TRAFFIC PREDICTION FOR INTELLIGENT TRANSPORTATION SYSTEMS USING MACHINE LEARNING

Mini Project submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

Submitted by:

VODELA NITYA	**221710305060**
SIROBHUSHANAM BHAVYA	**221710305054**
PAINDLA SAIGANESH	**221710305039**
SOMAYAJULA SRI LAKSHMI	**221710305055**

Under the esteemed guidance of

Ms. RIZAVIA SAYEED
Assistant Professor

GITAM
(DEEMED TO BE UNIVERSITY)
VISAKHAPATNAM ✹ HYDERABAD ✹ BENGALURU

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF TECHNOLOGY

GANDHI INSTITUTE OF TECHNOLOGY AND MANAGEMENT (GITAM)
(Declared as Deemed-to-be-University u/s 3 of UGC Act of 1956)

HYDERABAD CAMPUS

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF TECHNOLOGY

GANDHI INSTITUTE OF TECHNOLOGY AND MANAGEMENT (GITAM)
(Declared as Deemed-to-be-University u/s 3 of UGC Act of 1956)

HYDERABAD CAMPUS

DECLARATION

We hereby declare that the Mini Project entitled "Traffic Prediction for Intelligent Transportation Systems using Machine Learning" is an original work in Department of Computer science and Engineering, GITAM School of Technology, GITAM (Deemed-to-be-University), Hyderabad submitted in partial fulfillment of the award of the degree of B.Tech. in Computer Science and Engineering. The work has not been submitted to any other college or university for the award of any degree or diploma.

Date: 10 December 2020

GROUP: CSEBNUM_E10
VODELA NITYA 221710305060
SIROBHUSHANAM BHAVYA 221710305054
PAINDLA SAIGANESH 221710305039
SOMAYAJULA SRI LAKSHMI 221710305055

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF TECHNOLOGY

GANDHI INSTITUTE OF TECHNOLOGY AND MANAGEMENT (GITAM)
(Declared as Deemed-to-be-University u/s 3 of UGC Act of 1956)

HYDERABAD CAMPUS

CERTIFICATE

This is to certify that the Mini Project Report entitled "Traffic Prediction for Intelligent Transportation Systems using Machine Learning" is being submitted by **CSEBNUM_E10: Vodela Nitya (221710305060), Sirobhushanam Bhavya (221710305054), Paindla Saiganesh (221710305039), Somayajula Sri Lakshmi (221710305055)** in partial fulfillment of the requirement for the award of degree of Bachelor of Technology in CSE at GITAM (Deemed to Be University), Hyderabad during the academic year 2020-21. The Mini Project has been approved as it satisfies the academic requirements.

Ms. Rizavia Sayeed
Assistant Professor
Department of CSE

Prof. S. Phani Kumar
Head of the Department
Department of CSE

ACKNOWLEDGEMENT

Apart from our effort, the success of this mini project largely depends on the encouragement and guidance of our faculty. We take this opportunity to express our gratitude to the people who have helped us in the successful competition of this mini project.

We are extremely thankful to our honorable Pro-Vice Chancellor, **Prof. N. Siva Prasad** for providing necessary infrastructure and resources for the accomplishment of our seminar.

We are highly indebted to **Prof. N. Seetharamaiah**, Principal, School of Technology, for his support during the tenure of the seminar.

We are very much obliged to our beloved **Prof. S. Phani Kumar**, Head of the Department of Computer Science & Engineering for providing the opportunity to undertake this project and encouragement in completion of our mini project.

We hereby wish to express our deep sense of gratitude to **Dr. S Aparna**, Assistant Professor, Department of Computer Science and Engineering, School of Technology and to **Ms. Rizavia Sayeed**, our Project Guide and Assistant Professor, Department of Computer Science and Engineering, School of Technology for the esteemed guidance, moral support and invaluable advice provided by them for the success of the Mini Project as well as **Mrs. Hima Bindu**, AMC and Assistant Professor, Department of Computer Science and Engineering for always motivating us.

We are also thankful to all the staff members of the Computer Science and Engineering department who have cooperated in making our Mini Project a success.

Sincerely,
VODELA NITYA
SIROBHUSHANAM BHAVYA
PAINDLA SAIGANESH
SOMAYAJULA SRI LAKSHMI

CONTENTS

ABSTRACT

The future of our generation will be based on the concepts of Machine Learning and AI. This project is based on the patterns of traffic and predicting it titled "**Traffic Prediction for Intelligent Transportation Systems using Machine learning**". Using the models and various features of machine learning needed to make predictions, the prediction of traffic will be done over 4 junctions given in the data set. Furthermore this will be plotted based on real values to the predicted values. This can help develop the city and transportation to avoid any accidents or delay in schedule which occurs just due to traffic.

The vision of the project is to convert a city into a digital and intelligent city to improve the efficiency of services for the citizens. One of the problems faced by the government is traffic. The government wants to implement a robust traffic system for the city by being prepared for traffic peaks. They want to understand the traffic patterns of the four junctions of the city. Traffic patterns on holidays, as well as on various other occasions during the year, differ from normal working days. This is important to take into account for forecasting.

The sensors on each of these junctions were collecting data at different times, hence see traffic data from different time periods. To add to the complexity, some of the junctions have provided limited sparse data requiring thoughtfulness when creating future projections. Depending upon the historical data of 20 months, the government is looking for us to deliver accurate traffic projections for the coming four months. Our algorithm will become the foundation of the larger transformation to make our city smart and intelligent.

LIST OF FIGURES

LIST OF TABLES

LIST OF SCREENS

1. Training Set
2. Test set
3. Dataset Upload
4. Train Set top 5 rows
5. Describe train set
6. Describe test set
7. EDA
8. Lag Features
9. Normalization
10. After Normalization - Real Values
11. After Normalization - Predicted Values
12. Reshape Data
13. Validation
14. Epoch
15. Code for Plotting Prediction
16. Junction 1
17. Junction 2
18. Junction 3
19. Junction 4
20. RMSE Value

GROUP MEMBERS NAME	POSITION OF EVERY MEMBER	TASK PERFORMED BY MEMBER
Vodela Nitya	Team Leader	Analysis, Documentation, Coding, Testing
Sirobhushanam Bhavya	Team Member	Dataset gathering, Coding, Designing, Analysis
Paindla Saiganesh	Team Member	Concept Analysis, Documentation, Testing, Coding
Somayajula Sri Lakshmi	Team Member	Dataset gathering, Coding, Designing , Analysis

Table 1 : Task Distribution in the Group

1. INTRODUCTION

The motivation behind the project is the constant delay and problems caused due to traffic. Google maps does have this prediction but sometimes it does not give the correct prediction. Hence by this project predicting traffic can help resolve common issues, furthermore it can also help develop ITS - Intelligent Transportation System.traffic congestion at peak hours reaches unacceptable levels in many parts of the world. These are all serious issues caused by current transportation systems, and optimization through the usage of modern technologies is necessary for the required improvements. A lot of the innovation that is part of the solution already exists and is what makes up *Intelligent Transportation Systems* (ITS). Directive 2010/40/EU of the European Parliament and of the Council of 7 july 2010 defined ITS in the following way:

> ITS integrates telecommunications, electronics and information technologies with transport engineering in order to plan, design, operate, maintain and manage transport systems.

This definition indicates that any information technology that aids transportation in one way or another can be included as one of the many innovations under the term ITS. Applications that provide travel times or the most efficient route to a given destination are examples of such technologies. Traditionally, these technologies func- tioned based on simplistic evaluations, and could only be actively updated based on occuring events. However, proactive adaptation to the ever changing dynamics of urban traffic can be achieved. This is done through approximative forecasting of future traffic patterns.

Naturally, this would greatly improve the performance of existing ITS technologies. Achieving this, however, requires historic measurements of the parameters to be forecasted. Such parameters could include the traffic flow and speed at some location. Measurements of these parameters can be done in many ways, such as video detection, inductive loops and magnetic sensors .

Additionally, possession of historic data of traffic incidents and various weather parameters may also be useful, as these often impact the traffic quite heavily. Subsequently, this data can be analyzed and may reveal various traffic patterns. In turn, these patterns could make it possible to forecast future traffic situations.

1.1. MACHINE LEARNING

Prediction in this project is done using Machine Learning, including some concepts of Deep Learning. Machine Learning(ML) is the scientific study of algorithms and statistical models that computer systems use in order to perform a specific task effectively without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of Artificial Intelligence(AI).

1.1.1. Importance of Machine Learning

Consider some of the instances where machine learning is applied: the self-driving Google car, cyber fraud detection, online recommendation engines—like friend suggestions on Facebook, Netflix showcasing the movies and shows as suggestions, and "more items to consider" on Amazon—are all examples of applied machine learning.

All these examples echo the vital role machine learning has begun to take in today's data-rich world. Machines can aid in filtering useful pieces of information that help in major advancements, and it is already seen how this technology is being implemented in a wide variety of industries.

With the constant evolution of the field, there has been a subsequent rise in the uses, demands, and importance of machine learning. Big data has become quite a buzzword in the last few years; that's in part due to increased sophistication of machine learning, which helps analyze those big chunks of big data. Machine learning has also changed the way data extraction, and interpretation is done by involving automatic sets of generic methods that have replaced traditional statistical techniques.

The process flow depicted here represents how machine learning works :

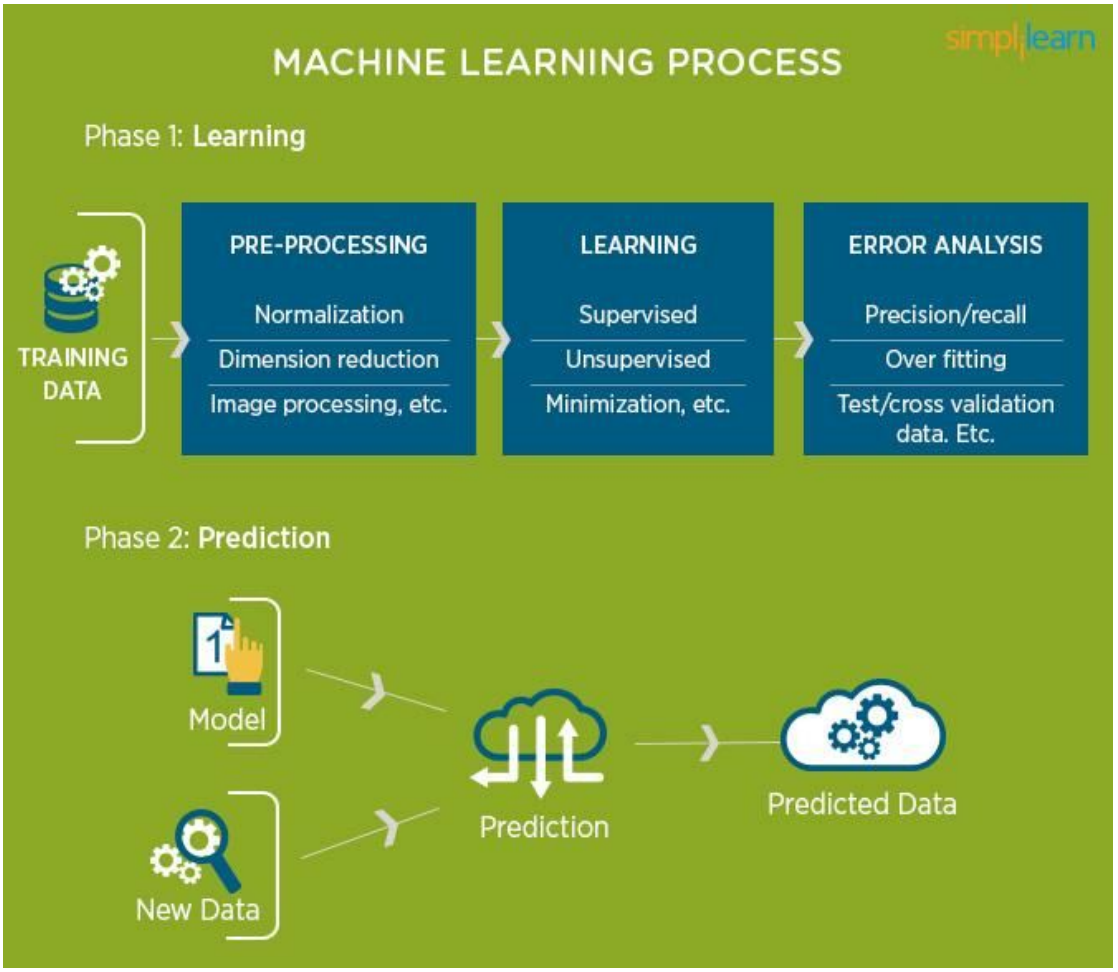


Figure 1.1 : Machine Learning Process

1.1.2. Applications of Machine Learning

Web search results, Real-time ads on web pages and mobile devices, email spam filtering, network intrusion detection, and pattern and image recognition are some of the many applications and uses of machine learning. All these are by-products of applying machine learning to analyze huge volumes of data.

Traditionally, data analysis was always being characterized by trial and error, an approach that becomes impossible when data sets are large and heterogeneous. Machine learning comes as the solution to all this chaos by proposing clever alternatives to analyzing huge volumes of data.

