A project report on

FLIGHT NETWORK ANALYSIS

Submitted in partial fulfillment for the course

Social and Information Networks

by

ARNAB MONDAL (20BCE1294)

SAPTARSHI MUKHERJEE (20BCE1719)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

April, 2023

FACULTY SIGN

Dr PUNITHA K

VIT
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

DECLARATION

We hereby declare that the thesis entitled “FLIGHT NETWORK ANALYSIS ” submitted by Arnab and Saptarshi, for the completion of the course, Social and Information Network (CSE3021) is a record of bonafide work carried out by us under the supervision of Dr Punitha K, our course instructor. We further declare that the work reported in this document has not been submitted and will not be submitted, either in part or in full, for any other courses in this institute or any other institute or university.

Place: Chennai

Date: Signature of the Candidate

School of Computer Science and Engineering

CERTIFICATE

This is to certify that the report entitled **"FLIGHT NETWORK ANALYSIS"** is prepared and submitted by **ARNAB MONDAL** (**20BCE1294**) and **SAPTARSHI MUKHERJEE (20BCE1719)** to Vellore Institute of Technology, Chennai, in partial fulfillment of the requirement for the course, **Social and Information Networks (CSE3021),** is a bonafide record carried out under my guidance. The project fulfills the requirements as per the regulations of this University and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the any other course and the same is certified.

Name: Dr. Punitha K Signature of the Faculty

Date: 10.04.23

ABSTRACT

This comprehensive and ambitious project employs advanced graph theory and cutting-edge network analysis techniques to conduct a rigorous analysis of a vast and complex dataset related to the air transportation industry. The study seeks to gain deeper insights into the intricate relationships and interdependencies between the diverse elements of the air transportation network, including airports, carriers, and routes. By representing these relationships in a graph-based framework, the study is able to reveal hidden patterns and trends in the distribution and flow of air traffic across the world. The findings of this research are of great significance, as they shed new light on the structure and behavior of the air transportation system, including the concentration of air traffic, the distribution of flights, and the identification of key players in the industry. Furthermore, this study has important implications for the global economy and environment. The air transportation system is a critical component of the global economy, facilitating the movement of people and goods across vast distances. However, it also has a significant impact on the environment, contributing to climate change and other environmental issues. By gaining a deeper understanding of the air transportation system through this network analysis, policymakers, researchers, and industry leaders can develop more effective strategies for addressing these complex challenges. The study provides a valuable contribution to the ongoing discourse on the future of air transportation, and serves as a model for future research in this area.

CONTENTS

LIST OF FIGURES

LIST OF ACRONYMS

Acronym	Phrase
USA	United States of America
ML	Machine Learning
Kaggle	Knowledge and Data Discovery Platform
DepTime	Departure Time
ArrTime	Arrival Time
UniqueCarrier	Unique Carrier Identifier
FlightNum	Flight Number
TailNum	Tail Number
ActualElapsedTime	Actual Elapsed Time
AirTime	Air Time
Origin	Origin Airport
Dest	Destination Airport
SQL	Structured Query Language
Excel	Microsoft Excel
US	United States
GDP	Gross Domestic Product
EU	European Union
NGO	Non-Governmental Organization
CEO	Chief Executive Officer
NN	Networkx Library

G20	Group of Twenty (international forum consisting of 19 countries and the European Union)
FNA	Flight Network Analysis
GDPR	General Data Protection Regulation
ATP	Air Transportation System
NA	Not Available
GT	Graph Theory
RT	Route Optimization
TM	Traffic Management
SCM	Supply Chain Management

