AN INTEGRATED AND DYNAMIC COMMUTER FLOW PROJECTION BASED ON LSTM

A MAJOR PROJECT REPORT

Submitted by

D.NIVAS GANDHI [Reg No: RA1811003020251]
S.RAJESH REDDY [Reg No: RA1811003020258]
G.AARTHI[Reg No: RA1811003020211]

Under the Guidance of

Dr ALAGUMANI S

(Assistant Professor, Department of Computer Science and Engineering)

In Fulfillment For The Award Of The Degree

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

RAMAPURAM - 600 089

MAY 2022

RAMAPURAM

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Deemed to be University U/S 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this B.Tech project report titled "**AN INTEGRATED AND DYNAMIC COMMUTER FLOW PROJECTION BASED ON LSTM"** is the bonafide work of **Mr.D.NIVAS GANDHI , Mr.S.RAJESH REDDY and Ms. G.AARTHI** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion for this or any other candidate.

Dr ALAGUMANI S M.E.,Ph.D.,	**Dr K.RAJA,M.E.,Ph.D.,**
Assistant Professor	Professor and Head
Computer and Engineering	Computer Science and Engineering
SRM Institute of Science and Technology	SRM Institute of Science and Technology
Ramapuram Campus, Chennai.	Ramapuram Campus, Chennai.

Submitted for the project viva voice held on 11-05-2022 at SRM Institution of Science and Technology, Ramapuram Campus, Chennai - 600089

Signature of Internal Examiner **Signature of External Examiner**

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

RAMAPURAM, CHENNAI - 600089

DECLARATION

We hereby declare that the entire work contained in this project report titled "AN INTEGRATED AND DYNAMIC COMMUTER FLOW PROJECTION BASED ON LSTM" has been carried out by DATTI.NIVAS GANDHI -RA1811003020251, S.RAJESH REDDY-RA1811003020258, G. AARTHI-RA1811003020211 at SRM Institute of Science and Technology Ramapuram Campus, Chennai - 600089, under the guidance of Dr.ALAGUMANI.S, Assistant Professor, Department of Computer Science and Engineering.

Place: Chennai

Date: 11-05-2022

D NIVAS GANDHI

S RAJESH REDDY

G AARTHI

ACKNOWLEDGEMENT

We place on record our deep sense of gratitude to our lionized Chairman **Dr. R.SHIVAKUMAR** for providing us with the requisite infrastructure throughout the course.

We take the opportunity to extend our hearty and sincere thanks to our Dean, **Dr.M.MURALI KRISHNA ,B.E., M.TECH., Ph.D. MISTEE,FIE,C.Engg.,** for manoeuvring us into accomplishing the project.

We take the privilege to extend our hearty and sincere gratitude to the Professor **Dr.K.RAJA,M.E.,Ph.D.,** and Head of the Department, for his suggestions, support and encouragement towards the completion of the project with perfection.

We express our hearty and sincere thanks to our guide **Dr. ALAGUMANI S, Assistant Professor,** Computer Science and Engineering Department for her encouragement, consecutive criticism and constant guidance throughout this project work.

Our thanks to the teaching and non-teaching staff of the Computer Science and Engineering Department of SRM Institute of Science and Technology, Ramapuram Campus, for providing necessary resources for our project.

D.NIVAS GANDHI

S. RAJESH REDDY

G. AARTHI

ABSTRACT

It is becoming more common in the rail transit industry to use short-term forecasting of passenger flow at metro stations to support daily operations and management in rail transit networks. Since then, it has grown to be the most critical part of traffic construction management and the answer to urban traffic congestion. When combined with Kernel Extreme Learning Machines, W-KELM is an effective forecasting model (KELM). Passenger flow data is segmented into high- and low-frequency sequences using the WT and Mallet technique. Learning and forecasting signals of multiple frequencies is done using the KELM technique. Back-propagated error decay prevented it from learning and remembering extended sequences. For short-term forecasting, a new neural network (LSTM NN) is constructed employing Long Short-Term Memory Neural Network (LSTM). Ongoing real volume series and anticipated volume series in light of occasional elements are contributions to this model. The LSTM model is trained using a two-stage technique that can more correctly and rapidly reflect large fluctuations in anomalous flow.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	**Name**
LSTM	Long Short Term Memory
LSTMNN	Long Short Term Memory Neural Network
WT	Wavelet Transform
KELM	Kernel Extreme Learning Machine
SVM	Support Vector Machine

CHAPTER 1
INTRODUCTION

1.1 Overview

In developing cities, Metro electric rail transportation is a critical component of metropolitan infrastructure. It is the most often used mode of transportation in metropolitan areas. To reduce resource waste, it is critical to forecast passenger flow accurately. A growing role for real-time short-term traffic flow prediction may be played not only in supplying essential travel information for individual passengers and businesses, but it can also help reduce carbon dioxide emissions and enhance safety on the road. Additionally, commuters may utilize this information to plan their journeys and arrive at their destinations on time.

More resources are being allocated to the metro transport industry to improve passenger safety and comfort in different Indian cities. As a consequence of this transportation system's equitable utilization of resources, areas where resources are scarce are burdened with a disproportionate burden. Diverting resources and labour to areas where they are not needed is a common occurrence. The metro transport services train schedules are not based on the number of passengers.

The trains run with fewer passengers on each route, resulting in operating expenses that are either unprofitable or barely cover the operational costs. Other critical difficulties include the schedule of availability, which leads in alternative routes being more frequent during less busy hours, resulting in the same dilemma as described before. A variety of methodologies are used to predict passenger flow in a rail transport system, including neural network models, Support Vector Machine (SVM), Convolution Neural Network (CNN), Graphical CNN, Recurrent Neural Networks (RNN), and Hybrid models, Time series models, Frequency-based models, and Elasticity-based models.

1.2 Introduction to Machine Learning

A sub-field of computer science known as "machine learning" makes predictions about the behaviour of systems by analyzing data and using artificial intelligence. Machine Learning algorithms use statistical techniques to find patterns in data. These algorithms are used to predict the future. It is possible to build prediction models that take into account a wide range of data from various sources thanks to machine learning. There have been a number of research on the use of ML algorithms to predict the flow of passengers.

1.2.1 Deep Learning approach

In compared to machine learning (ML) and statistical approaches, deep learning (DL) methods have consistently shown forecasting accuracy of approximately 90% and greater. Based on neural networks, Deep Learning techniques are used in the field.

Artificial Neural Networks (ANN) and Neural Networks (NN) are two or more layers of linked nodes (neurons) meant to operate like the human brain. Many distinct neural networks have been built for a variety of reasons. Here are a few examples from traffic modelling and forecasting.

The purpose of Recurrent Neural Networks (RNNs), as opposed to CNNs, is to analyse time-series data or observations gathered over a certain period of time. Observing traffic patterns is an excellent illustration of this. Research has shown that RNN models may accurately forecast the progression of congestion. They have a short-term memory, which is why RNNs are referred to as "having a short-term memory" because of the vanishing gradient problem. "Forgetfulness" makes training models more time-consuming and complex.

Variations of the RNN that solve the issue of vanishing gradients include LSTM and GRU (Gated Recurrent Unit). Compared to other models, the GRU model proved to be more accurate in traffic flow forecasts and simpler to train in a comparison study. Studies have shown that different kinds of neural networks, such as graph, fuzzy and Bayesian neural networks, may be used to forecast traffic patterns. In addition to this, hybrid approaches that integrate two or more algorithms can also be used. As of right now, there isn't an one ideal approach that can be used in all situations to provide the most accurate predictions.

1.3 Problem Statement

- Passenger flow estimate is based on previous data in the proposed system. Due to increased urban traffic and unpredictable passenger flow, the rail transit road network has become increasingly problematic. In order to take into account this shift in historical passenger flow and provide more accurate forecasts, a thorough understanding of time series characteristics and shifting passenger flow patterns, such as daily law, weekly law, holiday legislation, and so on, was completed prior to the prediction. Analyzing the statistical data features of historical rail transit passenger flow serves as a foundation for developing a short-term forecast scheme, enhancing the prediction technique, and achieving more accurate results.
- LSTM NNs have three layers: an info layer, a repetitive secret layer, and a result layer. They are all types of recurrent neural networks. Traditional Recurrent Neural Networks use the term "memory block" to describe the structure introduced by LSTM models. Memory blocks include four major components: an input gate, a self-recursive memory cell, a forget gate, and an output gate. They are nonlinear summation units that control the interactions between the memory cell and the environment. The input gate may either allow or stop incoming signals from changing the state of a memory cell. The memory cell's state may be sent to the rest of the network or prevented by the output gate.
- To evaluate the proposed LSTM model, the raw dataset is divided into two parts: a training set and a test set. An adaptive moment estimation optimization (AMEO) approach and the Mean Squared Error Loss (MSEL) function are used to train the model provided here. Recurrent neural network architecture is used in all forecasting models. All of the forecasting models use the same procedure and settings for separating the data. To further understand how parameters affect predicting accuracy, several parameter values will be examined.