By developing fast and efficient algorithms and data-driven models for real-time processing of data, machine learning can produce accurate results and analysis.

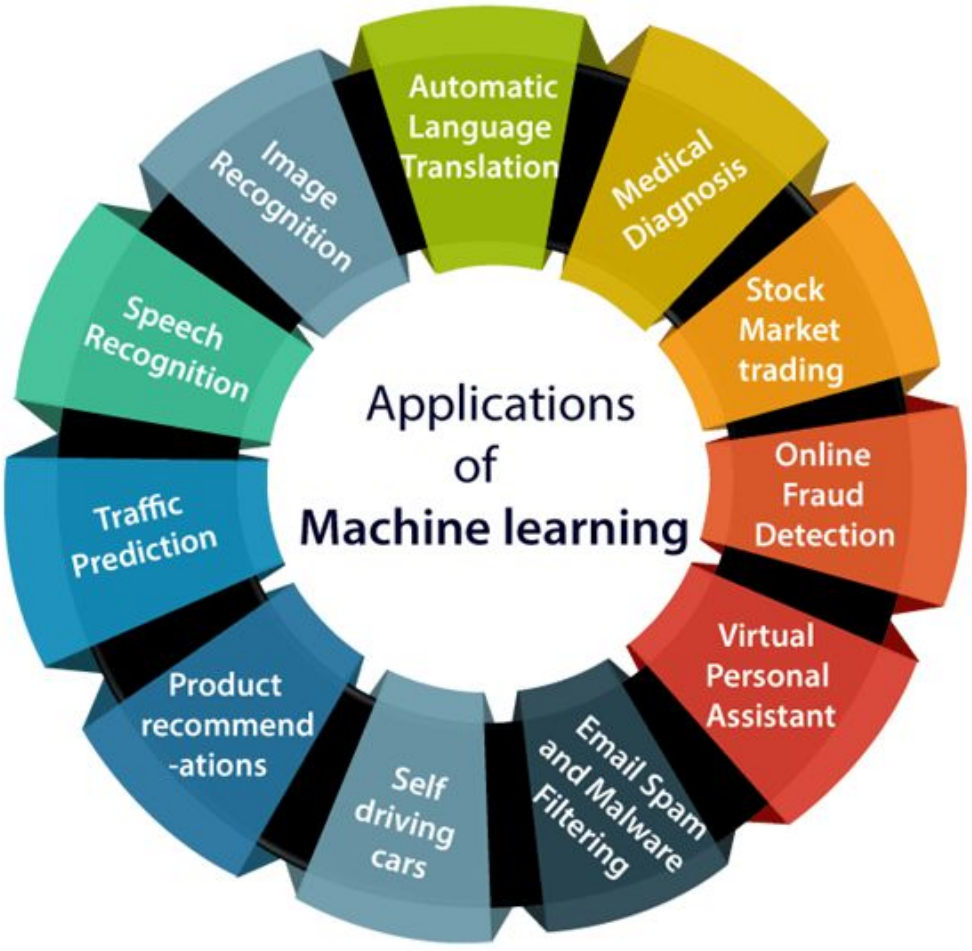


Figure 1.2 : Applications of Machine Learning

1.1.3. Machine Learning Types

As with any method, there are different ways to train machine learning algorithms, each with their own advantages and disadvantages. To understand the pros and cons of each type of machine learning first look at what kind of data they ingest. In ML, there are two kinds of data — labeled data and unlabeled data.

Labeled data has both the input and output parameters in a completely machine-readable pattern, but requires a lot of human labor to label the data, to begin with. Unlabeled data only has one or none of the parameters in a machine-readable form. This negates the need for human labor but requires more complex solutions.

There are also some types of machine learning algorithms that are used in very specific use-cases, but three main methods are used today.

The types of machine learning algorithms differ in their approach, the type of data they input and output, and the type of task or problem that they are intended to solve.

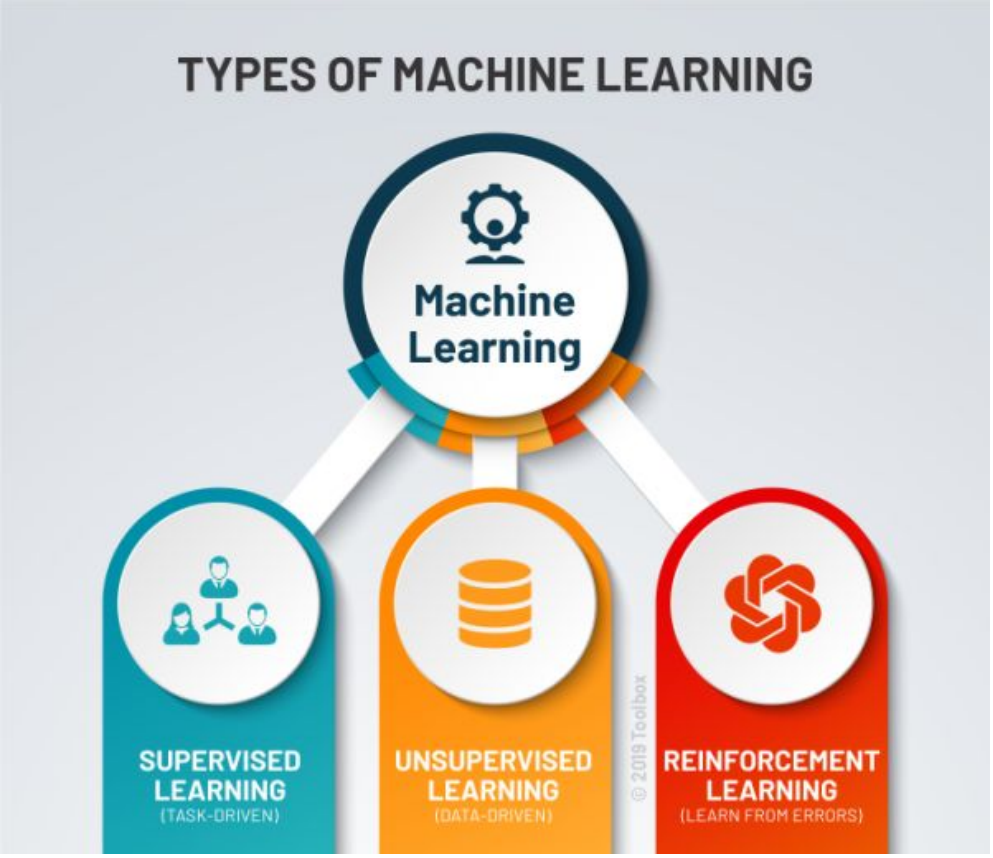


Figure 1.3 : Types of Machine Learning

1.1.3.1. Supervised Learning :

When an algorithm learns from example data and associated target responses that can consist of numeric values or string labels, such as classes or tags, in order to later predict the correct response when posed with new examples comes under the category of supervised learning.

Classification, in which the algorithm attempts to label each example by choosing between two or more different classes. Choosing between two classes is called binary classification, such as determining whether or not someone will default on a loan. Choosing between more than two classes is referred to as multiclass classification.

Supervised machine learning algorithms uncover insights, patterns, and relationships from a labelled training dataset – that is, a dataset that already contains a known value for the target variable for each record. Due to the provision of machine learning algorithms with the correct answers for a problem during training, it is able

to “learn” how the rest of the features relate to the target, to uncover insights and make predictions about future outcomes based on historical data. In this project, Supervised learning is used to help make predictions from the data set taken.

Examples of Supervised Machine Learning Techniques are Regression, in which the algorithm returns a numerical target for each example, such as how much revenue will be generated from a new marketing campaign.

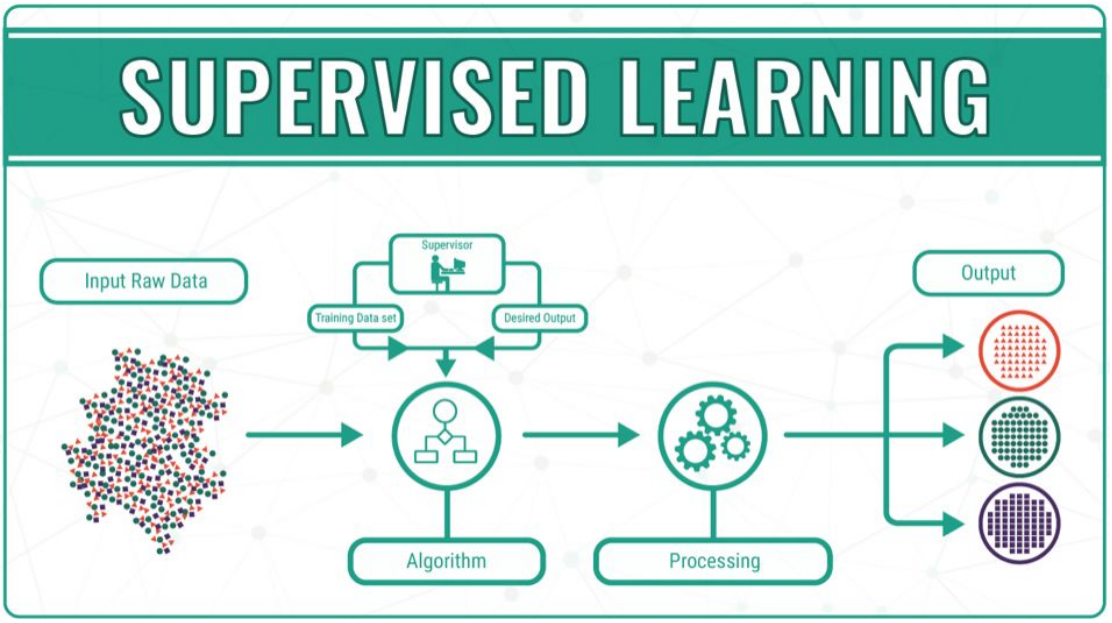


Figure 1.4 : Supervised Learning

1.1.3.2. Unsupervised Learning:

When an algorithm learns from plain examples without any associated response, leaving to the algorithm to determine the data patterns on its own. This type of algorithm tends to restructure the data into something else, such as new features that may represent a class or a new series of uncorrelated values.

They are quite useful in providing humans with insights into the meaning of data and new useful inputs to supervised machine learning algorithms.

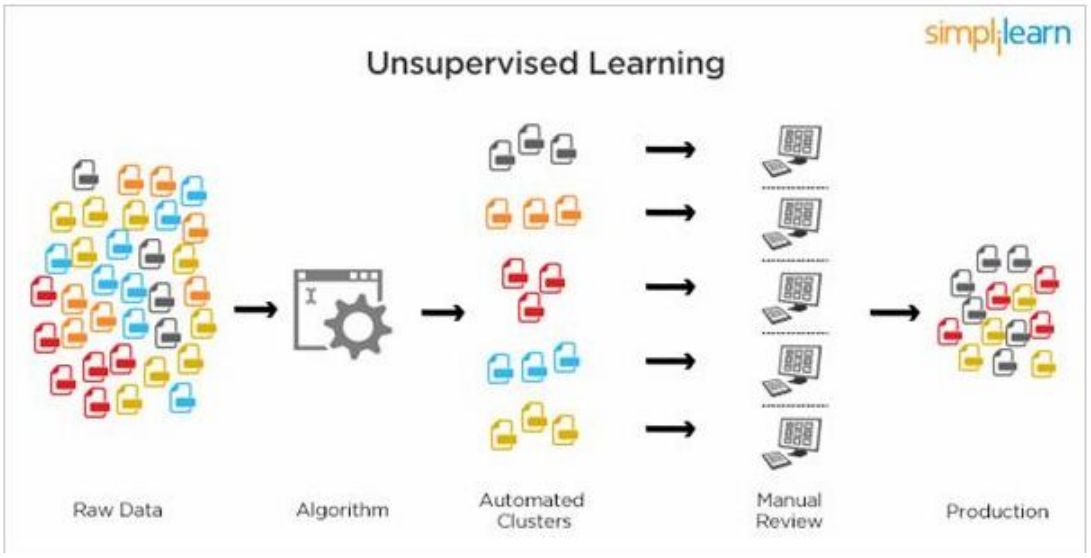


Figure 1.5 : Unsupervised Learning

Popular techniques where unsupervised learning is used also include self-organizing maps, nearest neighbor mapping, singular value decomposition, and k-means clustering. Basically, online recommendations, identification of data outliers, and segment text topics are all examples of unsupervised learning

1.1.3.3. Reinforcement Learning:

Describes a class of problems where an agent operates in an environment and must learn to operate using feedback.

Reinforcement learning is learning what to do — how to map situations to actions—so as to maximize a numerical reward signal. The learner is not told which actions to take, but instead must discover which actions yield the most reward by trying them.

The use of an environment means that there is no fixed training dataset, rather a goal or set of goals that an agent is required to achieve, actions they may perform, and feedback about performance toward the goal.