Chapter 1

INTRODUCTION

1.1 INTRODUCTION

Air transportation is an intricately woven tapestry that connects people, goods, and services across the globe, playing a pivotal role in the functioning of the global economy. As air transportation networks continue to expand and evolve, understanding their complex structure and behavior is essential to improving their efficiency and sustainability. Network analysis, a sophisticated method that leverages graph theory and network analysis techniques, is a powerful tool for analyzing complex systems such as air transportation networks. This ambitious project aims to conduct an in-depth network analysis on a vast and comprehensive flights dataset to unravel the intricate workings of the air transportation system. Through the application of cutting-edge analytical techniques, the study will provide a detailed understanding of the distribution of flights, the concentration of air traffic, and the identification of key players in the industry. The study's findings will be instrumental in improving our understanding of the air transportation system's underlying mechanisms, providing critical insights into its impact on the global economy and environment. The research will shed light on critical factors such as route optimization, capacity planning, and traffic management, enabling policymakers, aviation stakeholders, and the general public to make informed decisions that contribute to the industry's sustainability and growth. Moreover, the study will provide insights into the air transportation system's interdependence with other systems, such as logistics, supply chain management, and tourism, highlighting the complex web of interactions that shape the global

economy. The project will have far-reaching implications for decision-making processes across various sectors, including business, trade, and tourism. By providing a comprehensive understanding of the air transportation system, this project will facilitate the development of innovative solutions that improve the industry's efficiency, reduce its environmental footprint, and enhance its role as a catalyst for global economic growth. Ultimately, the study's results will create a positive impact on society by improving access to goods and services, facilitating global trade and commerce, and fostering greater cultural exchange and understanding.

1.2 OVERVIEW of FLIGHT NETWORK ANALYSIS

In this section, we present the outline of the proposed system. The experiment utilized datasets sourced from kaggle.com, an online platform for accessing databases. The chosen dataset pertains to flight data in the United States of America (USA). All aspects of our implementation, visualization, and conclusion are based solely on this dataset. To construct the system, we employed the networkx library, a Python-based tool designed for studying graphs and networks. The Flight Network Analysis enables us to uncover insights about flights, such as the busiest airport, the shortest path between airports, and connecting flights. The system also includes a unique feature for identifying the nearest airports based on user-defined distance.

1.3 CHALLENGES PRESENT

The implementation of a flight data analysis system may encounter several challenges, such as issues related to the quality and completeness of the data obtained from kaggle.com. The accuracy

and completeness of the dataset may be questionable, which could lead to difficulties in implementing the system accurately. Additionally, the interpretation of results may be challenging due to the complexity of the data, which may require specialized knowledge to fully comprehend. Another challenge is related to privacy concerns, as the use of flight data, even when anonymized, may raise concerns about the protection of individual passenger data. It will be crucial to handle the data carefully to ensure that privacy is not compromised. Furthermore, limitations in data availability may arise when data related to new airports is not available. Inaccurate data may lead to conflicting results and incomplete information, and the shortest path or airport data may be removed or abandoned due to real-world events. Addressing these challenges will require careful consideration and adaptation of the proposed system to ensure its accuracy and usefulness in analyzing flight data.

1.4 PROJECT STATEMENT

The project aims to develop a Flight Network Analysis system that utilizes datasets from kaggle.com, based on USA flight data, to provide insights about flights such as the busiest airports, shortest path between airports, and connecting flights. The system includes a unique feature of finding the nearest airports based on the distance entered. The project implementation involves preprocessing of data, generating nodes and underlying relationships, calculating centrality measures, and finding the shortest path and alternate paths between airports.

1.5 OBJECTIVES

The objective of the project is to provide underlying insights about flight data, identify patterns to help airlines increase their business,

enhance customer experience, and support the development of more airports to make air travel feasible and cheap while building a sustainable environment, generating employment, and improving financial experience under the G20 Summit objective.

1.6 SCOPE OF THE PROJECT

The project includes the use of the networkx library, a Python library for studying graphs and networks, to generate and visualize the flight network graph. The system will focus solely on the USA flight data obtained from kaggle.com. The system will provide various features such as finding busy airports, calculating the degree of a node, including the full names of the airports from their codes, and finding the nearest airport based on the airport name and the distance between the airports. The system will analyze the flight data to generate insights about the busiest airports, shortest paths between airports, and connecting flights. The project's scope also involves supporting the development of more airports, making air travel more feasible and cheaper while building a sustainable environment, generating employment, and improving financial experience under the G20 Summit objective.