1.4 Objective

Network tomography's estimate error may be significantly reduced using a Long Short Term Memory Neural Network (LSTM NN) compared to more traditional systems that employ a uniform or random strategy. This methodology also makes better use of unstructured data and reduces deployment costs.

CHAPTER 2
LITERATURE SURVEY

2.1 Introduction

Literature reviews are summaries of key books and other information on a certain subject. The review may include academic journals, novels, government publications, websites, and other sources. There includes a thorough description, summary, and evaluation of every resource used in the review. A graduate thesis or dissertation is the most frequent format for including it.

This project's goal is to do research on various machine learning algorithms in order to discover one that produces highly accurate results. The study provides algorithm results as well as accuracy measurements.

2.2 Existing System

Smart city planning and management is made possible by efficient traffic monitoring in metropolitan areas. It has become a popular method for managing urban traffic since it requires a very minimal number of traffic detectors, such as cameras. While past organization tomography-based traffic observing examinations have focused on creating assessors utilizing given start to finish travel time estimations, the plan of information assortment for effectively gathering and handling crude it is frequently forgotten to screen recordings to such estimations. Notwithstanding, this approach enjoys a few benefits. We might fill this hole by concentrating anxious insight for productive metropolitan traffic checking and resolving the accompanying two issues: limited broadcast communications assets and data transmission imperatives. Fisher Information Matrix (FIM) is utilized to decide the number of observing recordings each traffic locator requirements to produce to handle the start to finish travel time estimations, and when the concentrated handling of checking recordings alone is inadequate, calculation capacities from these edge gadgets are utilized, and utilize a multi-specialist support learning approach when the unified handling of observing recordings alone is lacking. Simulations show that the suggested method is successful in reducing network tomography estimate errors compared to existing techniques using either uniform or random strategy. There are a variety of ways in which Network Tomography may be used to correctly measure urban congestion and minimise the number of traffic monitors needed. The problem is that present studies primarily concentrate on how to install traffic monitors and estimate urban traffic, but do not include the gap between the transmission and handling of street traffic checking films and the gathering of start to finish trip time at the TCC. It presented an edge-powered urban traffic monitoring system to address the restricted telecommunications resources in this scenario. For each monitoring node, this provided a FIM-based quota system for the monitoring videos. Using the RRO technique, the FIM observation matrix may be made more efficient by removing any unnecessary rows. Even if enough monitors are available, the most useful quota system is one that allows for direct monitoring of the whole road network. It was then suggested that the maximum projected end-to-end transport time be calculated using the MAC method. The estimate error for network tomography was suggested based on extensive simulation findings.

2.3 Summary of Literature Survey

Now days commuters flow prediction is important to show solution for traffic flow congestion. Different methods was proposed by different authors. Demianiuk et al. proposed robust traffic monitoring, with the goal of precisely determining a stream measure for each stream regardless of organizational commotion. This model adhere to the open-circle worldview, which adds no control parcels, transfers stream status in-band by connecting a small number of control pieces (up to two or four) to bundles of the checked streams, and maintains idleness low.

Celesti et al.[1] proposed a potential alternative solution for fixing such a problem using portable traffic sensors that may be easily installed in private, public, and volunteer cars. In this case, timely and consistent management of massive traffic information is critical to avoiding disasters. Celesti focuses on an OpenGTS and MongoDB-based IOT Cloud architecture for traffic evaluation and readiness warning. IoT Cloud architecture is particularly useful for ambulance drivers, apart from the individual drivers.

Wang et al.[2] An online traffic data and discrete-time Markov chain approach was presented to predict gridlock and detect roadblocks. In the course of conducting a thorough analysis, the researchers discovered that the northern section of National Freeway 3 is likely to stay unclogged throughout rush hour.

Tangari et al.[3] proposed MONA, a versatile checking system that is resistant to impediment while extending the precision of observation reports over a client with a certain edge. Under adverse situations, MONA gradually decreases and reconfigures estimation work sets in order to recoup probable exactness faults. MONA offers a sophisticated attempt to improve mechanism that generates precision gauges based on recently observed traffic data to assess the checking exactness at run time.

Ali et al.[4] the use of Webster's dictionary and a modified version of the Webster's equation as the basis for a model for the construction of a flexible traffic light management system With these equations, it was possible to determine the most efficient process time depending on the current traffic situation. To compensate for the variation, the traffic status alternates between two progressive cycles, which are monitored and managed by the fluffy reasoning framework. The ideal interaction length acquired was utilized to decide adaptively the fruitful stage green times, for instance, is utilized to decide adaptively the greatest permissible extension cutoff of the green stage in the accompanying cycle. The proposed flexible control strategies are evaluated using the SUMO traffic test system.

Ruiz et al.[5] proposed a CURSA-SQ model, a method for breaking down network behaviour when explicit traffic generated by gatherings of administrative customers is infused. CURSA-SQ includes explicit assistance and buyer behaviour, as well as a consistent G/G/1/k line model based on calculated capacity and input traffic stream display with second and sub-second precision. The technique takes into consideration precisely focusing on the traffic flows at the information and outcomes of challenging problems using product line structures, as well as various measurements like postponements, while demonstrating observable adaptability.

Guidonietal.[6] proposed Re-RouTE model, an automotive Traffic Management System based on Traffic Engineering hypotheses, to further extend car traffic streams in dense urban regions. For blocked paths, the Re-RouTE administration uses a stream thickness clearly apparent traffic planning idea based on vehicle thickness on highways. Using a weighted diagram, it is possible to locate roadways with low traffic and just a modest increase in the amount of travelable space.

Guo et al.[7] In order to efficiently handle all street network traffic scenarios and create a succession of practical and secure computing conventions to speed up information handling, a protection protecting compressive detecting (PPCS) system was suggested (see figure). Existing model used cloud in particular to isolate the spatial-transient relationship between the various street segments and calculate the gridlock rates of a few street sections.

Zhang et al.[8] the use of test cars and GPS vehicle limitation is unnecessary when dealing with traffic observation, according to a new approach. Alternatively, it uses only a limited number of cameras stationed at street crossings to calculate vehicle travel duration's from start to finish. Zhang describe the problem within a hypothetical system of organization tomography in order to predict the voyaging seasons of every individual roadway fragments in the street organization.

2.4 Issues in Existing System

The following are the issues that were identified after reviewing the existing works.

- This system is Opportunistic and uncontrollable.
- Cannot exploit the new feature space.
- Cannot restrain behavior pattern in complex systems .
- Complexity of its Real time implementation.
- More time consuming while the number of network configurations is very large.
- Computationally intensive and require relatively large memory space .
- Poor performance and high variance on the validation data.

CHAPTER 3
SYSTEM DESIGN

3.1 Introduction

A general summary of how the system is intended to be forecasting is provided in this chapter. This chapter contains a brief overview of the system architecture as well as the procedures involved in the development of this system. Each module represents distinct stages of work completed in various portions of the system.

Designing a system's modules, architecture, components, and interfaces is known as "System Design" when it is done in accordance with the given requirements. Creating and constructing systems that suit a firm or organization's specific goals and expectations is referred to as system identification, creation, and design.

A well-functioning and coherent system need a systemic perspective. A base up or hierarchical methodology is expected to represent all framework related components. To characterize data and information in a steady framework structure, creators utilize displaying dialects. Both graphical and textual modelling languages may be used to describe the designs.

Among the graphical modelling languages are the following:

1. Unified Modelling Language (UML): To use graphical notation to explain software both structurally and behaviorally.
2. Flowchart: An algorithm's schematic or sequential layout.
3. Business Process Modelling Notation (BPMN): Process Modelling language is used in this language.
4. Systems Modelling Language (SysML): System engineering terminology.