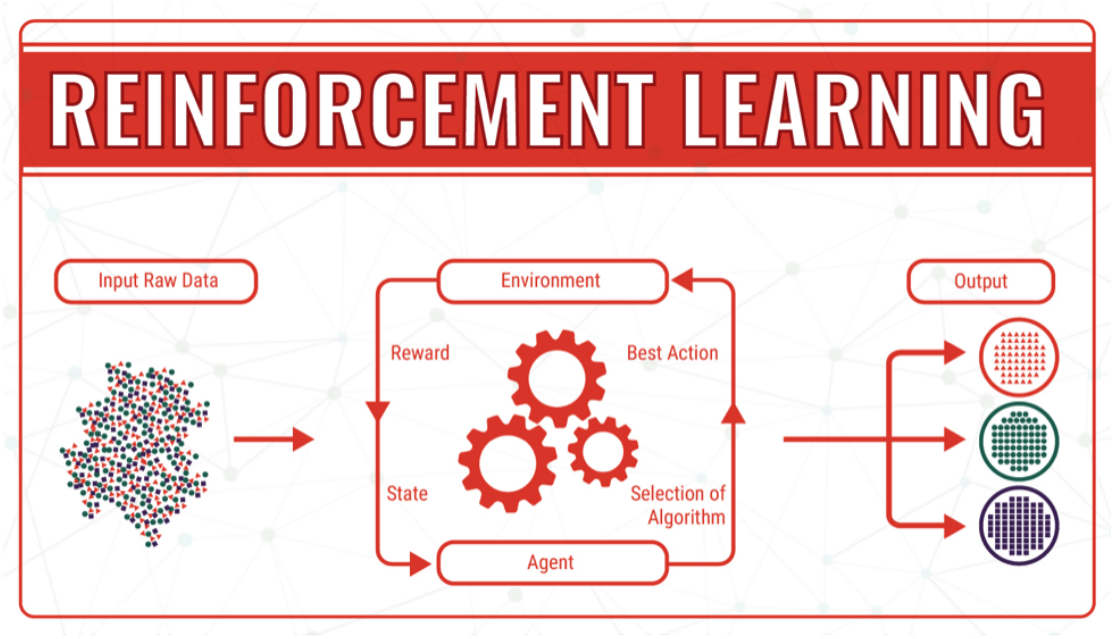


Figure 1.6 : Reinforcement Learning

An example of a reinforcement problem is playing a game where the agent has the goal of getting a high score and can make moves in the game and receive feedback in terms of punishments or rewards.

1.1.3.4. Semi Supervised Learning:

As the name suggests, semi-supervised learning is a bit of both supervised and unsupervised learning and uses both labeled and unlabeled data for training. In a typical scenario, the algorithm would use a small amount of labeled data with a large amount of unlabeled data.

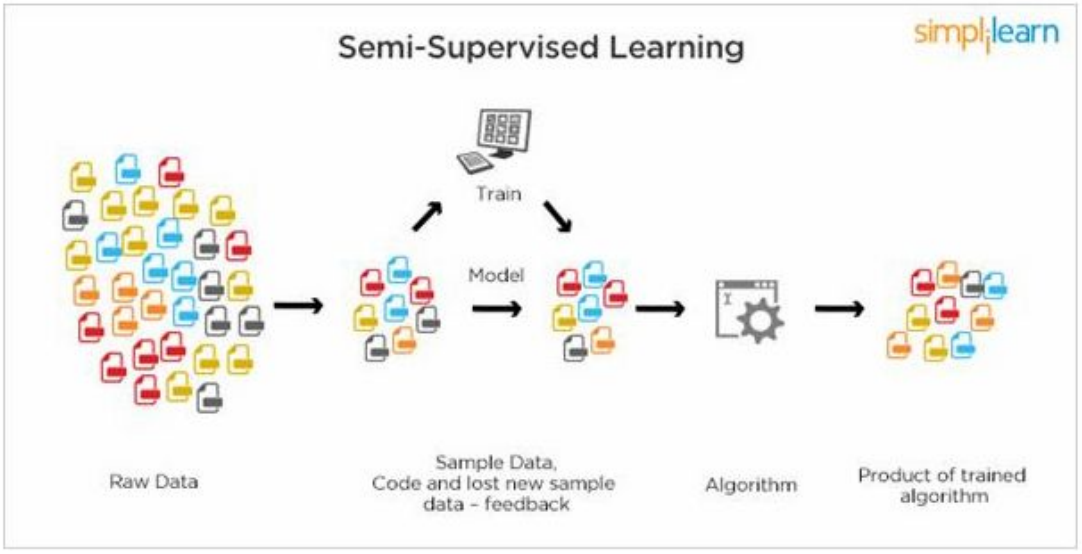


Figure 1.7 : Semi - Supervised Learning

A common example of an application of semi-supervised learning is a text document classifier. This is the type of situation where semi-supervised learning is ideal because it would be nearly impossible to find a large amount of labeled text documents. This is simply because it is not time efficient to have a person read through entire text documents just to assign it a simple classification.

1.2. PYTHON

Python programming language is a widely used, simple, easy and effective language to work with and can be understood by anybody with ease and this project is done solely based on python.

1.2.1. INTRODUCTION TO PYTHON

Python is a widely used general-purpose, high level programming language. It was created by Guido van Rossum in 1991 and further developed by the Python Software Foundation. It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code.

Python is a programming language that helps work quickly and integrate systems more efficiently. There are two major Python versions: Python 2 and Python 3. The latest version of python 3 is used around the world presently.

1.2.2. FEATURES OF PYTHON

- **Easy to code:** Python is a high-level programming language. Python is very easy to learn the language as compared to other languages like C, C#, Javascript, Java, etc. It is very easy to code in python language and anybody can learn python basics in a few hours or days. It is also a developer-friendly language.

- **Free and Open Source:** Python language is freely available at the official website. Since it is open-source, this means that source code is also available to the public.
- **Object-Oriented Language:** One of the key features of python is Object-Oriented programming. Python supports object-oriented language and concepts of classes, objects encapsulation, etc.
- **GUI Programming Support:** Graphical User interfaces can be made using a module such as PyQt5, PyQt4, wxPython, or Tk in python. PyQt5 is the most popular option for creating graphical apps with Python.
- **High-Level Language:** Python is a high-level language. Writing programs in python,it is not needed to remember the system architecture, nor manage the memory.
- **Extensible feature:** Python is an Extensible language. Writing some Python code into the C or C++ language and also be compiled in the C/C++ language.
- **Python is Portable language:** Python is also a portable language. For example, python code for windows and can run this code on other platforms such as Linux, Unix, and Mac then it is not needed to change and can run this code on any platform.
- **Python is an Integrated language:** Python is also an Integrated language because it can easily integrate python with other languages like c, c++, etc.
- **Interpreted Language:** Python is an Interpreted Language because Python code is executed line by line at a time. Unlike other languages C, C++, Java, etc. there is no need to compile python code this makes it easier to debug our code. The source code of python is converted into an immediate form called bytecode.
- **Large Standard Library:** Python has a large standard library which provides a rich set of modules and functions. There are many libraries present in python for such as regular expressions, unit-testing, web browsers, etc.
- **Dynamically Typed Language:** Python is a dynamically-typed language. That means the type (for example- int, double, long, etc.) for a variable is decided at run time not in advance because of this feature but don’t need to specify the type of variable.

2. LITERATURE SURVEY

Many works has already been proposed for predicting traffic using intelli- gent machine learning systems like Fei-Yue Wang proposed parallel control and management for intelligent transportation systems and Chun-Hsin Wu, Jan-Ming Ho, and D. T. Lee proposed deployment of support vector regres- sion for mapping traffic densities, Jason Brownlee has proposed the deploy- ment of bagging and random forest ensemble models for traffic predictions.

But many of these models use shallow traffic models and are still somewhat failing due to the enormous dataset dimension. This is where our project comes in clutch .

2.1. METHODS USED IN RESEARCH PAPERS

Some of the various types of research techniques which has been deployed are

2.1.1. TRANSACTIONS ON INTELLIGENT TRANSPORT

Fei-Yue Wang et al[1] : The development and deployment of Intelligent Trans- portation System provide better accuracy for Traffic flow prediction. It is dealt with as a crucial element for the success of advanced traffic management systems, advanced public transportation systems, and traveller information systems.

2.1.2. ACCELERATED INCIDENT DETECTION VIA VEHICLE KINEMATICS

Yongchang Ma, Mashrur Chowdhury, Mansoureh Jeihani, and Ryan Fries[2] : proposed The dependency of traffic flow is dependent on real-time traffic and historical data collected from various sensor sources, including inductive loops, radars, cameras, mobile Global Positioning System, crowdsourcing, social media.

2.1.3. DELEGATE MULTI AGENT SYSTEM

Rutger Claes, Tom Holvoet, and Danny Weyns[3] : have proposed the use of

decentralized approach for anticipatory vehicle routing using delegate multiagent systems.

2.1.4 SEMANTIC INDEXING

MehulMahrishiandSudhaMorwal[4] : proposed the use of multi-layer concepts of neural networks to mining the inherent properties in data from the lowest level to the highest level .

2.1.5. SHORT RANGE COMMUNICATION

Joseph D Crabtree and Nikiforos Stamatiadis[5] : proposed the implementation via driver as-sistance system (DAS), autonomous vehicles (AV)and Traffic Sign Recognition (TSR) .

2.1.6. KINEMATIC DATA FROM PROBE VEHICLES

H Qi, RL Cheu, and DH Lee.[6] : proposed the use of probe vehicles to gather test and simulation data to better understand the nature of the traffic

2.1.7. LONG SHORT TERM MEMORY NETWORKS

Z. Zhao, W. Chen, X. Wu, P. C. Y. Chen, and J. Liu.[7] : proposed the implementation of LONG SHORT TERM NEURAL NETS. In this control, strategies identify the potential congestion on the roads, and it directs the passengers to take some alternative routes to their destinations.

2.1.8. DEEP LEARNING IN NETWORKING

C. Zhang, P. Patras, and H. Haddadi.[8] : proposed the idea of deploying deep neural

nets in mobile and wireless communications.

2.1.9. TRAVEL TIME PREDICTION

Chun-Hsin Wu, Jan-Ming Ho, and D. T. Lee.[9] : proposed studying the travel time via support vector regression.

2.1.10. DECISION TREE METHODS

Yan-Yan Song and LU Ying.[10] : proposed deploying decision trees to study the traffic.

2.1.11. MERGING MOBILITY AND ENERGY

Yiming He, Mashrur Chowdhury, Yongchang Ma, and Pierluigi Pisu.[11] : proposed merging of mobility and energy visions which were in conflict with one another via SVM.

2.1.12. BAGGING AND RANDOM FOREST

Jason Brownlee[12] : proposed deploying bagging and random forest models.

3. PROBLEM ANALYSIS

3.1. PROBLEM

The dependency of traffic flow is dependent on real-time traffic and historical data collected from various sensor sources, including inductive loops, radars, cameras, mobile Global Positioning System, crowdsourcing, social media. Traffic data is exploding due to the vast use of traditional sensors and new technologies.

However, there are already lots of traffic flow prediction systems and models; most of them use shallow traffic models and are still somewhat failing due to the enormous dataset dimension.

Early traffic data is difficult to obtain, and the data environment is small and of low quality, leading to the use of small sample data in prediction research. Therefore, the complex model of pure mathematics theory has been introduced into a large number of prediction studies, which ignores the inherent characteristics and evolution mechanism of traffic flow. However, overly complex models, such as neural network and combination models, have complex processes and large computation, which are not conducive to the practical application of short-term traffic prediction.

Some problems based on predicting can be:

• Difficulty of implementation: How difficult to implement the model, including figuring out the structure of the model, the parameters or the hyper-parameters of the model. The accuracy of Regression, example-based and Kernel-based model is heavily dependent on the parameters, it will cost significant effort to find an adequate set of parameters.

• Cost of implementation: It takes the training time and hardware dependency of the model into consideration. Normally, the NN model spends more time on model training, and the computing capacity of the hardware will affect the performance of the model.

• Dataset requirement: Does the model need a big dataset to guar- antee the performance of the model? The NN model always needs a big dataset to adjust the parameters inside the model in a supervised way. Meanwhile, the Example-based model, i.e., KNN model, also has a relatively high requirement on the dataset, in order to build up a search pool of reliable candidates.

• Deep-learning structure: Determine if this type of model has Deep-learning structure.

• Ability of spatial feature and temporal feature extraction: whether the model has a clear structure to extract a specific feature.

• Time cost of prediction: How much time will the model cost to make the final prediction. Normally, this is highly related to the structure of the model. Greater complexity in the structure will result in greater time costs.

• Maintain cost: Includes the updates of the hyper-parameters of the model; for KNN it also includes the updates of the candidates search pool.

• Robustness: How well the model will deal with the changes of the length of the prediction interval, data loss, etc. Meanwhile, this also includes the performance when the model is applied to a bigger road network.

3.2. ROOT CAUSE OF PROBLEM

Traffic can be caused due to various factors in daily life and it can be difficult to predict. In the morning the traffic visibly increases and eventually reaches a peak (rush hour). After this, the traffic slowly decreases throughout the day, and then eventually increases back up again in the afternoon when people are driving home (rush hour).

Traffic in itself is not dangerous. Since excessive speed is the most common cause of accidents leading to death or bodily injury, and traffic results in slow movement of vehicles, a motorist who is caught in a traffic jam is not in danger of an accident caused by high speed. Any accidents that occur would be at low speed. But the extreme measures of distressed motorists can lead to danger, including aggressive lane changing, rat running down small streets, U-turns, and speeding after the jam is over to make up lost time can be hazardous.