Chapter 2

BACKGROUND

2.1 INTRODUCTION

In recent years, flight networks have become the subject of growing interest among researchers across various disciplines due to their complexity and importance in fields such as transportation, economics, and geography. Several studies have investigated the topology, robustness, traffic flows, and economic impact of these networks. Notably, researchers have found that the global airport network follows a scale-free structure, with a few airports having many connections while others have few. Additionally, the US domestic flight network is highly vulnerable to extreme weather events, highlighting the need for better management and contingency planning. Spatial and temporal patterns of air traffic have also been explored, with significant variations observed across regions and time periods. Further, air connectivity has been found to have a positive impact on regional economic growth, particularly in underdeveloped areas. These findings have important implications for improving the design and management of the global air transportation system. In addition to academic benefits, the analysis of flight networks has practical applications such as reducing fuel consumption by providing the shortest path between two nodes, offering connecting flights, and providing a feature that allows individuals to view all nearby airports to plan their travel more efficiently.

2.2 SURVEY

Air transportation has become a crucial part of modern society, connecting people and goods across the world. The analysis of the

air traffic network has gained significant attention in the scientific community in recent years. Several studies have used network theory to analyze the structure, connectivity, and efficiency of the global air transport network. Gabriella Schoier and Giuseppe Borusso (2014) used network theory to analyze the European air traffic network, finding that it is highly connected and efficient but also vulnerable to disruptions. They emphasized the importance of studying the network's robustness to ensure the continuity of air transport services. Massimiliano Zanin and Fabrizio Lillo (2013) examined the global air transport network, using complex network theory. They found that the network exhibits small-world properties and scale-free behavior, with a few highly connected airports serving as hubs. This study highlighted the importance of hub airports in the air transportation network. Oriol Lordon, Jose M Sallan, and Pep Simo (2014) also used network theory to analyze the global air transportation network. They focused on the network's topology, spatial distribution, and dynamics, finding that the network has become more connected and concentrated over time, with a few dominant hubs in the network. Chun-Hsiang Chan (2021) analyzed the spatial characteristics of airline alliances using network analysis. They focused on identifying spatial patterns of market regions in three airline alliances (Oneworld, Star Alliance, and SkyTeam) and characterized the differences among them, highlighting the regions of collaboration, competition, and dominance. This study provides insights for airlines and airline alliances to extend their market share or service areas. Weisi Guo, Bogdan Toader, Roxana Feier, Guillem Mosquera, Fabian Ying, Se-Wook Oh, Matthew Price-Williams, and Armin Krupp (2019) reviewed existing research on the global air transport network and analyzed it using two key approaches: a top-down multi-scale network science approach and a bottom-up entropy-maximization

interaction network approach. Using simple socioeconomic indicators, they constructed a very accurate interaction model that can predict traffic volume and estimate the impact of population growth or fuel cost. They also identified community structures and related them to economic outputs and saw how hubs evolved over time to become more influential. Overall, these studies have contributed to our understanding of the air transport network's structure, connectivity, and dynamics. They provide insights into the role of hub airports, the importance of studying network robustness, and the characteristics of airline alliances. These findings are essential for policymakers, airlines, and airport authorities to optimize the global air transportation network's efficiency and resilience.

Chapter 3

OVERVIEW

In this section, the overview of the proposed system is presented. The datasets utilized for this experiment were acquired from kaggle.com, a renowned online platform for accessing databases. To implement the system, an array of cutting-edge algorithms and sophisticated Python libraries were employed, enabling the system to deliver remarkable outcomes. It is important to note that the results obtained from this implementation heavily depend on the quality and quantity of data available in the dataset. With the flight network analysis, a plethora of useful information can be gleaned such as the busiest airport, the shortest path between airports, efficiency, airports within a state, and the nearest airport from a specific location. These insights can provide immense benefits to both customers and airlines alike. Customers can make more informed decisions about their travel arrangements, while airlines can optimize their operations by identifying high-traffic areas and optimizing routes. Therefore, this system has the potential to revolutionize the airline industry, leading to more efficient and seamless air travel experiences for all.

3.1 DATASET

This project utilizes datasets sourced from Kaggle datasets, an immensely popular online platform for acquiring diverse datasets. These datasets contain several columns, each of which provides essential information about flights operating in the United States of America (USA) during a specific year.