3.2 System Architecture

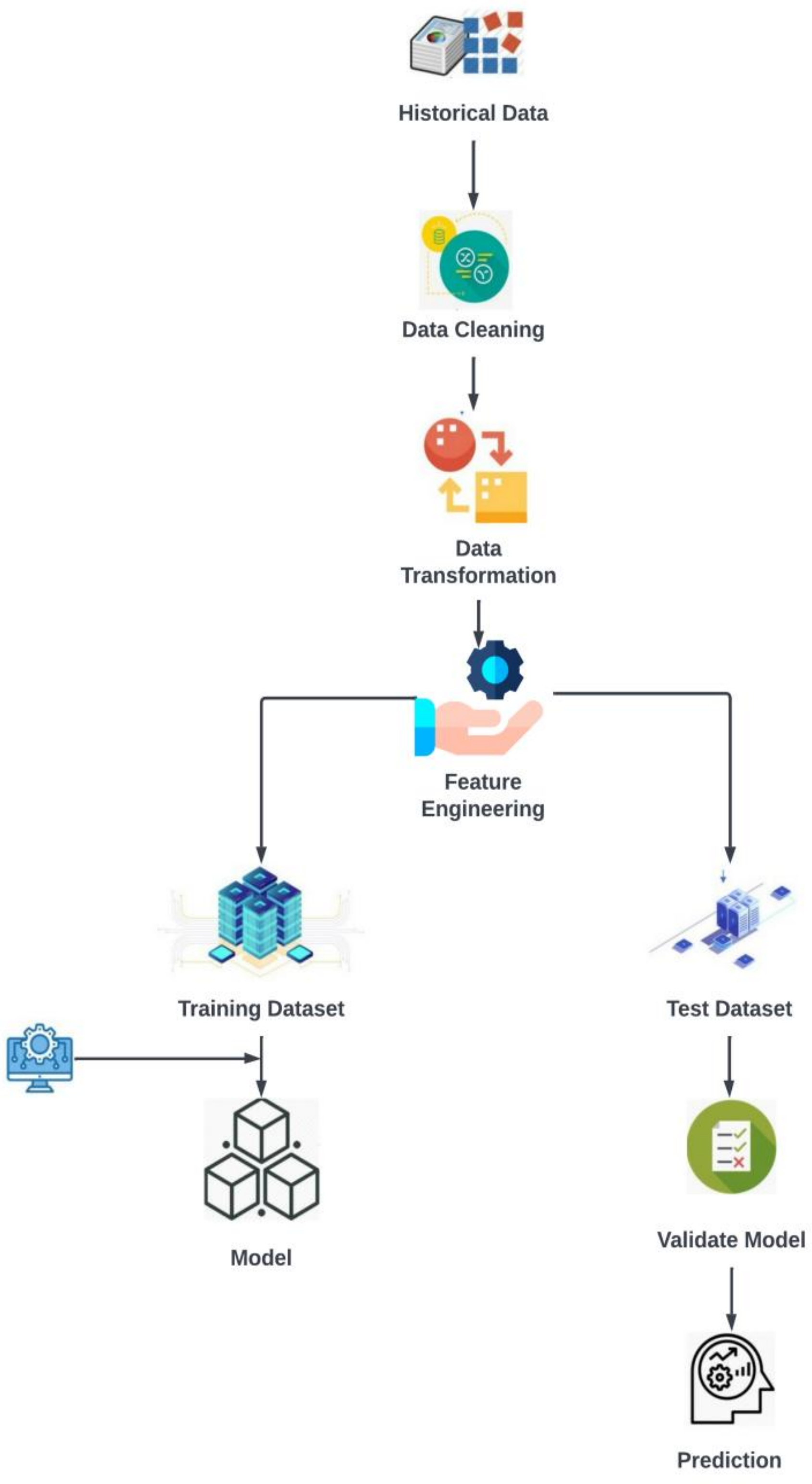


Fig 3.1: System Architecture

3.2.1 Histroical Dataset

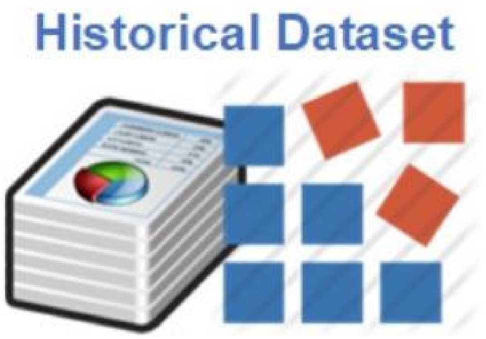


Fig 3.2 Historical Data set

Predictive analysis relies heavily on past data. Collection of data regarding previous events and situations relevant to one's topic is known as historical data.

3.2.2 Data Transformation

Fig:3.3 Data Transformation

To lay it out plainly, information change is the method involved with changing information between one arrangement and another, frequently starting with one source framework then onto the next. Most information reconciliation and information the executives exercises, for example, information fighting and information warehousing, incorporate information change.

3.2.3 Feature Engineering

The feature engineering pipeline is a set of preprocessing stages that convert crude information into highlights that might be used in AI strategies, like prescient models.

In the feature engineering process, the most helpful predictor variables are produced and chosen to be used in a predictive model's model building. Since 2016, some machine learning software has included automated feature engineering. There are four primary phases to feature engineering: Feature Creation, Transformations, Feature Extraction, and Feature Selection.

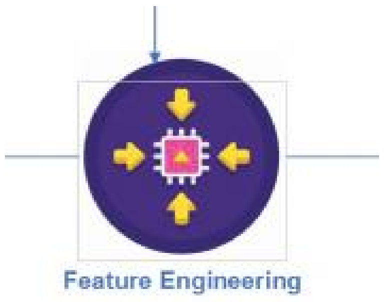


Fig 3.4 Feature engineering

3.2.4 Training Dataset

Fig 3.5 Training Data set

An enormous amount of data is gathered for the purpose of training a machine learning model. Prediction models that employ machine learning algorithms are taught how to extract attributes related to certain business objectives using training data. The training data is labelled in supervised ML models. Unsupervised ML models are trained on unlabeled data. The notion of utilising training data in machine learning algorithms is a basic concept, but it is also highly essential to the way that these technologies function. This first batch of data is used to teach a software how to utilise neural networks and other advanced techniques to learn and create more complex results. Additional data sets termed validation and testing sets may be used in conjunction with the first set of data. The terms "training data" and "training dataset" may all refer to the same thing: a collection of practise data.

3.2.5 Test Dataset

Fig 3.6 Test data set

Using a train-test dataset, which divides the dataset into training and testing halves, the learning model may be taught with the training data and its performance evaluated with the testing data. A model is trained and tested using the complete dataset in most advanced approaches.

3.3 System Requirements

Hardware Requirements

1. Laptop/Desktop with minimum specs
2. Dedicated GPU

Software Requirements

1.Anaconda

2. Jupyter Notebook

3. Python

a) numpy

b) Pandas

c) Math plotlib

3.3.1 Introduction to Python

Python is a progressively semantic, significant level, object-situated it is interpretable to programme language that. As a prearranging or stick language that interfaces existing parts thanks to it's an extraordinary decision for Faster Development. The clarity and simplicity of Python's syntax help to keep software upkeep costs low. As a result of its support for modules and packages, Python facilitates both programme modularity and reuse. For all major working frameworks, Python's translator and standard library are unreservedly accessible in source or twofold structure.

A defect or improper input in a Python programme will never cause a segmentation fault. An exception is thrown by the interpreter when an error is discovered. Stack traces are generated by the interpreter if the programme fails to handle a special case. It is feasible to look at neighborhood and worldwide factors and articulations, assess inconsistent articulations, produce breakpoints, etc utilizing a source level debugger . Python's introspective nature is shown through the debugger, which is written in the language. Another approach to debugging software is by adding a couple of print proclamations to the source code.

3.3.2 Features of Anaconda

An easy-to-install Python and R distribution, Anaconda is free and open source. Analysis and machine learning may be done using Anaconda's scientific computing environment. Anaconda 5.3, which was released in October 2018, is the latest current distribution. Over 1000 open-source programmes are included in the condo package, as well as an environment manager. As a default part of the Anaconda distribution, the Anaconda Navigator GUI may be found on your computer's desktop. It enables us to run Anaconda-provided apps. Without using command-line commands can quickly distribute condo packages, environments, and channels.

3.3.3 Jupyter Notebook (Anaconda)

The Jupyter Notebook App is a server-client program that empowers online altering and execution of journal papers. This article explains how to use the Jupyter Notebook App on a local desktop without a network connection, or step by step instructions to utilize a far off server and interface with it by means of the web.

The Jupyter Notebook App has a "Dashboard" (Notebook Dashboard), a "control board" that shows neighborhood records and empowers you to open or excuse journal pages.

3.3.4 TensorFlow

Deep Learning is an area of AI that depends on the design and capacity of the human cerebrum and is an assortment of algorithms. Machining learning includes a subclass called deep learning. The term "deep learning" refers to the kind of machine learning that we use in our work. Deep learning, for example, employs neural networks, which mimic the human brain in a way. In contrast to classical machine learning, which often employs structured data, deep learning also requires the analysis of huge volumes of unstructured data. Images, video, audio, text, and other types of unstructured data may all be used to feed the system. TensorFlow is Google's subsequent AI structure for planning, building, and preparing profound learning models. This library might be utilized to do mathematical computations, which may not appear very noteworthy except that data flow graphs are employed to carry out these computations. Edges represent the data (typically multidimensional arrays or tensors) that connect the nodes of the graphs with the nodes of the graphs' edges. Name TensorFlow comes from the multidimensional data arrays or tensors that neural networks process. It's a stream of tensors, plain and simple. Deep learning models may be built, evaluated, and used in a few lines of code using tf.keras. Classification and regression predictive modelling may now be done using deep learning problems like this. TensorFlow's scalability is another essential feature. For training purposes, you may use CPU, GPU, or a cluster of these devices to execute your code. Training the model is where most of the computation occurs. In addition, the training procedure is repeated numerous times to address any difficulties that may develop. As a result of the increased power usage, a distributed computing system is required. TensorFlow makes it simple to process massive volumes of data by distributing the code. Graphics processing units, or GPUs, have grown in popularity in recent years. As a market leader, Nvidia has a lot of influence in this area. Subsequently, it assumes a urgent part in the advancement of deep learning algorithms. In addition to the C++ and Python APIs, TensorFlow includes built-in support for TensorFlow.