Some of the major factors include:

- Accidents
- Vehicle breakdown
- Absence of required management (Traffic Police)
- Unexpected Rally or Riots
- Rubbernecking of accidents, disabled vehicles, stopped motorists, or other sights out of the ordinary
- Construction, which may result in lane closures or the need to drive more slowly than normal
- Events that draw large crowds
- Inclement weather, leading motorists to have to drive more slowly or cautiously
- Unforeseen emergencies

3.3. DRAWBACKS

Due to human and natural factors, such as communication problems, equipment failures there is

- Increasing **traffic** congestion, air pollution, and fuel consumption.
- Increase in use of less-adequate roads to avoid **traffic** signs.
- Excessive delay due to time allocated by the **traffic** signals.

4. DESIGN

This chapter describes the deployment of some specific types of neural networks to predict and intelligently determine traffic density and congestion at various places. In this project, Deep Neural Networks because they can deal with high amounts of datasets and their enormous dimensions.

The more the dimensions and dense the data is the better the performance of the model. This makes our model redundant to specific anomalies in the dataset. If a small dataset has anomalies it will result in very biased models. But in high dimensional datasets, the anomalies can be encoded or even discarded while having minuscule to no impact on the accuracy, because of the model moving on to another dimension without any anomalies.

The types of Neural networks the deployed models are :

1. Long Short Term Memory Network
2. Recurrent And Sequential Networks
3. Dense Networks
4. Feature Engineering- Generating Lag Features
5. Rectified Linear Activation Function And Adam Optimiser

4.1. LONG SHORT TERM MEMORY NETWORKS

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems. This is a behavior required in complex problem domains like machine translation, speech recognition, and more.

LSTMs are a complex area of deep learning. It can be hard to understand what LSTMs are, and how terms like bidirectional and sequence-to sequence relate to the field. Recurrent neural networks are different from traditional feed-forward neural networks. This difference in the addition of complexity comes with the promise of new behaviors that the traditional methods cannot achieve.

Recurrent networks have an internal state that can represent context information. They keep information about past inputs for an amount of time that is not fixed a priori, but rather depends on its weights and on the input data.

A recurrent network whose inputs are not fixed but rather constitute an input sequence can be used to transform an input sequence into an output sequence while taking into account contextual information in a flexible way.

The three basic requirements of recurrent neural networks are :

I. The system is able to store information for an arbitrary duration.
II. The system be resistant to noise i.e. fluctuations of the inputs that are random or irrelevant to predicting a correct output
III. That the system parameters be trainable

The standard RNNs fail to learn in the presence of time lags greater than 5 – 10 discrete time steps between relevant input events and target signals. The vanishing error problem casts doubt on whether standard RNNs can indeed exhibit significant practical advantages over time window-based feed forward networks. A recent model, “Long Short-Term Memory” (LSTM), is not affected by this problem. LSTM can learn to bridge minimal time lags in excess of 1000 discrete time steps by enforcing constant error flow through “constant error carousels” (CECs) within special units, called cells .

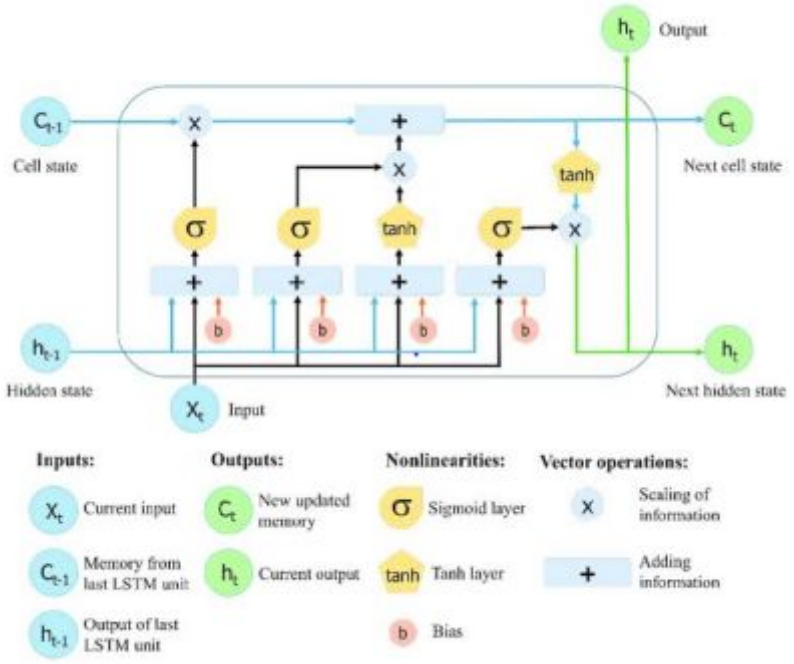


Figure 4.1 : LSTM architecture

Each memory cell's internal architecture guarantees constant error ow within its constant error carousel CEC. This represents the basis for bridging very long time lags. Two gate units learn to open and close access to error now within each memory cell's CEC. The multiplicative input gate affords protection of the CEC from perturbation by irrelevant inputs. Likewise, the multiplicative output gate protects other units from perturbation by currently irrelevant memory contents.

The functions and equations driving an LSTM are :

$$i_t = \delta(x_t u^i + h_{t-1} W^i)$$

$$f_t = \delta(x_t U^t + h_{t-1} W^t)$$

$$o_t = \delta(x_t U^o + h_{t-1} W^o)$$

$$C^0{}_t = \tanh(x_t U^t + h_{t-1} W^g)$$

$$C_t = \delta(f_t C_{t-1} + i_t * C^0{}_t)$$

$$h_t = \tanh(C_t) + o_t$$

The functioning of LSTM can be visualized by understanding the function ing of a news channel's team covering a murder story. Now, a news story is built around facts, evidence and statements of many people. Whenever a new event occurs , take either of the three steps.

Let's say for example, assuming that the murder was done by 'poisoning' the victim, but the autopsy report that just came in said that the cause of death was 'an impact on the head'. Being a part of this news team what is to be done? It is natural to immediately forget the previous cause of death and all stories that were woven around this fact.

What, if an entirely new suspect is introduced into the picture. A person who

had grudges with the victim and could be the murderer?further input this information into the news feed, right?

Now all these broken pieces of information cannot be served on mainstream media. So, after a certain time interval, need to summarize this information and output the relevant things to the audience. Maybe in the form of “XYZ turns out to be the prime suspect.”.

4.1.1. ADVANTAGES

I. Language modelling or text generation, that involves the computation of words when a sequence of words is fed as input. Language models can be operated at the character level, n-gram level, sentence level or even paragraph level.

II. Image processing, that involves performing analysis of a picture and concluding its result into a sentence. For this, it’s required to have a dataset consisting of a good amount of pictures with their corresponding descriptive captions. A model that has already been trained is used to predict features of images present in the dataset. This is photo data. The dataset is then processed in such a way that only the words that are most suggestive are present in it. This is text data. Using these two types of data, try to fit the model. The work of the model is to generate a descriptive sentence for the picture one word at a time by taking input words that were predicted previously by the model and also the image.

III. Speech and Handwriting Recognition

IV. Music generation which is quite similar to that of text generation where LSTMs predict musical notes instead of text by analyzing a combination of given notes fed as input.

V. Language Translation involves mapping a sequence in one language to a sequence in another language. Similar to image processing, a dataset, containing phrases and their translations, is first cleaned and only a part of it is used to train the model. An encoder-decoder LSTM model is used which first converts input sequence to its vector representation (encoding) and then outputs it to its translated version.

4.1.2. LIMITATIONS

LSTMs became popular because they could solve the problem of vanishing gradients. But it turns out, they fail to remove it completely. The problem lies in the fact that the data still has to move from cell to cell for its evaluation. Moreover, the cell has become quite complex now with the additional features (such as forget gates) being brought into the picture.

I. They require a lot of resources and time to get trained and become ready for real-world applications. In technical terms, they need high memory bandwidth because of linear layers present in each cell which the system usually fails to provide for. Thus, hardware-wise, LSTMs become quite inefficient.
II. With the rise of data mining, developers are looking for a model that can remember past information for a longer time than LSTMs. The source of inspiration for such a model is the human habit of dividing a given piece of information into small parts for easy remembrance.
III. LSTMs are prone to overfitting and it is difficult to apply the dropout algorithm to curb this issue. Dropout is a regularization method where in put and recurrent connections to LSTM units are probabilistically excluded from activation and weight updates while training a network.

4.2. RECURRENT AND SEQUENTIAL NETWORKS

Sequential networks are the logic networks with memory. An example is a single input single output network that produces 1 iff three consecutive l's appear in the inputs. Sequential networks are represented by state diagrams or state tables. Flip-flops are used for memory elements. A design of a sequential network is done as follows: First, minimize the number of states. Second, assign a binary code to each state. Third, allocate flip-flops. And, finally, realize the networks for flip-flops and outputs .

Sequence models are the machine learning models that input or output sequences of data. Sequential data includes text streams, audio clips, video clips, time-series data and etc. Recurrent Neural Networks (RNNs) is a popular algorithm

used in sequence models.

One form of Sequential Neural networks are Recurrent Neural networks. Recurrent Neural Network (RNN) is a Deep learning algorithm and it is a type of Artificial Neural Network architecture that is specialized for processing sequential data. RNNs are mostly used in the field of Natural Language Processing (NLP). RNN maintains internal memory, due to this they are very efficient for machine learning problems that involve sequential data. RNNs are also used in time series predictions as well.

The main advantage of using RNNs instead of standard neural networks is that the features are not shared in standard neural networks. Weights are shared across time in RNN. RNNs can remember its previous inputs but Standard Neural Networks are not capable of remembering previous inputs. RNN takes historical information for computation.

$$L(y', y) = \sum_{t=1}^{T} L(y'^{<t>}, y^{<t>})$$

Figure 4.2 : RNN

There are several RNN architectures based on the number of inputs and outputs

I. One to Many Architecture: Image captioning is one good example of this architecture. In image captioning, it takes one image and then outputs a sequence of words. Here there is only one input but many outputs.
II. Many to One Architecture: Sentiment classification is one good example of this architecture. In sentiment classification, a given sentence is classified as positive or negative. In this case, the input is a sequence of words and output is a binary classification
III. Many to Many Architecture: There are two cases in many to many architectures

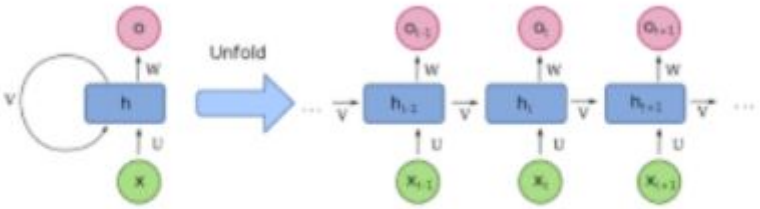


Figure 4.3 : RNN Architecture

4.2.1. ADVANTAGES

I. RNN can process inputs of any length.
II. An RNN model is modeled to remember each information throughout the time which is very helpful in any time series predictor.
III. Even if the input size is larger, the model size does not increase.
IV. The weights can be shared across the time steps.
V. RNN can use their internal memory for processing the arbitrary series of inputs which is not the case with feedforward neural networks.

4.2.2. LIMITATIONS

I. Due to its recurrent nature, the computation is slow.
II. Training of RNN models can be difficult.
III. Using relu or tanh as activation functions, it becomes very difficult to process sequences that are very long.
IV. Prone to problems such as exploding and gradient vanishing.

4.3. DENSE NETWORKS

Dense layer is the regular deeply connected neural network layer. It is the most common and frequently used layer. Dense layer does the below operation on the input and returns the output.

