology. https://ietresearch.onlinelibrary.wiley.com/doi/full/10.1049/itr2.12075

.

BIOGRAPHICAL INFORMATION

Prithvidhar Pudu is a senior Computer Science student at The University of Texas at Arlington and a member of the institution's Honors College. His research interests include Machine Learning, Artificial Intelligence, and Data Science. His projects include a web application to classify images of wild animals and a text classifier to classify biased and non-biased articles. He aims to become a software developer to apply high-quality design patterns to large-scale applications and build pipelines to train and monitor machine learning models. He is also interested in continuing his higher education by completing his master's in computer science.