Higher-dimensional arrays make up a tensor, a mathematical entity. The neural network is supplied with a variety of data arrays, each with a varied size and rank. The tensors are represented by these objects.

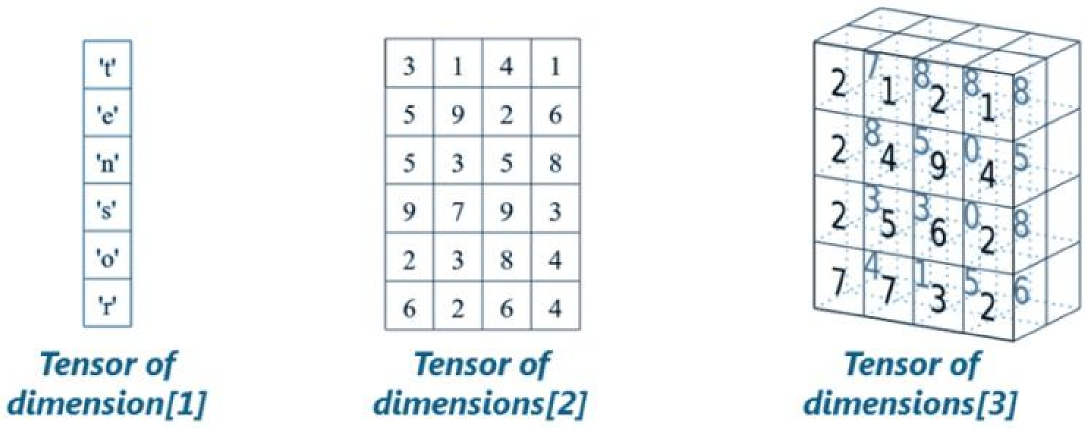


Fig 3.7Tensor Flow

3.4 Summary

This section has discussed about the overall structure and the elements of the architectural structure in detail with their roles and responsibilities. The proposed structure is more effective and efficient in the prediction

CHAPTER 4
MODULE DESCRIPTION

4.1 Introduction

A module is a combination of source files and build parameters that allow developers to break a project into separate functional components. The project may have one or more modules, and one module may rely on another module. Each module may be produced, tested, and debugged individually

4.2 Feature Extraction

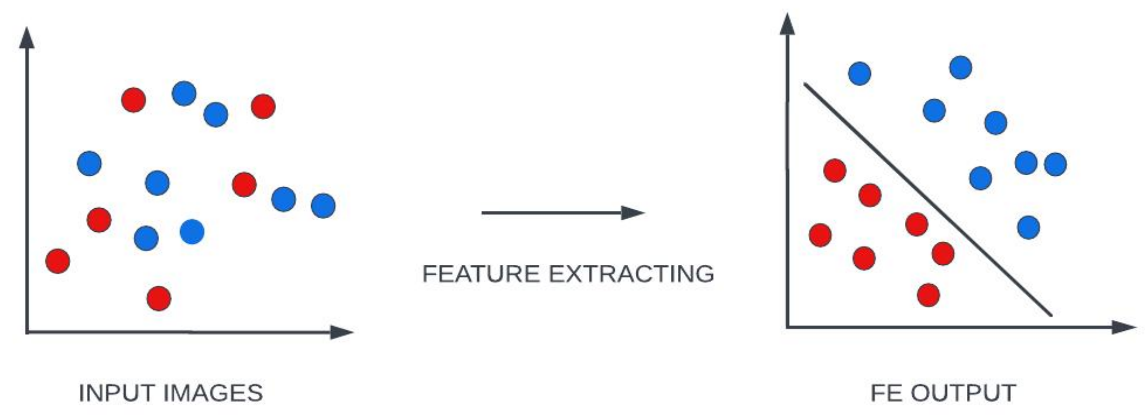


Fig 4.1 Feature Extraction

The process of picking a subset of the original feature set on the basis of relevance of characteristics is known as feature selection. Wrappers, filters, and embedding techniques are the three types of feature selection methods. The process of extracting a new set of features from the collection of features created during the feature selection step is known as feature extraction. If a characteristic is regularly found in one category, it will get a higher score from the Ambiguity Measure (AM). Each feature has an AM score assigned to it. If the feature is unambiguous, this approach assigns a score near to 1; otherwise, it gives a score close to 0. Features with AM scores less than or equal to the threshold are discarded, while those with AM scores greater than the threshold are used in the learning phase. The difference in entropy between whether or not a feature occurs in text is used to calculate the feature's information gain. What matters most is how much information is gained, and how much that knowledge contributes to the text. High-information-gain features will be prioritize for inclusion.

4.3 Exploratory Data Analysis

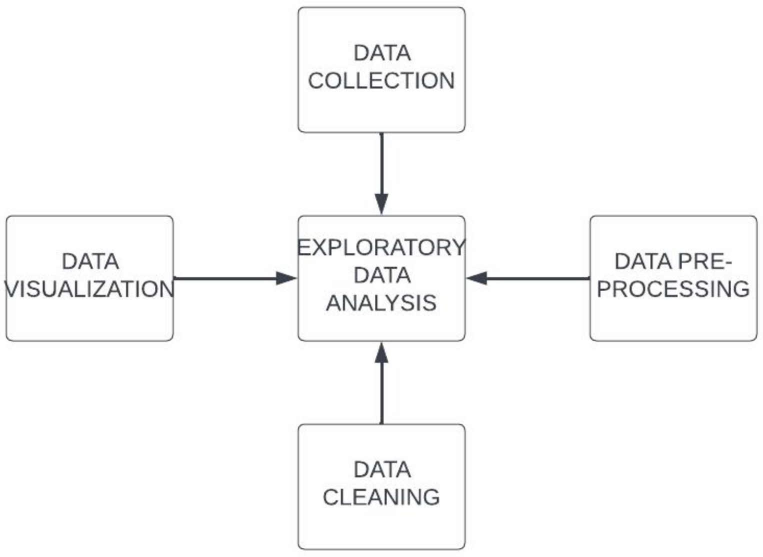


Fig 4.2 Exploratory Data Analysis

Based on the distribution of data in the dataset, exploratory data analysis is a technique of data analysis. In EDA, visual approaches are used to uncover the structure in the dataset. Visual data analysis techniques have been around for millennia, and their widespread usage is due in large part to the fact that human eyes and brains have a high anatomical capacity to detect. A number of human models may be used in visual analysis, which uses a unique manner of displaying data. When analysing data, analysts usually do exploratory data analysis first, and then pick either a structure-quantity or stochastic-quantit style of data analysis. It's also possible to discover deviations from the norm via exploratory data analysis. The most important aspect of exploratory data analysis is not only the ability to adapt to the data structure, but also the ability to adapt to the disclosed mode of the subsequent analysis step.

Evaluation of observable modes or effects-evoked by Exploratory Data Analysis is the goal of verification. Other closed-related information is combined with the newly acquired data, which is then analysed to ensure the accuracy of the final findings. An important part of EDA is visualising information and drawing conclusions from it using graphs and charts. These low-dimensional representations are easy to understand, but they often need a deep understanding of statistical methods and mathematical concepts in order to use. If you're looking for a quick and straightforward way to investigate several features of a dataset, visualizations and graphs may be the best option. The ultimate objective is to provide short summaries of the facts that are relevant to your query (s). As an intermediate step in data science, it's nonetheless a crucial one.

4.4 Exponential Smoothing

Outstanding smoothing of time series information appropriates dramatically diminishing loads to the latest to the most seasoned perceptions. That is to say, the less importance (weight) is given to data that is more dated. Data that is less than a year old is deemed less helpful and is given less weight. Loads for not entirely set in stone by smoothing boundaries (smoothing constants), which are frequently addressed as. Univariate time series forecasting may be done using exponential smoothing. As a result of time series approaches such as the Box-Jenkins ARIMA family, models are created in which the figure is a weighted direct amount of ongoing past perceptions or delays Exponentially declining weights for prior data distinguish exponential smoothing from other forecasting systems in that their predictions are weighted sums of previous observations. Because exponential smoothing is so inaccurate when utilised for longer-term predictions, it's often only used for short-term projections. An outstanding moving normal with diminishing loads is utilized for straightforward (single) dramatic smoothing. While managing information that demonstrates a pattern, Holts pattern amended twofold outstanding smoothing is regularly more exact than the single procedure. Parabolic trends or data showing trends and seasonality are better served by triple exponential smoothing (also known as the Multiplicative Holt-Winters).