The “Deep” in deep-learning comes from the notion of increased complexity

resulting by stacking several consecutive (hidden) non-linear layers. Here are some graphs of the most famous activation functions like Sigmoid, Relu, Hyperbolic tangent and many more

Obviously, see now that dense layers can be reduced back to linear layers if using a linear activation

Dense layers add an interesting non-linearity property, thus they can model any mathematical function. However, they are still limited in the sense that for the same input vector always get the same output vector. They can't detect repetition in time, or produce different answers on the same input

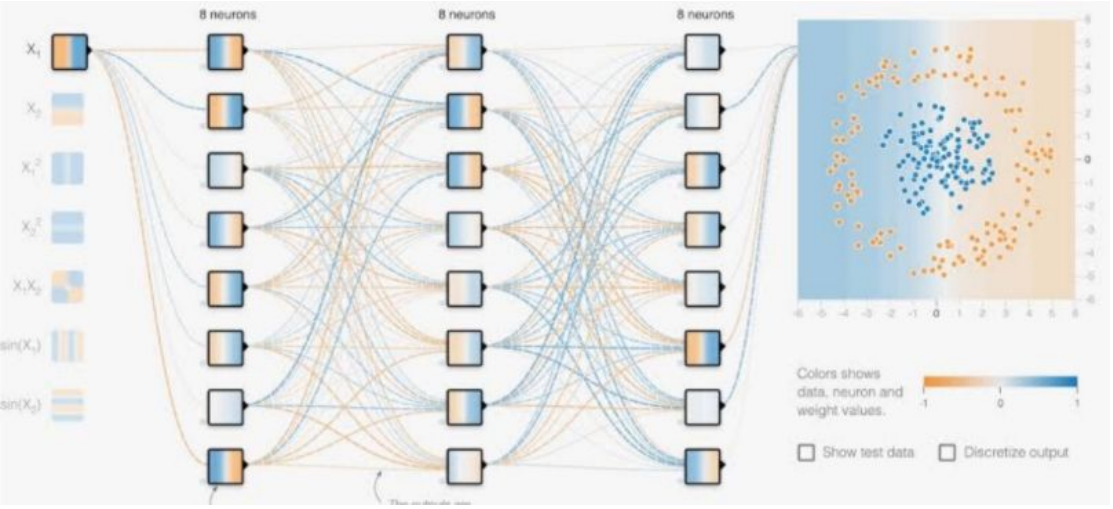


Figure 4.4 : Dense Network

The output shape of the Dense layer will be affected by the number of neu ron units specified in the Dense layer. For example, if the input shape is (8,) and the number of units is 16, then the output shape is (16,). All layers will have batch size as the first dimension and so, input shape will be represented by (None, 8) and the output shape as (None, 16). Currently, batch size is None as it is not set. Batch size is usually set during the training phase.

From an architecture point of view, any single convolution can be replaced by a Dense layer that would perform the same association of neighboring pixels for each pixel. It would mean one neuron per pixel with not-null coefficients only on the neighbors. Convolutional layer is enforcing parameter sharing: the processing of

each pixel is identical by design, not by learning. It means a dramatic reduction in the number of parameters to learn, still with a very good performance.

some of the activation functions are :

1. Sigmoid

$$\delta = \frac{1}{1 + e^{-x}} \tag{1}$$

2. Softmax

$$\phi(z) = ln(1 + e^{z}) \tag{2}$$

3. Hyperbolic Tangent

$$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{3}$$

4. Rectified Linear i.e RELU

$$\phi(z) = max(0, z) \tag{4}$$

Figure 4.5 : Activation Functions in Dense Network

4.3.1. ADVANTAGES

I. A fully connected layer offers learned features from all the combinations of the features of the previous layer, where a convolutional layer relies on local spatial coherence with a small receptive field.

II. Having fault tolerance: Corruption of one or more cells of ANN does not prevent it from generating output. This feature makes the networks fault tolerant

III. Having a distributed memory: In order for ANN to be able to learn, it is necessary to determine the examples and to teach the network according to the desired output by showing these examples to the network. The network's success is directly proportional to the selected instances, and if the event can not be shown to the network in all its aspects, the network can produce false

output.

4.3.2. LIMITATIONS

I. Hardware dependence: Artificial neural networks require processors with parallel processing power, in accordance with their structure. For this reason, the realization of the equipment is dependent.

II. The duration of the network is unknown: The network is reduced to a certain value of the error on the sample means that the training has been completed. This value does not give us optimum results.

III. Unexplained behavior of the network: This is the most important problem of ANN. When ANN produces a probing solution, it does not give a clue as to why and how. This reduces trust in the network.

4.4. FEATURE ENGINEERING- GENERATING LAG FEATURES

The dataset working with has time series data in it. For our model to understand time series data in a better way use what are called ”Lag Features”.

Time Series data must be re-framed as a supervised learning dataset before start using machine learning algorithms. There is no concept of input and output features in time series. Instead, choose the variable to be predicted and use feature engineering to construct all of the inputs that will be used to make predictions for future time steps.

This is also called FEATURE ENGINEERING. The goal of feature engineering is to provide strong and ideally simple relationships between new input features and the output feature for the supervised learning algorithm to model.

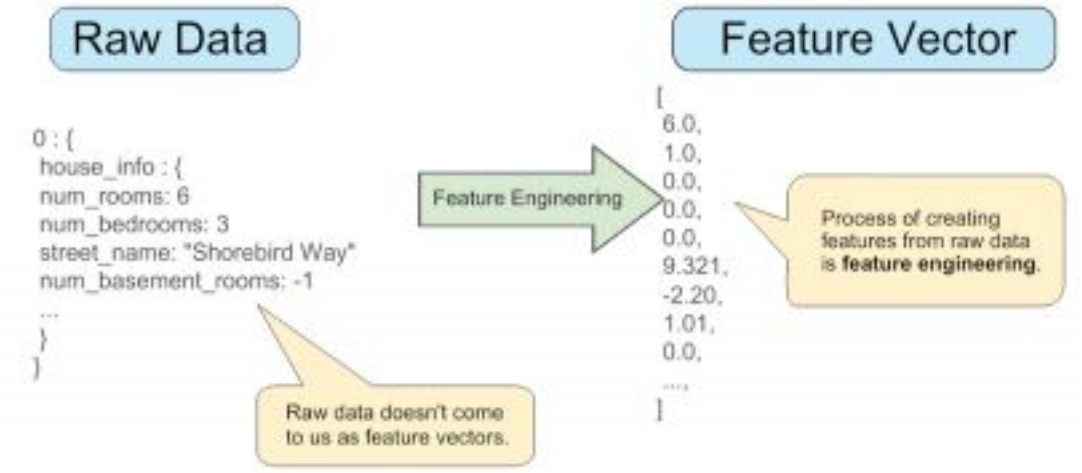


Figure 4.6 : Feature Engineering

Complexity exists in the relationships between the input and output data. In the case of time series, there is no concept of input and output variables; must invent these too and frame the supervised learning problem from scratch.

A lag features is a fancy name for a variable which contains data from prior time steps. For time-series data, convert it into rows. Every row contains data about one observation and includes all previous occurrences of that observation.

To generate Lag features the steps required are :

1. Select only the time-series data related to that one observation.
2. Extract all values of the time-series variables
3. Shift the target variables five times to get five lag features and the new dependent feature
4. Shift the other time-series variable six times to get all lag values of that independent feature.
5. Copy the non-time-series variables.
6. Split the data frame into the independent features and the dependent features.
7. Store them in arrays that will be used later for feature scaling, splitting into training/validation/test sets, and finally for the training of a model.

4.5. RELU AND ADAM FUNCTIONS

The rectified linear is the activation function used in our code and the Adam function is the optimiser.

4.5.1. RECTIFIED LINEAR FUNCTION

The Rectified Linear Function(RELU) goes as :

$$\varphi(z) = max(0, z)$$

In a neural network, the activation function is responsible for transforming the summed weighted input from the node into the activation of the node or output for that input.

The rectified linear activation function or ReLU for short is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance.

In order to use stochastic gradient descent with backpropagation of errors to train deep neural networks, an activation function is needed that looks and acts like a linear function, but is, in fact, a nonlinear function allowing complex relationships in the data to be learned.

A node or unit that implements this activation function is referred to as a rectified linear activation unit. Often, networks that use the rectifier function for the hidden layers are referred to as rectified networks.

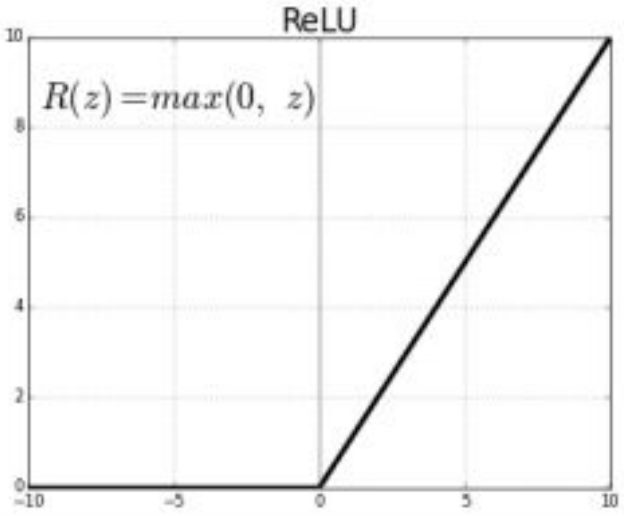


Figure 4.7 : RELU Function

Because rectified linear units are nearly linear, they preserve many of the properties that make linear models easy to optimize with gradient-based methods. They also preserve many of the properties that make linear models generalize well.

4.5.2 ADAM OPTIMIZER

The Adam optimization algorithm is an extension to stochastic gradient descent that has recently seen broader adoption for deep learning applications in computer vision and natural language processing.

The choice of optimization algorithm for the deep learning model can mean the difference between good results in minutes, hours, and days.

Adam is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights iteratively based on training data.

Adam is used since it is :

1. Straightforward to implement.
2. Computationally efficient.

3. Little memory requirements.
4. Invariant to diagonal rescale of the gradients.
5. Well suited for problems that are large in terms of data and/or parameters.
6. Appropriate for non-stationary objectives.
7. Appropriate for problems with very noisy/or sparse gradients.
8. Hyper-parameters have intuitive interpretation and typically require little tuning.

Adam is different to classical stochastic gradient descent. Stochastic gradient descent maintains a single learning rate (termed alpha) for all weight updates and the learning rate does not change during training. A learning rate is maintained for each network weight (parameter) and separately adapted as learning unfolds.

The method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients.

Adaptive Gradient Algorithm (AdaGrad) that maintains a per-parameter learning rate that improves performance on problems with sparse gradients (e.g. natural language and computer vision problems).

Root Mean Square Propagation (RMSProp) that also maintains per-parameter learning rates that are adapted based on the average of recent magnitudes of the gradients for the weight (e.g. how quickly it is changing). This means the algorithm does well on online and non-stationary problems (e.g. noisy).

For each parameter

$$w^j$$

$$v_t = \rho\, v_{t-1} + (1-\rho) * g_t^2$$

$$\delta w_t = -\frac{\eta}{\sqrt{v_t + \epsilon}} * g_t$$

$$w_{t+1} = w_{t-1} + \delta\, w_t$$

Figure 4.8 : Adam formulas

Adam can be looked at as a combination of RMSprop and Stochastic

Gradient Descent with momentum. It uses the squared gradients to scale the learning rate like RMSprop and it takes advantage of momentum by using moving average of the gradient instead of gradient itself like SGD with momentum.

Figure 4.9 : ADAM Optimizer

Adam is an adaptive learning rate method, which means, it computes individual learning rates for different parameters. Its name is derived from adaptive moment estimation, and the reason it's called that is because Adam uses estimations of first and second moments of gradient to adapt the learning rate for each weight of the neural network.

5. IMPLEMENTATION

This chapter is about how projects were implemented in python using machine learning. Executed in jupyter notebook through Anaconda IDE. predictions are depicted using MatplotLib

Dataset collected from Kaggle, a subsidiary of Google LLC, is an online community of data scientists and machine learning practitioners.

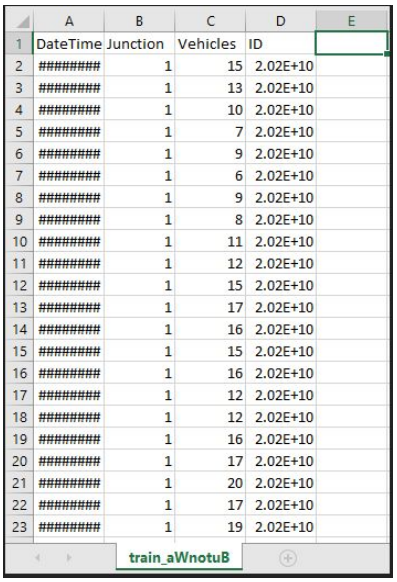
5.1. Dataset

Use sensors on each of these junctions where data is collected at different times, hence there will be traffic data from different time periods.

There are 48120 instances of training data (data each hour from 2015-11-01 to 2017-06-30 for 4 junctions) And 11808 instances of testing data.