The DepTime and ArrTime columns indicate the departure and arrival times of the flights, respectively, enabling the analysis of

punctuality and timeliness. The UniqueCarrier column provides the carrier's unique identifier, allowing for the identification of different airlines operating in the region. Simultaneously, the FlightNum column provides the flight number, allowing for a more granular analysis of individual flights.

The TailNum column, on the other hand, indicates the tail number of the aircraft used for the flight, offering insights into the types of aircraft used and their performance metrics. The ActualElapsedTime and AirTime columns provide data on the actual elapsed time and airtime of the flight, respectively, providing insights into the efficiency of operations.

The Origin and Dest columns indicate the origin and destination airports, respectively, allowing for the identification of high-traffic areas and their associated airlines. Finally, the Distance column provides data on the distance traveled by the aircraft during the flight, enabling analysis of fuel efficiency and optimization of flight routes.

With such comprehensive information, this dataset is a valuable resource for the analysis and optimization of airline operations in the USA. By leveraging the insights gained from this dataset, airlines can improve their operations, providing better customer experiences and more efficient operations. Overall, this dataset holds immense potential for transforming the airline industry in the USA, leading to a more seamless and efficient air travel experience for all.

Chapter 4

PROPOSED SYSTEM

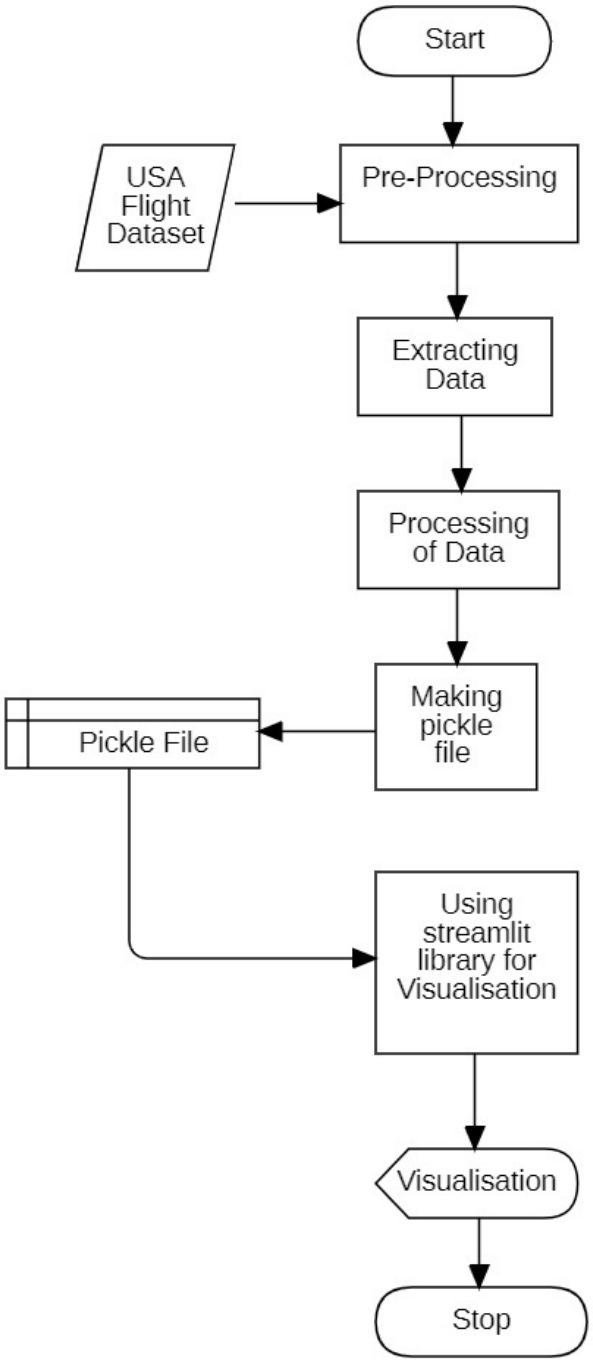

Fig 4.1: Process Flow Diagram

PRE-PROCESSING

Data pre-processing is a crucial step in Machine Learning that involves preparing raw data to make it suitable for building and training Machine Learning models. In the case of the USA flight dataset used in this project, data preprocessing includes cleaning the data, dropping unwanted columns, and removing null values. By carrying out these preprocessing steps, the data becomes more structured and suitable for accurate analysis and modeling, ultimately leading to more effective Machine Learning models.