4.5 Prediction

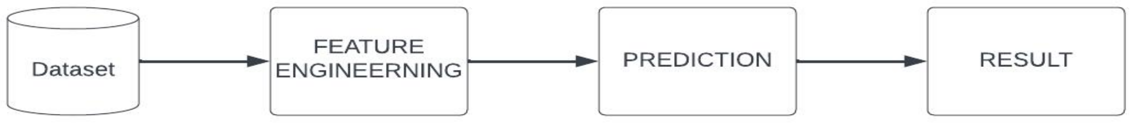


Fig:4.3 Prediction

To put it simply, ARIMA, short for Auto Regressive Integrated Moving Average, is essentially a family of models that may be used to predict the future of a particular time series by taking into account its own delays and the delayed prediction errors. Time series data may be studied and forecasted using an ARIMA model. For creating accurate time series predictions, it specifically accommodates to a wide range of standard structures in time series data. AutoRegressive Integrated Moving Average (ARIMA) is an abbreviation for the term. Because it incorporates integration, it's a more complex version of the more straightforward AutoRegressive Moving Average. This abbreviation is succinct, highlighting the model's most important features. One that leverages the link between one observed event and a number of delayed events. I'm a person who likes to be integrated. Deciding time series steadiness by eliminating crude information (for example by deducting a perception from the previous time step). Mama represents Moving Average, and it's a famous pointer. Moving normal model lingering mistakes applied to slacked information are utilized in this model. ARIMA models might be utilized to address any non-occasional time series that has designs and is certifiably not an irregular repetitive sound. It is possible to describe an ARIMA model using three terms: (p), (d), and (q). In mathematics, the word q represents the rank of a mathematical algebraic expression. The number of differencing operations necessary to keep the time series steady is denoted by the word d. If a time series exhibits seasonal trends, the word "SARIMA," short for "Seasonal ARIMA," must be included. After we complete ARIMA, we'll have more to say about it.

CHAPTER 5
SYSTEM IMPLEMENTATION

5.1 Introduction

Python is the programming language of choice for the proposed system. Python may be utilized for many different types of projects because of its versatility and ease of usage. Web development, AI, machine learning, operating systems, mobile applications, and video game production are all examples of Python's wide range of applications. Jupyter note books by Anaconda is used for implementation of this model.

5.2 Overview

5.2.1 Python

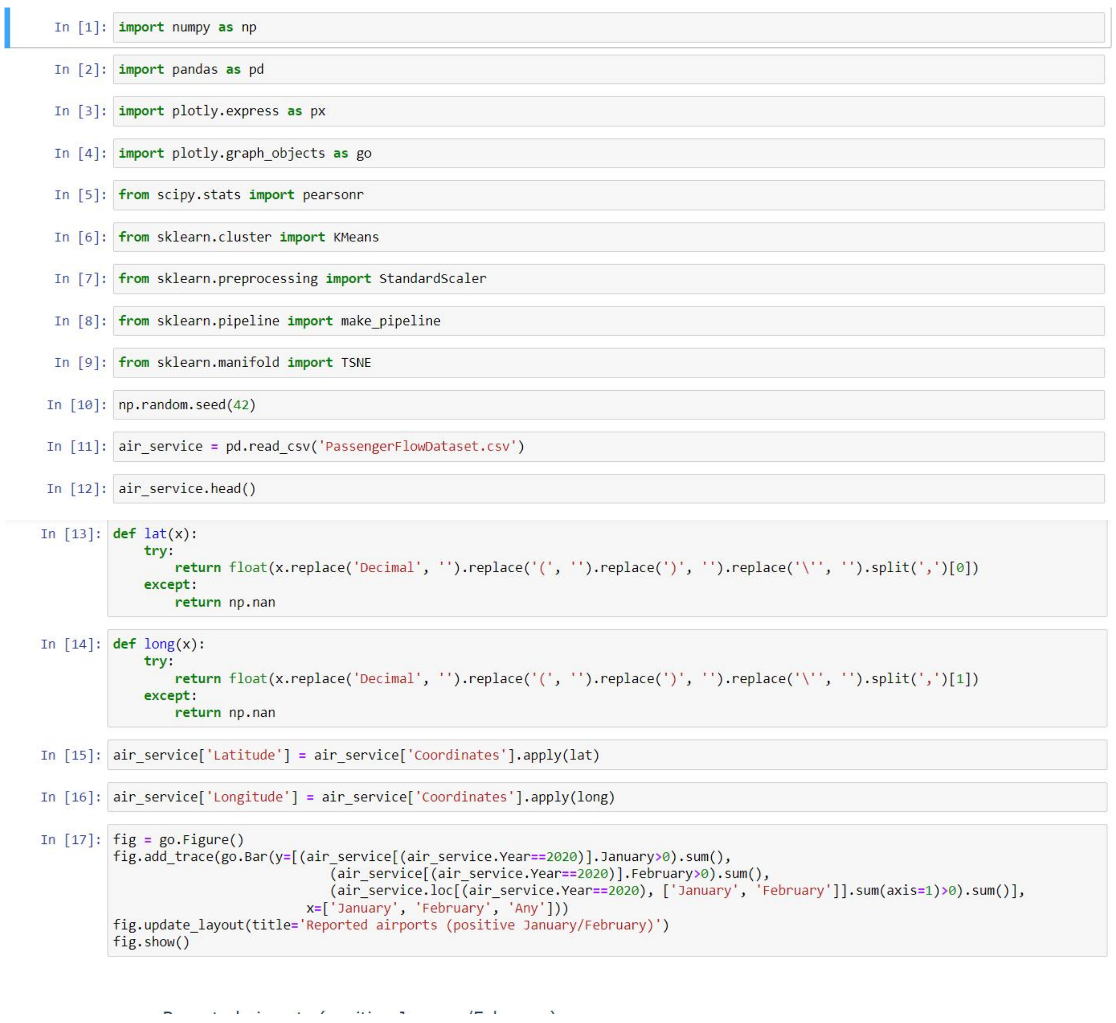
Python is a progressively semantic, undeniable level, object-arranged it is interpretable to programme language that. As a pre arranging or stick language that interfaces existing parts thanks to its undeniable level underlying information structures, dynamic composing, and dynamic restricting, it's an extraordinary decision for Faster Development. The clarity and simplicity of Python's syntax help to keep software upkeep costs low. As a result of its support for modules and packages, Python facilitates both programme modularity and reuse.

A defect or improper input in a Python programme will never cause a segmentation fault. An exception is thrown by the interpreter when an error is discovered. Stack traces are generated by the interpreter if the programme fails to handle an exception. It is feasible to look at nearby and worldwide factors and articulations, assess erratic articulations, produce breakpoints, etc utilizing a source level debugger. Python's introspective nature is shown through the debugger, which is written in the language. Another approach to debugging software is by adding . a couple of print articulations to the source code. This short alter test-troubleshoot cycle simplifies this method so successful.

5.2.2 Data Set

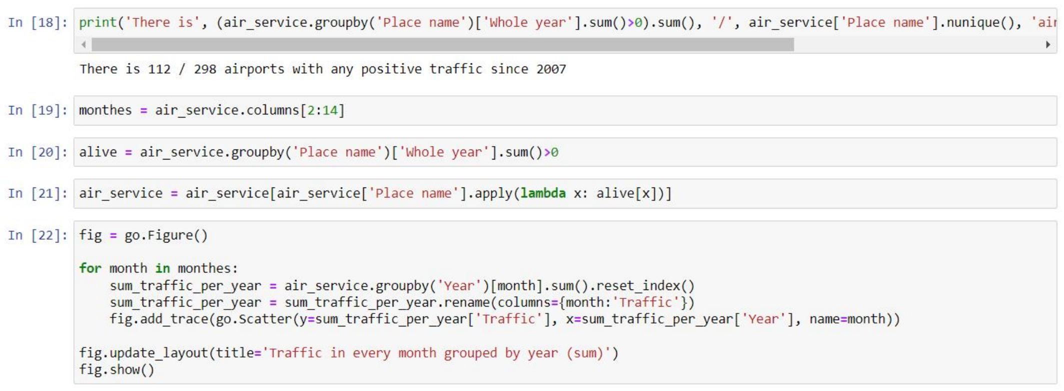
The dataset contains historical passenger statistics for the previous thirteen years. This data-set was obtained through Kaggle. The data-set includes four thousand occurrences and sixteen characteristics, including the location's name, year, months, and coordinates (Latitude and Longitude).