Columns in Dataset :

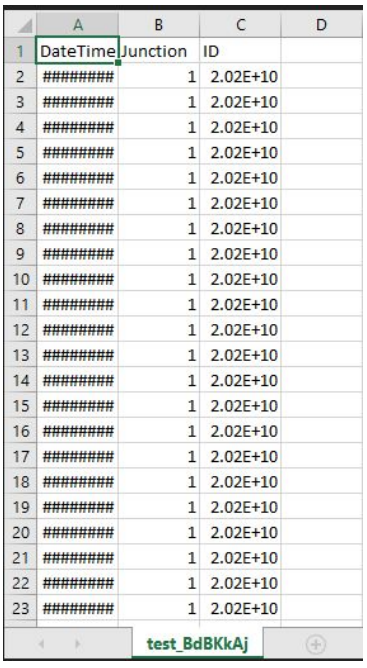
DateTime
Junction
Vehicles
ID



	A	B	C	D	E
1	DateTime	Junction	Vehicles	ID	
2	########	1	15	2.02E+10	
3	########	1	13	2.02E+10	
4	########	1	10	2.02E+10	
5	########	1	7	2.02E+10	
6	########	1	9	2.02E+10	
7	########	1	6	2.02E+10	
8	########	1	9	2.02E+10	
9	########	1	8	2.02E+10	
10	########	1	11	2.02E+10	
11	########	1	12	2.02E+10	
12	########	1	15	2.02E+10	
13	########	1	17	2.02E+10	
14	########	1	16	2.02E+10	
15	########	1	15	2.02E+10	
16	########	1	16	2.02E+10	
17	########	1	12	2.02E+10	
18	########	1	12	2.02E+10	
19	########	1	16	2.02E+10	
20	########	1	17	2.02E+10	
21	########	1	20	2.02E+10	
22	########	1	17	2.02E+10	
23	########	1	19	2.02E+10	

train_aWnotuB

Screen 1: Training set



	A	B	C	D
1	DateTime	Junction	ID	
2	########	1	2.02E+10	
3	########	1	2.02E+10	
4	########	1	2.02E+10	
5	########	1	2.02E+10	
6	########	1	2.02E+10	
7	########	1	2.02E+10	
8	########	1	2.02E+10	
9	########	1	2.02E+10	
10	########	1	2.02E+10	
11	########	1	2.02E+10	
12	########	1	2.02E+10	
13	########	1	2.02E+10	
14	########	1	2.02E+10	
15	########	1	2.02E+10	
16	########	1	2.02E+10	
17	########	1	2.02E+10	
18	########	1	2.02E+10	
19	########	1	2.02E+10	
20	########	1	2.02E+10	
21	########	1	2.02E+10	
22	########	1	2.02E+10	
23	########	1	2.02E+10	

test_BdBKkAj

Screen 2: Test set

5.1.1. IMPORTING DATASET WITH INFORMATION

Pandas in python provide an interesting method read_csv(). The read_csv function reads the entire dataset from a comma separated values file and can assign it to a DataFrame to which all the operations can be performed. It helps us to access

each and every row as well as columns and each and every value can be accessed using the dataframe. Any missing value or NaN value have to be cleaned.

Loading the dataset ¶

```
#read the dataset into dataframes, training and testing set respectively
df_train = pd.read_csv('train_aWnotuB.csv', parse_dates=[0], infer_datetime_format=True)
df_test = pd.read_csv('test_BdBKkAj.csv', parse_dates=[0], infer_datetime_format=True)
```

Screen 3: Dataset upload

```
df_train.head()
```

	DateTime	Junction	Vehicles	ID
0	2015-11-01 00:00:00	1	15	20151101001
1	2015-11-01 01:00:00	1	13	20151101011
2	2015-11-01 02:00:00	1	10	20151101021
3	2015-11-01 03:00:00	1	7	20151101031
4	2015-11-01 04:00:00	1	9	20151101041

Screen 4: Train set top 5 rows

```
df_train.describe()
```

	Junction	Vehicles	ID
count	48120.000000	48120.000000	4.812000e+04
mean	2.180549	22.791334	2.016330e+10
std	0.966955	20.750063	5.944854e+06
min	1.000000	1.000000	2.015110e+10
25%	1.000000	9.000000	2.016042e+10
50%	2.000000	15.000000	2.016093e+10
75%	3.000000	29.000000	2.017023e+10
max	4.000000	180.000000	2.017063e+10

Screen 5: describe train set

```
df_test.describe()
```

	Junction	ID
count	11808.000000	1.180800e+04
mean	2.500000	2.017087e+10
std	1.118081	1.124665e+05
min	1.000000	2.017070e+10
25%	1.750000	2.017073e+10
50%	2.500000	2.017083e+10
75%	3.250000	2.017100e+10
max	4.000000	2.017103e+10

Screen 6: describe test set

5.1.2. EXPLORATORY DATA ANALYSIS

Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns,to spot anomalies,to test hypotheses and to check assumptions with the help of summary statistics and graphical representations.

It is important to know the EDA of the dataset used in order to proceed further in the project.

5.2. JUPYTER NOTEBOOK

The **Jupyter Notebook** is an open source web application that can be used to create and share documents that contain live code, equations, visualizations, and text. **Jupyter Notebooks** are a spin-off project from the **IPython** project, which used to have an **IPython Notebook** project itself.

The name, Jupyter, comes from the core supported programming languages that it supports: Julia, Python, and R. Jupyter ships with the IPython kernel, which allows to write programs in Python, but there are currently over 100 other kernels that can also be used. Install Jupyter using Anaconda and conda. Recommend downloading Anaconda's latest Python 3 version (currently Python 3.5).

Figure 5.1: Jupyter Logo

5.3. NUMPY

NumPy is a Python library used for working with arrays. It also has functions for working in the domain of linear algebra, fourier transform, and matrices.

` **NumPy** stands for Numerical Python.In Python there are lists that serve the purpose of arrays, but they are slow to process.NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.

The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.

Arrays are very frequently used in data science, where speed and resources are very important.

NumPy arrays are stored at one continuous place in memory unlike lists, so processes can access and manipulate them very efficiently.This behavior is called locality of reference in computer science.This is the main reason why NumPy is faster than lists. Also it is optimized to work with the latest CPU architectures.

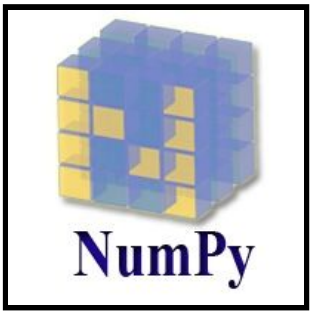


Figure 5.2: Numpy Logo

5.4. PANDAS

Pandas is a software library written for the **Python** programming language for data manipulation and analysis. In particular, it offers data structures and operations

for manipulating numerical tables and time series. It is free software released under the three-clause BSD license.

The name is derived from the term”panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals. Its name is a play on the phrase "Python data analysis" itself. Pandas is mainly used for data analysis. Pandas allows importing data from various file formats such as comma-separated values, JSON, SQL, Microsoft Excel.

Figure 5.3: Pandas Logo

5.5. KERAS

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result as fast as possible is key to doing good research.

The core data structures of Keras are **layers** and **models**. The simplest type of model is the Sequential model, a linear stack of layers. For more complex architectures, use the Keras functional API, which allows to build arbitrary graphs of layers, or write models entirely from scratch via subclassing.

Keras comes packaged with TensorFlow 2.0 as tensorflow.keras. To start using Keras, simply install TensorFlow 2.0.

Keras/TensorFlow are compatible with:

- Python 3.5–3.8
- Ubuntu 16.04 or later
- Windows 7 or later
- macOS 10.12.6 (Sierra) or later.

Figure 5.4: Keras Logo

5.6. TENSORFLOW

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.

TensorFlow can train and run deep neural networks for handwritten digit classification, image recognition, word embeddings, recurrent neural networks, sequence-to-sequence models for machine translation, natural language processing, and PDE (partial differential equation) based simulations. Best of all, TensorFlow supports production prediction at scale, with the same models used for training.

TensorFlow allows developers to create *dataflow graphs*—structures that describe how data moves through a graph, or a series of processing nodes. Each node in the graph represents a mathematical operation, and each connection or edge between nodes is a multidimensional data array, or *tensor*.TensorFlow provides all of this for the programmer by way of the Python language.

Python is easy to learn and work with, and provides convenient ways to express how high-level abstractions can be coupled together. Nodes and tensors in TensorFlow are Python objects, and TensorFlow applications are themselves Python applications.The actual math operations, however, are not performed in Python.

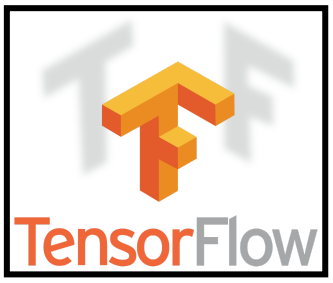


Figure 5.5: TensorFlow Logo

5.7. MATPLOTLIB

"matplotlib.pyplot" is a collection of functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

In matplotlib.pyplot various states are preserved across function calls, so that it keeps track of things like the current figure and plotting area, and the plotting functions are directed to the current axes (please note that "axes" here and in most places in the documentation refers to the *axes* part of a figure and not the strict mathematical term for more than one axis).

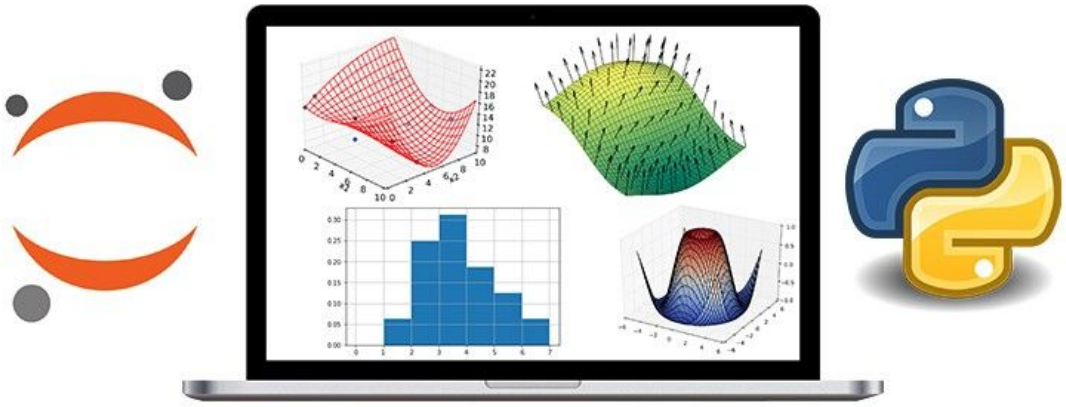


Figure 5.6:Matplotlib

6. TESTING AND VALIDATION

In this chapter, the testing and validation of the project is done by describing the following section:

6.1. CREATE TIME TARGETS

Time targets are created in the EDA which is described in the previous chapter , section 5.1.2.

EDA - Exploratory Data Analysis

```
In [13]: df_tmp = df_train.set_index(['Junction', 'DateTime'])

In [14]: level_values = df_tmp.index.get_level_values

In [15]: time_targets = df_tmp.groupby([level_values(0)] + [pd.Grouper(freq='1M', level=-1)])['Vehicles'].sum()
         time_targets

Out[15]: Junction  DateTime
         1         2015-11-30    14736
                   2015-12-31    15487
                   2016-01-31    17940
                   2016-02-29    20813
                   2016-03-31    22215
                                  ...
         4         2017-02-28     5564
                   2017-03-31     4931
                   2017-04-30     4454
                   2017-05-31     4877
                   2017-06-30     6097
         Name: Vehicles, Length: 66, dtype: int64
```

Screen 7: EDA

6.2. GENERATE LAG FEATURES

Lag features is used to make a data frame which stores the value of different time stamps as rows. It's a more intelligent way to represent time data.

For example, if there are 60 values of traffic lights from 12:00 to 1pm, every minute the data set continues the traffic light value. So now create a table also called as a data-frame. In this table, map each minute to the corresponding value.

Data frames are very easy for processing instead of individual time stamps. As seen in the code. This created a data frame using the values which are time dependent.

```
Xy_train = gen_lag_features(train)
Xy_train
```

DateTime	Junction 1 (H-1)	Junction 2 (H-1)	Junction 3 (H-1)	Junction 4 (H-1)	Junction 1 (H)	Junction 2 (H)	Junction 3 (H)	Junction 4 (H)
2015-11-01 01:00:00	15.0	6.0	9.0	0.0	13.0	6.0	7.0	0.0
2015-11-01 02:00:00	13.0	6.0	7.0	0.0	10.0	5.0	5.0	0.0
2015-11-01 03:00:00	10.0	5.0	5.0	0.0	7.0	6.0	1.0	0.0
2015-11-01 04:00:00	7.0	6.0	1.0	0.0	9.0	7.0	2.0	0.0
2015-11-01 05:00:00	9.0	7.0	2.0	0.0	6.0	2.0	2.0	0.0
...	...	...	...	...	...	...	...	...
2017-06-30 19:00:00	95.0	34.0	38.0	17.0	105.0	34.0	33.0	11.0
2017-06-30 20:00:00	105.0	34.0	33.0	11.0	96.0	35.0	31.0	30.0
2017-06-30 21:00:00	96.0	35.0	31.0	30.0	90.0	31.0	28.0	16.0
2017-06-30 22:00:00	90.0	31.0	28.0	16.0	84.0	29.0	26.0	22.0
2017-06-30 23:00:00	84.0	29.0	26.0	22.0	78.0	27.0	39.0	12.0

14591 rows × 8 columns

Screen 8: Lag Features

The output shows the table the values of real values (Junctions (H-1)) and predicted values (Junctions (H)) for the timeseries.

6.3. NORMALISE FEATURES

Normalization is a rescaling of the data from the original range so that all values are within the new range of 0 and 1.