EXTRACTING DATA

Extracting data from a dataset involves retrieving relevant information from a larger dataset by applying specific filters or conditions. This process can be done manually or through automated methods using programming languages or tools such as SQL, Python, or Excel. The extracted data can then be utilized for various purposes, such as analysis, reporting, or training machine learning models. It is an essential step in data processing that allows for efficient and effective data management. In the case of the USA flight dataset, the data extraction process involves identifying and storing the unique airport names in a list for further analysis and modeling purposes.

PROCESSING OF DATA

Processing data in machine learning involves transforming and organizing data for analysis and model development. This includes tasks such as data cleaning, feature engineering, and normalization. It is a critical step that directly impacts model accuracy. In the USA flight dataset, processing data includes declaring nodes for each airport, finding total nodes, making graphical representations, finding centrality measures, and calculating shortest path distances.

Other tasks include finding the busiest airports, printing airport flight operations, making distance matrices, finding airports near a particular airport, among others.

MAKING PICKLE FILE

A pickle file is a serialized object that stores and retrieves Python objects. It is used in machine learning to save trained models and to share data between different Python applications. However, it is important to be cautious when unpickling data from untrusted sources. In the case of the USA flight dataset, pickle is used to store airport names, airport codes, graph, nearest airport, state codes, and state names

VISUALISATION

Visualizing data and models is essential in ML for gaining insights and improving performance by identifying patterns, relationships, and issues. Visualization creates graphical representations using tools like scatter plots, histograms, heatmaps, etc. Streamlit Python library can be used to create interactive web pages for visualization, as seen in the USA flight dataset.

4.2 LIBRARIES

The following libraries are used in the context at hand to accomplish specific tasks:

- networkx: A Python library that enables the creation, manipulation, and study of complex networks, such as graphs and nodes. It offers an extensive set of tools for analyzing and visualizing network data, which is particularly useful for various Machine Learning applications.
- numpy: A widely used Python library for scientific computing, particularly in the fields of Mathematics, Physics, and Engineering. It provides an efficient interface for handling large multi-

dimensional arrays and matrices, along with an extensive set of mathematical functions to perform operations on them.

- matplotlib.pyplot: A Python library for creating high-quality visualizations, including line plots, scatter plots, histograms, and bar charts. It provides a flexible interface for customizing plots with various attributes, such as colors, fonts, and labels.
- pandas: A data manipulation library that provides easy-to-use data structures and data analysis tools for working with structured data. It allows for efficient data ingestion, cleaning, and transformation, along with flexible data querying and aggregation capabilities.
- pickle: A built-in Python module for object serialization and deserialization, which allows for the conversion of complex data structures into a format that can be easily stored and retrieved from disk.
- streamlit: A Python library for building interactive web applications, particularly in the field of Machine Learning. It offers a simple and intuitive interface for creating customizable web pages, incorporating data visualizations, and interacting with Machine Learning models in real-time.

Chapter 5

METHODOLOGY

5.1 METHODOLOGY

In this project, we utilized several methods from the networkx library in Python to perform data processing and generate the final graph, dataset, and all necessary data-related tasks. The obtained insights were then stored in a pickle file to avoid additional processing since it can be both costly and time-consuming. To present the results in a user-friendly manner, we employed the streamlit library, which allows for limited interactivity with the web page to deliver customized results to the user.