5.2.3 SAMPLE CODING

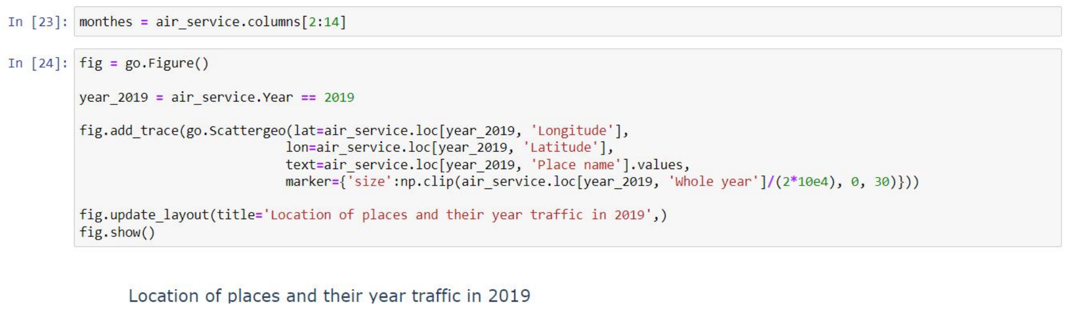


```
In [1]: import numpy as np

In [2]: import pandas as pd

In [3]: import plotly.express as px

In [4]: import plotly.graph_objects as go

In [5]: from scipy.stats import pearsonr

In [6]: from sklearn.cluster import KMeans

In [7]: from sklearn.preprocessing import StandardScaler

In [8]: from sklearn.pipeline import make_pipeline

In [9]: from sklearn.manifold import TSNE

In [10]: np.random.seed(42)

In [11]: air_service = pd.read_csv('PassengerFlowDataset.csv')

In [12]: air_service.head()

In [13]: def lat(x):
             try:
                 return float(x.replace('Decimal', '').replace('(', '').replace(')', '').replace('\'', '').split(',')[0])
             except:
                 return np.nan

In [14]: def long(x):
             try:
                 return float(x.replace('Decimal', '').replace('(', '').replace(')', '').replace('\'', '').split(',')[1])
             except:
                 return np.nan

In [15]: air_service['Latitude'] = air_service['Coordinates'].apply(lat)

In [16]: air_service['Longitude'] = air_service['Coordinates'].apply(long)

In [17]: fig = go.Figure()
         fig.add_trace(go.Bar(y=[(air_service[(air_service.Year==2020)].January>0).sum(),
                                 (air_service[(air_service.Year==2020)].February>0).sum(),
                                 (air_service.loc[(air_service.Year==2020), ['January', 'February']].sum(axis=1)>0).sum()],
                              x=['January', 'February', 'Any']))
         fig.update_layout(title='Reported airports (positive January/February)')
         fig.show()
```

Fig 5.1 Coding screen shots

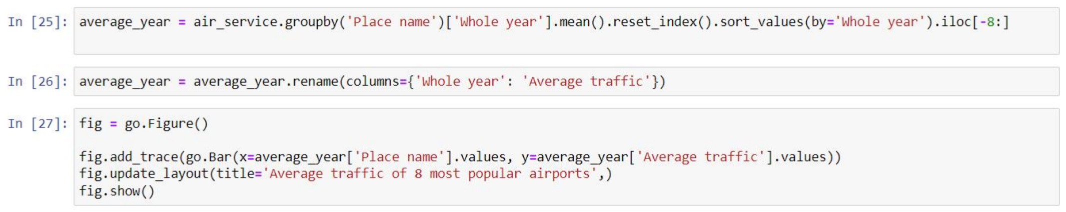


```
In [18]: print('There is', (air_service.groupby('Place name')['Whole year'].sum()>0).sum(), '/', air_service['Place name'].nunique(), 'ai

         There is 112 / 298 airports with any positive traffic since 2007

In [19]: monthes = air_service.columns[2:14]

In [20]: alive = air_service.groupby('Place name')['Whole year'].sum()>0

In [21]: air_service = air_service[air_service['Place name'].apply(lambda x: alive[x])]

In [22]: fig = go.Figure()

         for month in monthes:
             sum_traffic_per_year = air_service.groupby('Year')[month].sum().reset_index()
             sum_traffic_per_year = sum_traffic_per_year.rename(columns={month:'Traffic'})
             fig.add_trace(go.Scatter(y=sum_traffic_per_year['Traffic'], x=sum_traffic_per_year['Year'], name=month))

         fig.update_layout(title='Traffic in every month grouped by year (sum)')
         fig.show()
```

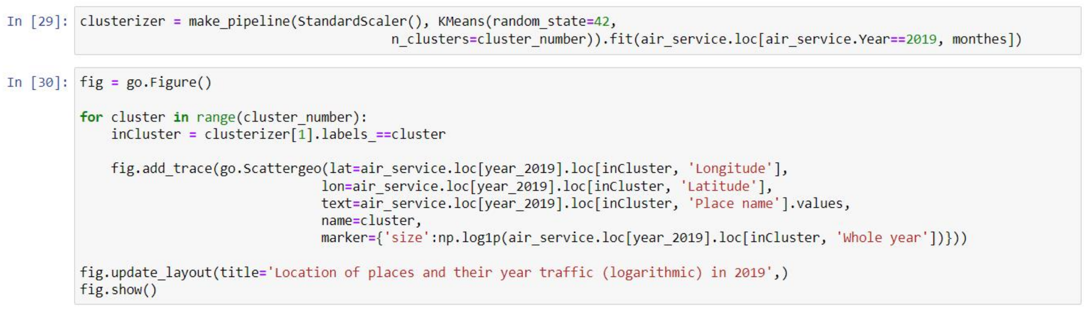


```
In [23]: monthes = air_service.columns[2:14]

In [24]: fig = go.Figure()

         year_2019 = air_service.Year == 2019

         fig.add_trace(go.Scattergeo(lat=air_service.loc[year_2019, 'Longitude'],
                                     lon=air_service.loc[year_2019, 'Latitude'],
                                     text=air_service.loc[year_2019, 'Place name'].values,
                                     marker={'size':np.clip(air_service.loc[year_2019, 'Whole year']/(2*10e4), 0, 30)}))

         fig.update_layout(title='Location of places and their year traffic in 2019',)
         fig.show()
```

Location of places and their year traffic in 2019

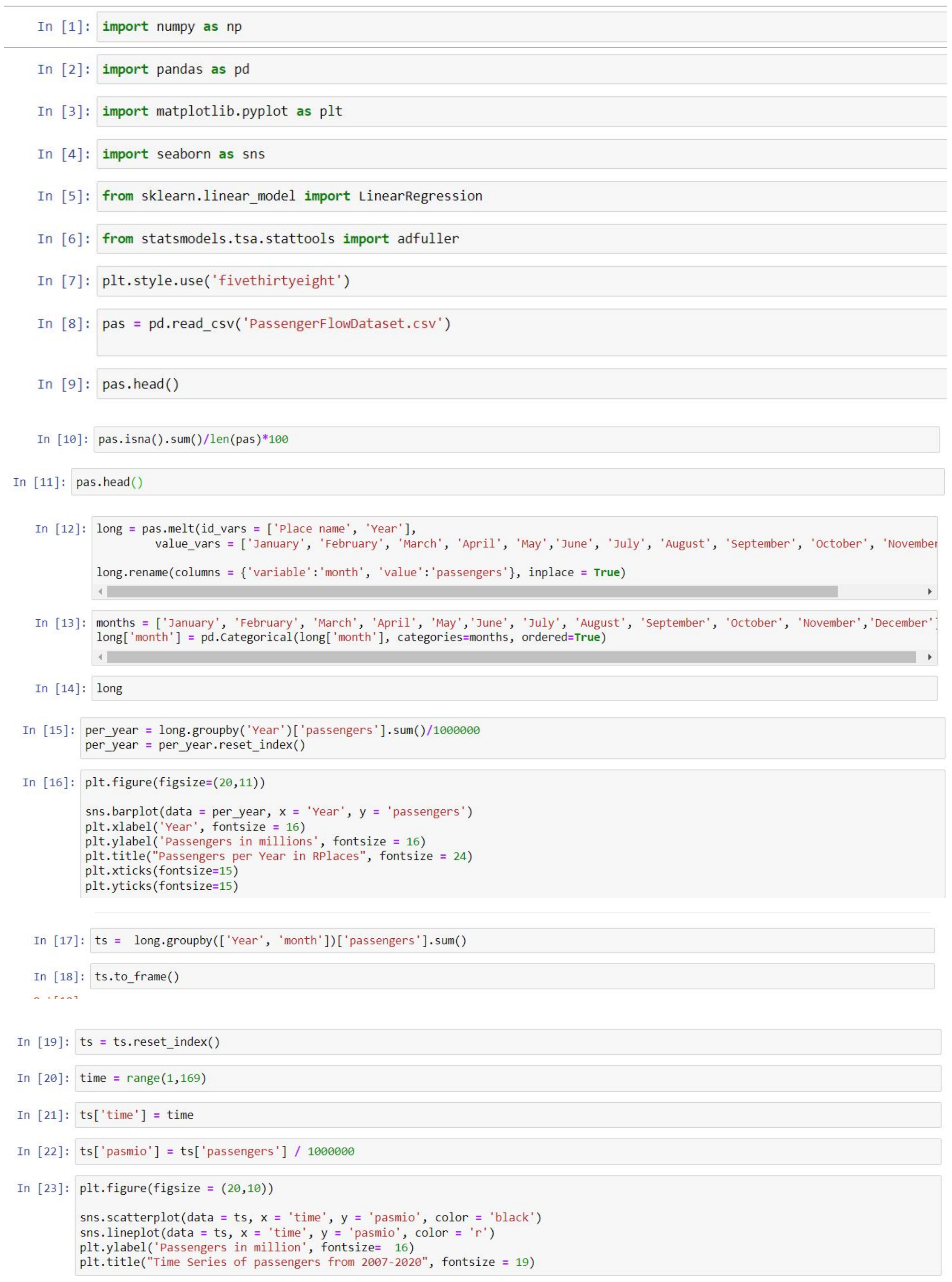


```
In [25]: average_year = air_service.groupby('Place name')['Whole year'].mean().reset_index().sort_values(by='Whole year').iloc[-8:]

In [26]: average_year = average_year.rename(columns={'Whole year': 'Average traffic'})

In [27]: fig = go.Figure()

         fig.add_trace(go.Bar(x=average_year['Place name'].values, y=average_year['Average traffic'].values))
         fig.update_layout(title='Average traffic of 8 most popular airports',)
         fig.show()
```