Normalization requires that it is known or is able to accurately estimate the minimum and maximum observable values.

```
from sklearn.preprocessing import MinMaxScaler, StandardScaler
# scaler = MinMaxScaler(feature_range=(0, 1)), MinMaxScaler subtracts the minimum value in the feature and
#then divides by the range. The range is the difference between the original maximum and original minimum.
#MinMaxScaler preserves the shape of the original distribution.

# scaler = StandardScaler(), Fit to data, then transform it.
#Fits transformer to X and y with optional parameters fit_params and returns a transformed version of X.
scaler = MinMaxScaler(feature_range=(0, 1))
Xy_train[Xy_train.columns] = scaler.fit_transform(Xy_train[Xy_train.columns])

Xy_train
```

DateTime	Junction 1 (H-1)	Junction 2 (H-1)	Junction 3 (H-1)	Junction 4 (H-1)	Junction 1 (H)	Junction 2 (H)	Junction 3 (H)	Junction 4 (H)
2015-11-01 01:00:00	0.066225	0.106383	0.044693	0.000000	0.052980	0.106383	0.033520	0.000000
2015-11-01 02:00:00	0.052980	0.106383	0.033520	0.000000	0.033113	0.085106	0.022346	0.000000
2015-11-01 03:00:00	0.033113	0.085106	0.022346	0.000000	0.013245	0.106383	0.000000	0.000000
2015-11-01 04:00:00	0.013245	0.106383	0.000000	0.000000	0.026490	0.127660	0.005587	0.000000
2015-11-01 05:00:00	0.026490	0.127660	0.005587	0.000000	0.006623	0.021277	0.005587	0.000000
...	...	...	...	...	...	...	...	...
2017-06-30 19:00:00	0.596026	0.702128	0.206704	0.472222	0.662252	0.702128	0.178771	0.305556
2017-06-30 20:00:00	0.662252	0.702128	0.178771	0.305556	0.602649	0.723404	0.167598	0.833333
2017-06-30 21:00:00	0.602649	0.723404	0.167598	0.833333	0.562914	0.638298	0.150838	0.444444
2017-06-30 22:00:00	0.562914	0.638298	0.150838	0.444444	0.523179	0.595745	0.139665	0.611111
2017-06-30 23:00:00	0.523179	0.595745	0.139665	0.611111	0.483444	0.553191	0.212291	0.333333

14591 rows × 8 columns

Screen 9: Normalization

6.4. SPLIT THE DATA

After normalizing the data, it is split to differentiate the real values from the predicted values.

Real values are:

```
X_train = Xy_train[Xy_train.index < '2017-04-01'].iloc[:,0:4]
X_train
```

DateTime	Junction 1 (H-1)	Junction 2 (H-1)	Junction 3 (H-1)	Junction 4 (H-1)
2015-11-01 01:00:00	0.066225	0.106383	0.044693	0.000000
2015-11-01 02:00:00	0.052980	0.106383	0.033520	0.000000
2015-11-01 03:00:00	0.033113	0.085106	0.022346	0.000000
2015-11-01 04:00:00	0.013245	0.106383	0.000000	0.000000
2015-11-01 05:00:00	0.026490	0.127660	0.005587	0.000000
...	...	...	...	...
2017-03-31 19:00:00	0.476821	0.574468	0.178771	0.166667
2017-03-31 20:00:00	0.496689	0.531915	0.156425	0.222222
2017-03-31 21:00:00	0.483444	0.638298	0.156425	0.222222
2017-03-31 22:00:00	0.403974	0.574468	0.150838	0.250000
2017-03-31 23:00:00	0.423841	0.553191	0.162011	0.166667

12407 rows × 4 columns

Screen 10: After normalization -Real Values

Predicted values are:

```
y_train = Xy_train[Xy_train.index < '2017-04-01'].iloc[:,4:]
y_train
```

DateTime	Junction 1 (H)	Junction 2 (H)	Junction 3 (H)	Junction 4 (H)
2015-11-01 01:00:00	0.052980	0.106383	0.033520	0.000000
2015-11-01 02:00:00	0.033113	0.085106	0.022346	0.000000
2015-11-01 03:00:00	0.013245	0.106383	0.000000	0.000000
2015-11-01 04:00:00	0.026490	0.127660	0.005587	0.000000
2015-11-01 05:00:00	0.006623	0.021277	0.005587	0.000000
...	...	...	...	...
2017-03-31 19:00:00	0.496689	0.531915	0.156425	0.222222
2017-03-31 20:00:00	0.483444	0.638298	0.156425	0.222222
2017-03-31 21:00:00	0.403974	0.574468	0.150838	0.250000
2017-03-31 22:00:00	0.423841	0.553191	0.162011	0.166667
2017-03-31 23:00:00	0.417219	0.553191	0.162011	0.166667

12407 rows × 4 columns

Screen 11: After normalization -Predicted Values

6.5. RESHAPE THE DATA

Reshaping of data is required in this project since the dataset is a time series data containing variables with some kinds of sequences.

Reshape the Data

In [24]:
```
print(X_train.shape, y_train.shape)
```

```
(12407, 4) (12407, 4)
```

In [25]:
```
X_train = np.expand_dims(X_train.values, axis=2)
print(X_train.shape)

y_train = y_train.values
print(y_train.shape)
```

```
(12407, 4, 1)
(12407, 4)
```

Screen 12: Reshape data

6.6. VALIDATION

Validation is done for reassuring that the data taken is accurate and can be used for further prediction.

```
X_valid = Xy_train[Xy_train.index >= '2017-04-01'].iloc[:,0:4]
X_valid
```

DateTime	Junction 1 (H-1)	Junction 2 (H-1)	Junction 3 (H-1)	Junction 4 (H-1)
2017-04-01 00:00:00	0.417219	0.553191	0.162011	0.166667
2017-04-01 01:00:00	0.384106	0.510638	0.122905	0.166667
2017-04-01 02:00:00	0.317881	0.574468	0.078212	0.138889
2017-04-01 03:00:00	0.238411	0.361702	0.083799	0.111111
2017-04-01 04:00:00	0.225166	0.361702	0.055866	0.111111
...	...	...	...	...
2017-06-30 19:00:00	0.596026	0.702128	0.206704	0.472222
2017-06-30 20:00:00	0.662252	0.702128	0.178771	0.305556
2017-06-30 21:00:00	0.602649	0.723404	0.167598	0.833333
2017-06-30 22:00:00	0.562914	0.638298	0.150838	0.444444
2017-06-30 23:00:00	0.523179	0.595745	0.139665	0.611111

2184 rows × 4 columns

```
X_valid = np.expand_dims(X_valid.values, axis=2)
y_pred = regressor.predict(X_valid)
```

```
# We rescale y in the integer count range
# To do that we must first reconcatenate with the X data as scaler expects a shape of 8

y_pred = scaler.inverse_transform(np.concatenate((X_valid.squeeze(), y_pred), axis = 1))[:, 4:]
y_pred
```

```
array([[71.31680802, 24.52757156, 23.5383566 ,  7.20727265],
       [65.90196571, 22.78899035, 18.97216295,  6.8159287 ],
       [54.63330489, 23.50201994, 11.80922507,  5.35743463],
       ...,
       [84.53445709, 32.71266031, 30.90840027, 18.04449034],
       [89.19468713, 28.00408405, 24.26491259, 12.31003904],
       [82.36936259, 26.17880315, 26.32577333, 14.99347436]])
```

Screen 13: Validation

6.7. CREATE AND TRAIN MODEL

This is the most important and is the last step to do in order to get results for the prediction.

All the described models and algorithms in chapter 4 are used to create the model for this project along with RMSE.

RMSE or Root Mean Square Error provides the error value for the predicted data.

To fit the model to the data, "**epoch**" - epoch is used to train the model by fitting it many number of times as mentioned to the data in order to increase accuracy of the prediction with a true result. This can be achieved with the help of regression analysis..

Regression analysis consists of a set of machine learning methods that allow us to predict a continuous outcome variable (y) based on the value of one or multiple predictor variables (x). Briefly, the goal of a regression model is to build a mathematical equation that defines y as a function of the x variables. Next, this equation can be used to predict the outcome (y) on the basis of new values of the predictor variables (x).

```
Epoch 1/20
97/97 [==============================] - 0s 4ms/step - loss: 0.0601
Epoch 2/20
97/97 [==============================] - 0s 4ms/step - loss: 0.0389
Epoch 3/20
97/97 [==============================] - 0s 5ms/step - loss: 0.0371
Epoch 4/20
97/97 [==============================] - 0s 4ms/step - loss: 0.0358
Epoch 5/20
97/97 [==============================] - 1s 5ms/step - loss: 0.0345
Epoch 6/20
97/97 [==============================] - 1s 6ms/step - loss: 0.0335
Epoch 7/20
97/97 [==============================] - 1s 6ms/step - loss: 0.0330
Epoch 8/20
97/97 [==============================] - 0s 4ms/step - loss: 0.0328
Epoch 9/20
97/97 [==============================] - 0s 4ms/step - loss: 0.0327
Epoch 10/20
97/97 [==============================] - 0s 5ms/step - loss: 0.0326
Epoch 11/20
97/97 [==============================] - 0s 4ms/step - loss: 0.0325
Epoch 12/20
97/97 [==============================] - 0s 5ms/step - loss: 0.0325
Epoch 13/20
97/97 [==============================] - 0s 4ms/step - loss: 0.0324
Epoch 14/20
97/97 [==============================] - 0s 5ms/step - loss: 0.0323
Epoch 15/20
97/97 [==============================] - ETA: 0s - loss: 0.032 - 1s 6ms/step - loss: 0.0324
Epoch 16/20
97/97 [==============================] - 1s 6ms/step - loss: 0.0322
Epoch 17/20
97/97 [==============================] - 1s 7ms/step - loss: 0.0323
Epoch 18/20
97/97 [==============================] - 0s 5ms/step - loss: 0.0323A: 0s - loss:
Epoch 19/20
97/97 [==============================] - 1s 5ms/step - loss: 0.0322
Epoch 20/20
97/97 [==============================] - 1s 6ms/step - loss: 0.0322

<tensorflow.python.keras.callbacks.History at 0x28d6563e040>
```

Screen 14:Epoch

The next chapter will show the results of the above done steps and used algorithms.

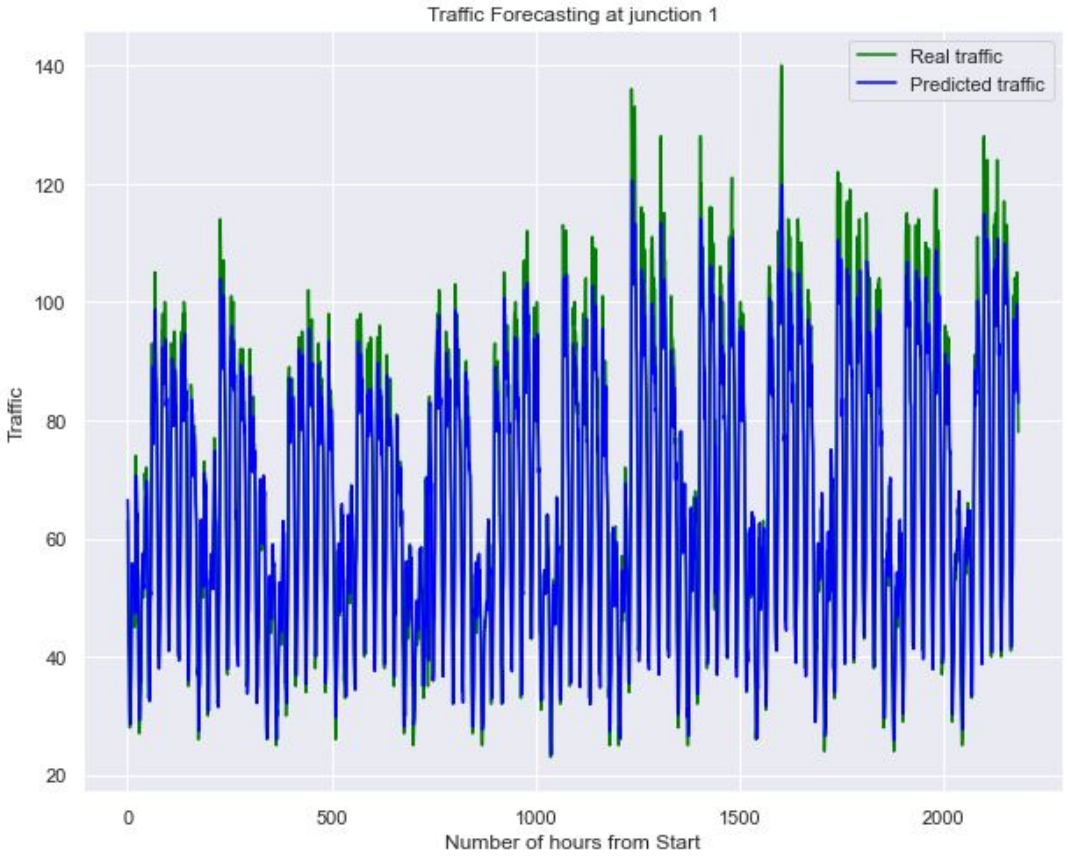
7. RESULT ANALYSIS

This chapter will show the results of the prediction made using matplotlib at 4 junctions where green represents real traffic and blue represents predicted traffic.