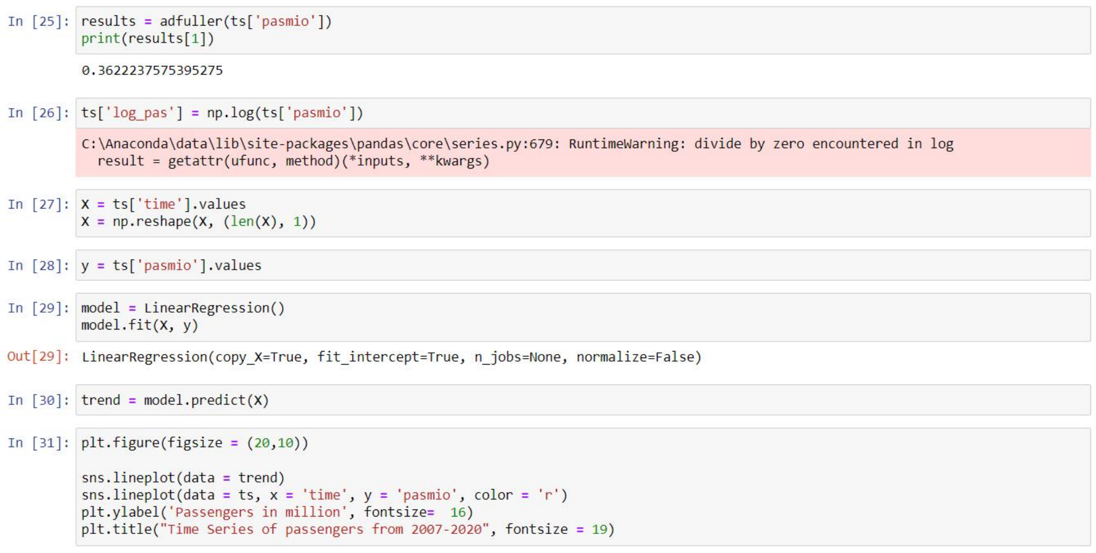
```
In [29]: clusterizer = make_pipeline(StandardScaler(), KMeans(random_state=42,
                                         n_clusters=cluster_number)).fit(air_service.loc[air_service.Year==2019, monthes])

In [30]: fig = go.Figure()

         for cluster in range(cluster_number):
             inCluster = clusterizer[1].labels_==cluster

             fig.add_trace(go.Scattergeo(lat=air_service.loc[year_2019].loc[inCluster, 'Longitude'],
                                         lon=air_service.loc[year_2019].loc[inCluster, 'Latitude'],
                                         text=air_service.loc[year_2019].loc[inCluster, 'Place name'].values,
                                         name=cluster,
                                         marker={'size':np.log1p(air_service.loc[year_2019].loc[inCluster, 'Whole year'])}))

         fig.update_layout(title='Location of places and their year traffic (logarithmic) in 2019',)
         fig.show()
```

Fig 5.2 Coding screen shots

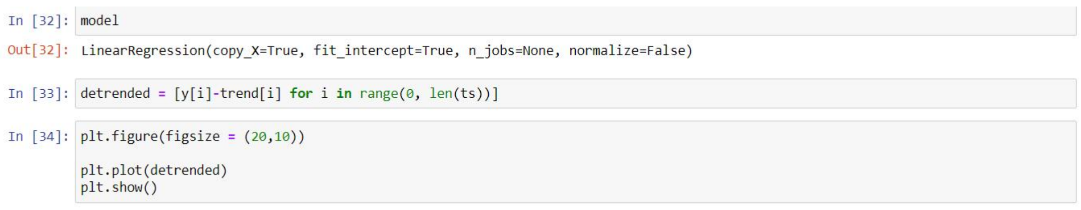


```
In [1]: import numpy as np

In [2]: import pandas as pd

In [3]: import matplotlib.pyplot as plt

In [4]: import seaborn as sns

In [5]: from sklearn.linear_model import LinearRegression

In [6]: from statsmodels.tsa.stattools import adfuller

In [7]: plt.style.use('fivethirtyeight')

In [8]: pas = pd.read_csv('PassengerFlowDataset.csv')

In [9]: pas.head()

In [10]: pas.isna().sum()/len(pas)*100

In [11]: pas.head()

In [12]: long = pas.melt(id_vars = ['Place name', 'Year'],
                  value_vars = ['January', 'February', 'March', 'April', 'May','June', 'July', 'August', 'September', 'October', 'Novembe

         long.rename(columns = {'variable':'month', 'value':'passengers'}, inplace = True)

In [13]: months = ['January', 'February', 'March', 'April', 'May','June', 'July', 'August', 'September', 'October', 'November','December']
         long['month'] = pd.Categorical(long['month'], categories=months, ordered=True)

In [14]: long

In [15]: per_year = long.groupby('Year')['passengers'].sum()/1000000
         per_year = per_year.reset_index()

In [16]: plt.figure(figsize=(20,11))

         sns.barplot(data = per_year, x = 'Year', y = 'passengers')
         plt.xlabel('Year', fontsize = 16)
         plt.ylabel('Passengers in millions', fontsize = 16)
         plt.title("Passengers per Year in RPlaces", fontsize = 24)
         plt.xticks(fontsize=15)
         plt.yticks(fontsize=15)

In [17]: ts =  long.groupby(['Year', 'month'])['passengers'].sum()

In [18]: ts.to_frame()

In [19]: ts = ts.reset_index()

In [20]: time = range(1,169)

In [21]: ts['time'] = time

In [22]: ts['pasmio'] = ts['passengers'] / 1000000

In [23]: plt.figure(figsize = (20,10))

         sns.scatterplot(data = ts, x = 'time', y = 'pasmio', color = 'black')
         sns.lineplot(data = ts, x = 'time', y = 'pasmio', color = 'r')
         plt.ylabel('Passengers in million', fontsize=  16)
         plt.title("Time Series of passengers from 2007-2020", fontsize = 19)
```

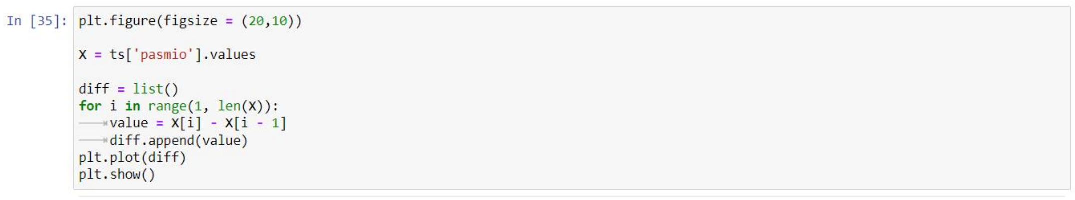
Fig 5.3 Coding screen shots



```
In [25]: results = adfuller(ts['pasmio'])
         print(results[1])

         0.3622237575395275

In [26]: ts['log_pas'] = np.log(ts['pasmio'])

         C:\Anaconda\data\lib\site-packages\pandas\core\series.py:679: RuntimeWarning: divide by zero encountered in log
           result = getattr(ufunc, method)(*inputs, **kwargs)

In [27]: X = ts['time'].values
         X = np.reshape(X, (len(X), 1))

In [28]: y = ts['pasmio'].values

In [29]: model = LinearRegression()
         model.fit(X, y)

Out[29]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)

In [30]: trend = model.predict(X)

In [31]: plt.figure(figsize = (20,10))

         sns.lineplot(data = trend)
         sns.lineplot(data = ts, x = 'time', y = 'pasmio', color = 'r')
         plt.ylabel('Passengers in million', fontsize=  16)
         plt.title("Time Series of passengers from 2007-2020", fontsize = 19)
```



```
In [32]: model

Out[32]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)

In [33]: detrended = [y[i]-trend[i] for i in range(0, len(ts))]

In [34]: plt.figure(figsize = (20,10))

         plt.plot(detrended)
         plt.show()
```



```
In [35]: plt.figure(figsize = (20,10))

         X = ts['pasmio'].values

         diff = list()
         for i in range(1, len(X)):
             value = X[i] - X[i - 1]
             diff.append(value)
         plt.plot(diff)
         plt.show()
```

Fig 5.4 Coding screen shots

CHAPTER 6
RESULT ANALYSIS

This model is built with anaconda, Jupyter Notebook. The file covers historical passenger statistics for the previous thirteen years. This data-set was obtained through Kaggle. The data-set includes four thousand occurrences and sixteen characteristics, including the location's name, year, months, and coordinates (Latitude and Longitude). The above-mentioned algorithm is used to train and test the data. As a result, the commuter flow is anticipated by this model. The anticipated graph depicts commuter movement as a Line Graph. It depicts the number of commuters in millions, while the horizontal axis represents the length of time in which they were on the train. The "Original data" graph represents the actual commuters flow.

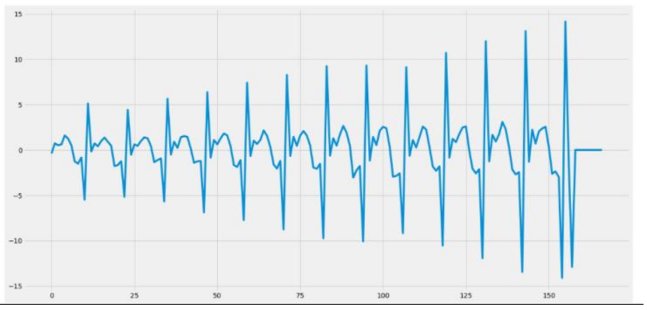


Fig 6.1: ORIGINAL FLOW

The designed model produced accurate results.The "Predicted data" graph depicts predicted people flow. As a result, this model can aid in anticipating commuter flow.

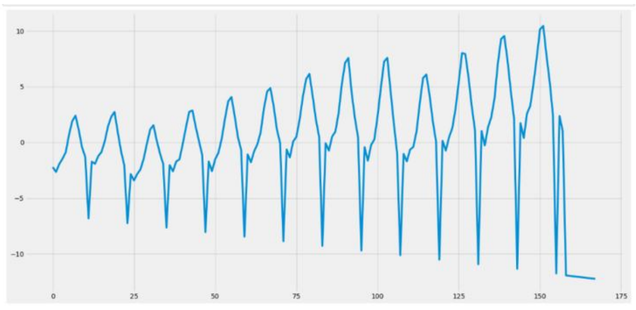


Fig 6.2 : PREDICTED FLOW

The designed model produced accurate results when compared to the existing model. The graph "original data" depicts the real flow of people in a metro station. The "Predicted data" graph depicts predicted people flow. As a result, this model can aid in anticipating commuter flow.

CHAPTER 7
CONCLUSION AND FUTUREWORK

The proposed work aims at predicting the commuters flow.The proposed work is capable of remembering over lengthy sequences, hence overcoming back propagating decay. It is possible to estimate the movement of commuters at a high frequency in a short period of time using this network design. Future iterations of the system will include real-time data from the Internet of Things (IOT).

REFERENCES

[1] Demianiuk, V., Gorinsky, S., Nikolenko, S. I., & Kogan, K. (2020). Robust distributed monitoring of traffic flows. IEEE/ACM Transactions on Networking, 29(1), 275-288.

[2] Celesti, A., Galletta, A., Carnevale, L., Fazio, M., Ĺay-Ekuakille, A., & Villari, M. (2017). An IoT cloud system for traffic monitoring and vehicular accidents prevention based on mobile sensor data processing. IEEE Sensors Journal, 18(12), 4795-4802.

[3] Wang, W., Chen, J. C., & Wu, Y. J. (2019). The prediction of freeway traffic conditions for logistics systems. IEEE Access, 7, 138056-138061.

[4] Tangari, G., Charalambides, M., Tuncer, D., & Pavlou, G. (2020). Accuracy-Aware Adaptive Traffic Monitoring for Software Dataplanes. IEEE/ACM Transactions on Networking, 28(3), 986-1001.

[5] Ali, M. E. M., Durdu, A., Çeltek, S. A., & Yilmaz, A. (2021). An adaptive method for traffic signal control based on fuzzy logic with webster and modified webster formula using SUMO traffic simulator. IEEE Access, 9, 102985-102997.

[6] Ruiz, M., Coltraro, F., & Velasco, L. (2018). CURSA-SQ: A methodology for service-centric traffic flow analysis. Journal of optical communications and networking, 10(9), 773-784.

[7] Guidoni, D. L., Maia, G., Souza, F. S., Villas, L. A., & Loureiro, A. A. (2020). Vehicular traffic management based on traffic engineering for vehicular ad hoc networks. IEEE Access, 8, 45167-45183.

[8] Guo, W., Li, J., Liu, X., & Yang, Y. (2020). Privacy-preserving compressive sensing for real-time traffic monitoring in urban city. IEEE Transactions on Vehicular Technology, 69(12), 14510-14522.

[9] Zhang, R., Newman, S., Ortolani, M., & Silvestri, S. (2018). A network tomography approach for traffic monitoring in smart cities. IEEE Transactions on Intelligent Transportation Systems, 19(7), 2268-2278.

[10] Lu, R., Liang, X., Li, X., Lin, X., & Shen, X. (2012). EPPA: An efficient and privacy-preserving aggregation scheme for secure smart grid communications. IEEE Transactions on Parallel and Distributed Systems, 23(9), 1621-1631.

[11] Basudan, S., Lin, X., & Sankaranarayanan, K. (2017). A privacy-preserving vehicular crowdsensing-based road surface condition monitoring system using fog computing. *IEEE Internet of Things Journal*, *4*(3), 772-782.

[12] He, Y., Sun, L., Li, Z., Li, H., & Cheng, X. (2014, August). An optimal privacy-preserving mechanism for crowdsourced traffic monitoring. In Proceedings of the 10th ACM international workshop on Foundations of mobile computing (pp. 11-18).

[13] Fallgren, M., Abbas, T., Allio, S., Alonso-Zarate, J., Fodor, G., Gallo, L., ... & Vivier, G. (2019). Multicast and broadcast enablers for high-performing cellular V2X systems. IEEE Transactions on Broadcasting, 65(2), 454-463.

[14] Epstein, R. A., Patai, E. Z., Julian, J. B., & Spiers, H. J. (2017). The cognitive map in humans: spatial navigation and beyond. Nature neuroscience, 20(11), 1504-1513.

[15] Li, Y., Gai, K., Qiu, L., Qiu, M., & Zhao, H. (2017). Intelligent cryptography approach for secure distributed big data storage in cloud computing. Information Sciences, 387, 103-115.

[16] Lu, S., Knoop, V. L., & Keyvan-Ekbatani, M. (2018). Using taxi GPS data for macroscopic traffic monitoring in large scale urban networks: calibration and MFD derivation. Transportation research procedia, 34, 243-250.

[17] Zhou, P., Zheng, Y., & Li, M. (2012, June). How long to wait? Predicting bus arrival time with mobile phone based participatory sensing. In Proceedings of the 10th international conference on Mobile systems, applications, and services (pp. 379-392).

[18] Oh, S., Ritchie, S. G., & Oh, C. (2001). Real time traffic measurement from single loop inductive signatures.

[19] Cao, C., Li, C., Yang, Q., Liu, Y., & Qu, T. (2018). A novel multi-objective programming model of relief distribution for sustainable disaster supply chain in large-scale natural disasters. Journal of Cleaner Production, 174, 1422-1435.

[20] Wu, Y. J., & Chen, J. C. (2021). A structured method for smart city project selection. International Journal of Information Management, 56, 101981.

[21] Wang, X., Fang, Y., Liu, Y., & Horn, B. (2018, October). A survey on the status of open data and its future. In 2018 4th International Conference on Universal Village (UV) (pp. 1-4). IEEE.