In [33]:
```
# Visualising Result for the junctions
for junction in range(4):
    plt.figure
    plt.plot(y_truth.values[:,junction], color = 'green', label = 'Real traffic')
    plt.plot(y_pred[:,junction], color = 'blue', label = 'Predicted traffic')
    plt.title('Traffic Forecasting at junction %i' % (junction+1))
    plt.xlabel('Number of hours from Start')
    plt.ylabel('Traffic')
    plt.legend()
    plt.show()
```

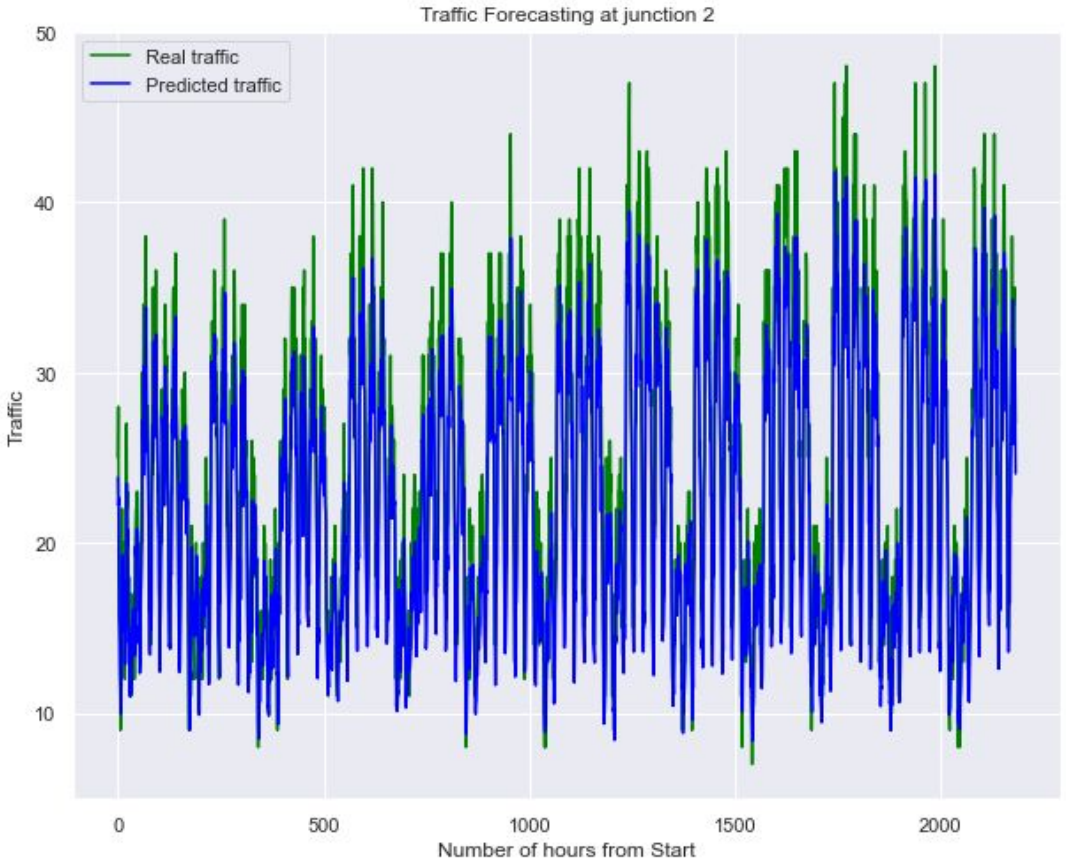
Screen 15: Code for plotting the prediction

7.1. PREDICTION FOR JUNCTION 1



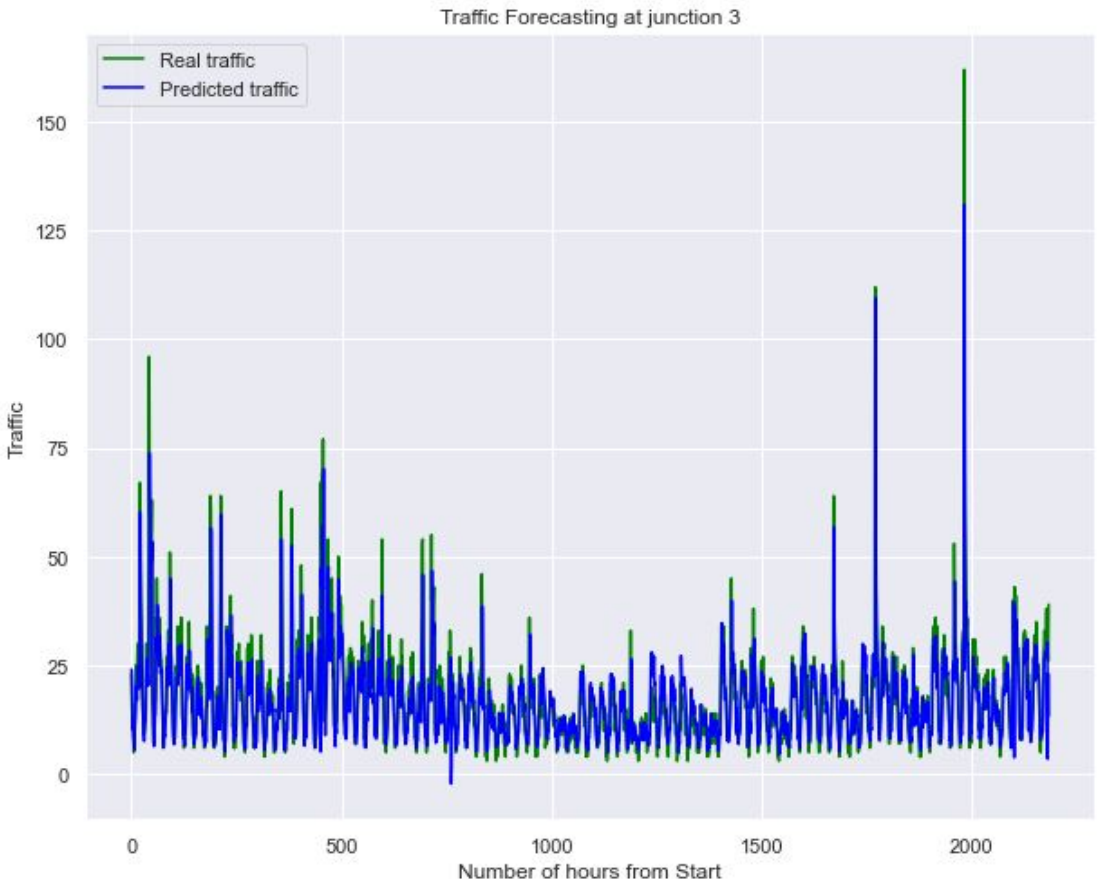
Screen 16: Junction 1

7.2. PREDICTION FOR JUNCTION 2



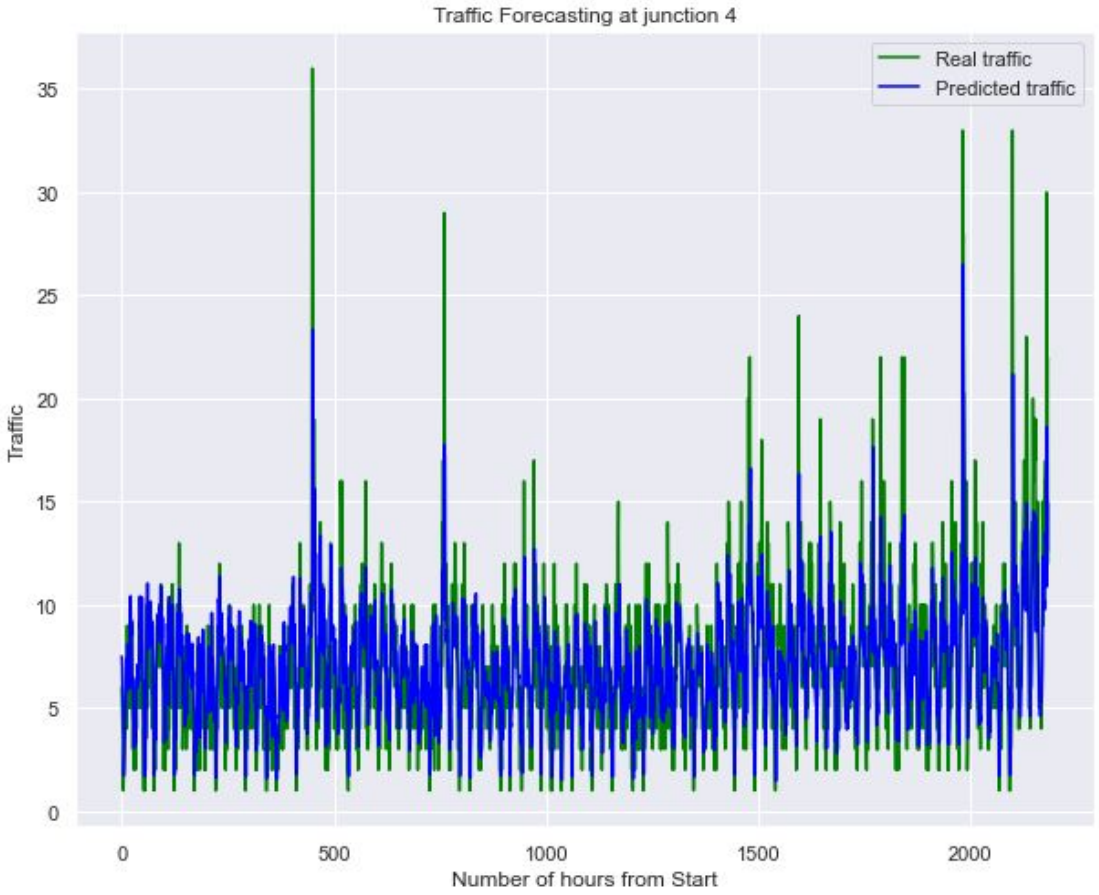
Screen 17: Junction 2

7.3. PREDICTION FOR JUNCTION 3



Screen 18: Junction 3

7.4. PREDICTION FOR JUNCTION 4



Screen 19: Junction 4

7.5. RMSE VALUE

```
In [32]: from sklearn.metrics import mean_squared_error
         from math import sqrt

         def rmse(y_true, y_pred):
            return sqrt(mean_squared_error(y_true, y_pred))
         rmse(y_truth, y_pred)

Out[32]: 5.846986416264791
```

Screen 20: RMSE Value

8. CONCLUSION AND FUTURE WORK

It is clear that machine learning has great potential when it comes to time series forecasting. This has been shown in this project as well as in other referenced literature. Existing statistical approaches should however not be underestimated. The baseline methods did in fact achieve decent results and are faster to evaluate compared to the ML techniques.

The future work for this project will be using models like ARIMA and other CNN models to complete traffic forecasting for months to come.Furthermore, help create an Intelligent Transportation System for the development of Smart City of the future.

REFERENCES

[1] Fei-Yue Wang et al. Parallel control and management for intelligent transportation systems: Concepts, architectures, and applications. IEEE Transactions on Intelligent Transportation Systems, 2010.

[2] Yongchang Ma, Mashrur Chowdhury, Mansoureh Jeihani, and Ryan Fries. Accelerated incident detection across transportation networks using vehicle kinetics and support vector machines in cooperation with infrastructure agents. IET intelligent transport systems, 4(4):328–337, 2010.

[3] Rutger Claes, Tom Holvoet, and Danny Weyns. A decentralized approach for anticipatory vehicle routing using delegate multiagent systems. IEEE Transactions on Intelligent Transportation Systems, 12(2):364–373, 2011.

[4] Mehul Mahrishi and Sudha Morwal.Index point detection and semantic indexing of videos - a comparative review. Advances in Intelligent Systems and Computing, Springer, 2020.

[5] Joseph D Crabtree and Nikiforos Stamatiadis. Dedicated short-range communications technology for freeway incident detection: Performance assessment based on traffic simulation data. Transportation Research Record, 2000(1):59–69, 2007.

[6] H Qi, RL Cheu, and DH Lee. Freeway incident detection using kinematic data from probe vehicles. In 9th World Congress on Intel- ligent Transport SystemsITS America, ITS Japan, ERTICO (Intelligent Transport Systems and Services-Europe), 2002.

[7] Z. Zhao, W. Chen, X. Wu, P. C. Y. Chen, and J. Liu. Lstm network: a deep learning approach for short-term traffic forecast. IET Intelligent Transport Systems, 11(2):68–75, 2017.

[8] C. Zhang, P. Patras, and H. Haddadi. Deep learning in mobile and wireless networking: A survey. IEEE Communications Surveys Tutorials, 21(3):2224–2287,

third quarter 2019.

[9] Chun-Hsin Wu, Jan-Ming Ho, and D. T. Lee. Travel-time prediction with Support vector regression. IEEE Transactions on Intelligent Transportation Systems, 5(4):276–281, Dec 2004.

[10] Yan-Yan Song and LU Ying. Decision tree methods: applications for classification and prediction. Shanghai archives of psychiatry, 27(2):130, 2015.

[11] Yiming He, Mashrur Chowdhury, Yongchang Ma, and Pierluigi Pisu. Merging mobility and energy vision with hybrid electric vehicles and vehicle infrastructure integration. Energy Policy, 41:599–609, 2012.

[12] Jason Brownlee. Bagging and random forest ensemble algorithms for machine learning. Machine Learning Algorithms, pages 4–22, 2016.

Base paper : https://ieeexplore.ieee.org/document/9091758
Kaggle dataset : https://www.kaggle.com/utathya/smart-city-traffic-patterns
https://en.wikipedia.org/wiki/Smart_city