Chapter 6

IMPLEMENTATION

6.1 PROCESSING

Code:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import networkx as nx
G = nx.MultiDiGraph()
my_dataset = pd.read_csv('airlinedelaycauses_DelayedFlights.csv',
low_memory=False)
my_dataset
```

	Unnamed: 0	DepTime	ArrTime	UniqueCarrier	FlightNum	TailNum	ActualElapsedTime	AirTime	Origin	Dest	Distance
0	0	2003.0	2211.0	WN	335	N712SW	128.0	116.0	IAD	TPA	810
1	1	754.0	1002.0	WN	3231	N772SW	128.0	113.0	IAD	TPA	810
2	2	628.0	804.0	WN	448	N428WN	96.0	76.0	IND	BWI	515
3	4	1829.0	1959.0	WN	3920	N464WN	90.0	77.0	IND	BWI	515
4	5	1940.0	2121.0	WN	378	N726SW	101.0	87.0	IND	JAX	688
...	...	...	...	...	...	...	...	...	...	...	...
1936753	7009710	1250.0	1617.0	DL	1621	N938DL	147.0	120.0	MSP	ATL	906
1936754	7009717	657.0	904.0	DL	1631	N3743H	127.0	78.0	RIC	ATL	481
1936755	7009718	1007.0	1149.0	DL	1631	N909DA	162.0	122.0	ATL	IAH	689
1936756	7009726	1251.0	1446.0	DL	1639	N646DL	115.0	89.0	IAD	ATL	533
1936757	7009727	1110.0	1413.0	DL	1641	N908DL	123.0	104.0	SAT	ATL	874

1936758 rows × 11 columns

Fig 6.1.1: Raw Dataset

Code:

```
my_dataset =
my_dataset.drop(['Year','Month','DayofMonth','CRSDepTime','CR
SArrTime','CRSElapsedTime','ArrDelay','DepDelay','Cancelled','C
ancellationCode','Diverted','CarrierDelay','WeatherDelay','NASDel
ay','SecurityDelay','LateAircraftDelay','TaxiIn','TaxiOut','DayOfW
eek'],axis=1)
```

my_dataset

	Unnamed: 0	DepTime	ArrTime	UniqueCarrier	FlightNum	TailNum	ActualElapsedTime	AirTime	Origin	Dest	Distance
0	0	2003.0	2211.0	WN	335	N712SW	128.0	116.0	IAD	TPA	810
1	1	754.0	1002.0	WN	3231	N772SW	128.0	113.0	IAD	TPA	810
2	2	628.0	804.0	WN	448	N428WN	96.0	76.0	IND	BWI	515
3	4	1829.0	1959.0	WN	3920	N464WN	90.0	77.0	IND	BWI	515
4	5	1940.0	2121.0	WN	378	N726SW	101.0	87.0	IND	JAX	688
...	...	...	...	...	...	...	...	...	...	...	...
1936753	7009710	1250.0	1617.0	DL	1621	N938DL	147.0	120.0	MSP	ATL	906
1936754	7009717	657.0	904.0	DL	1631	N3743H	127.0	78.0	RIC	ATL	481
1936755	7009718	1007.0	1149.0	DL	1631	N909DA	162.0	122.0	ATL	IAH	689
1936756	7009726	1251.0	1446.0	DL	1639	N646DL	115.0	89.0	IAD	ATL	533
1936757	7009727	1110.0	1413.0	DL	1641	N908DL	123.0	104.0	SAT	ATL	874

1936758 rows × 11 columns

Fig 6.1.2: Dataset after removing Unwanted columns.

Code:

```
my_dataset = my_dataset.dropna()
#thresh=half_count,
my_dataset
```

	Unnamed: 0	DepTime	ArrTime	UniqueCarrier	FlightNum	TailNum	ActualElapsedTime	AirTime	Origin	Dest	Distance
0	0	2003.0	2211.0	WN	335	N712SW	128.0	116.0	IAD	TPA	810
1	1	754.0	1002.0	WN	3231	N772SW	128.0	113.0	IAD	TPA	810
2	2	628.0	804.0	WN	448	N428WN	96.0	76.0	IND	BWI	515
3	4	1829.0	1959.0	WN	3920	N464WN	90.0	77.0	IND	BWI	515
4	5	1940.0	2121.0	WN	378	N726SW	101.0	87.0	IND	JAX	688
...	...	...	...	...	...	...	...	...	...	...	...
1936753	7009710	1250.0	1617.0	DL	1621	N938DL	147.0	120.0	MSP	ATL	906
1936754	7009717	657.0	904.0	DL	1631	N3743H	127.0	78.0	RIC	ATL	481
1936755	7009718	1007.0	1149.0	DL	1631	N909DA	162.0	122.0	ATL	IAH	689
1936756	7009726	1251.0	1446.0	DL	1639	N646DL	115.0	89.0	IAD	ATL	533
1936757	7009727	1110.0	1413.0	DL	1641	N908DL	123.0	104.0	SAT	ATL	874

1928368 rows × 11 columns

Fig 6.1.3: Dataset after dropping NA values.

Code:

```
uniqueValues =
(my_dataset['Origin'].append(my_dataset['Dest'])).unique
()
print(uniqueValues)
```

```
/var/folders/db/1rbb5sxj5d9b_5cpqjpvf7qw0000gn/T/ipykernel_7167/3508961446.py:1: FutureWarning: The series.append method is deprecated and will be remo
  uniqueValues = (my_dataset['Origin'].append(my_dataset['Dest'])).unique()
['IAD' 'IND' 'ISP' 'JAN' 'JAX' 'LAS' 'LAX' 'LBB' 'LIT' 'MAF' 'MCI' 'MCO'
 'MDW' 'MHT' 'MSY' 'OAK' 'OKC' 'OMA' 'ONT' 'ORF' 'PBI' 'PDX' 'PHL' 'PHX'
 'PIT' 'PVD' 'RDU' 'RNO' 'RSW' 'SAN' 'SAT' 'SDF' 'SEA' 'SFO' 'SJC' 'SLC'
 'SMF' 'SNA' 'STL' 'TPA' 'TUL' 'TUS' 'ABQ' 'ALB' 'AMA' 'AUS' 'BDL' 'BHM'
 'BNA' 'BOI' 'BUF' 'BUR' 'BWI' 'CLE' 'CMH' 'CRP' 'DAL' 'DEN' 'DTW' 'ELP'
 'FLL' 'GEG' 'HOU' 'HRL' 'ROC' 'ORD' 'EWR' 'SYR' 'IAH' 'CRW' 'FAT' 'COS'
 'MRY' 'LGB' 'BFL' 'EUG' 'ICT' 'MEM' 'BTV' 'MKE' 'LFT' 'BRO' 'PWM' 'MSP'
 'SRQ' 'CLT' 'CVG' 'GSO' 'SHV' 'DCA' 'TYS' 'GSP' 'RIC' 'DFW' 'BGR' 'DAY'
 'GRR' 'CHS' 'CAE' 'TLH' 'XNA' 'GPT' 'VPS' 'LGA' 'ATL' 'MSN' 'SAV' 'BTR'
 'LEX' 'LRD' 'MOB' 'MTJ' 'GRK' 'AEX' 'PNS' 'ABE' 'HSV' 'CHA' 'MFE' 'MLU'
 'DSM' 'MGM' 'AVL' 'LCH' 'BOS' 'MYR' 'CLL' 'DAB' 'ASE' 'ATW' 'BMI' 'CAK'
 'CID' 'CPR' 'EGE' 'FLG' 'FSD' 'FWA' 'GJT' 'GRB' 'HNL' 'KOA' 'LAN' 'LIH'
 'MBS' 'MDT' 'MFR' 'OGG' 'PSP' 'RAP' 'ROA' 'SBA' 'SBN' 'SPI' 'TEX' 'YUM'
 'AVP' 'CWA' 'DRO' 'ILM' 'ITO' 'JFK' 'TVC' 'HDN' 'SBP' 'CLD' 'HPN' 'MIA'
 'SGF' 'TRI' 'SUN' 'SGU' 'SLE' 'MSO' 'BZN' 'YKM' 'GTF' 'BIL' 'JAC' 'MOD'
 'FAR' 'GUC' 'AZO' 'PIA' 'MLI' 'LNK' 'FCA' 'IDA' 'BIS' 'PSC' 'COD' 'TWF'
 'RDM' 'HLN' 'RDD' 'PIH' 'EKO' 'LWS' 'SMX' 'ACV' 'OXR' 'CEC' 'CIC' 'PMD'
 'RFD' 'BTM' 'IYK' 'IPL' 'BLI' 'CDC' 'SJU' 'STT' 'ANC' 'STX' 'MLB' 'PHF'
 'PFN' 'FAY' 'AGS' 'GNV' 'ABY' 'DHN' 'EVV' 'FNT' 'OAJ' 'TOL' 'SWF' 'EWN'
 'MEI' 'GTR' 'LYH' 'HHH' 'EYW' 'VLD' 'CSG' 'ACY' 'FSM' 'MCN' 'CHO' 'FLO'
 'BQK' 'SCE' 'TUP' 'TYR' 'LAW' 'SPS' 'ABI' 'GGG' 'ACT' 'SJT' 'TXK' 'CMI'
 'ROW' 'RST' 'MQT' 'LSE' 'DBQ' 'GFK' 'MOT' 'DLH' 'SUX' 'PLN' 'BGM' 'ERI'
 'ALO' 'CMX' 'RHI' 'ELM' 'JNU' 'KTN' 'SIT' 'PSG' 'CDV' 'YAK' 'BRW' 'OME'
 'OTZ' 'BET' 'FAI' 'SCC' 'ADQ' 'WRG' 'ADK' 'PSE' 'BQN' 'BPT' 'RKS' 'GCC'
 'MKG' 'DLG' 'AKN' 'LWB' 'ACK' 'WYS' 'BJI' 'INL' 'GST' 'PUB' 'OTH' 'LMT'
 'ITH' 'HTS' 'PIR']
```

Fig 6.1.4: Unique Airport Codes.

Code:

```
print(type(uniqueValues))
import pickle
with open('airport.pkl', 'wb') as m:
    pickle.dump(uniqueValues, m)
for airport in uniqueValues:
    G.add_node(airport, name=airport)
G.number_of_nodes()
nx.draw(G, with_labels=True ,node_size=0.1)
```

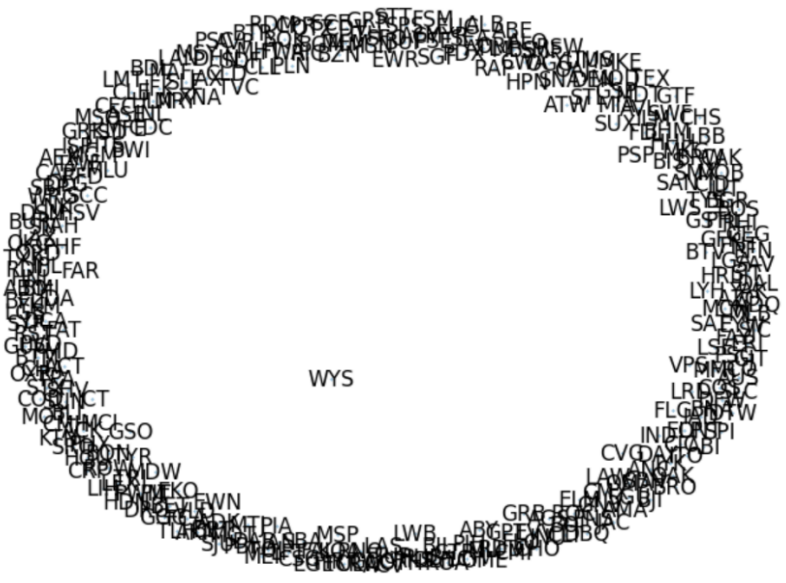

Fig 6.1.5: Graphical Representation of Nodes.

Code:

```
for flight in my_dataset.index:
    # print (my_dataset['FlightNum'][train])
    #                   G.add_edge(my_dataset['Origin'],
my_dataset['Dest'],
weight=my_dataset['ActualElapsedTime'])
    #            G.add_edge(my_dataset['Origin'][train],
my_dataset['Dest'][train],
l=my_dataset['ActualElapsedTime'][train],flight=my_datas
et['FlightNum'][train])

    if    not    G.has_edge(my_dataset['Origin'][flight],
my_dataset['Dest'][flight]):
        G.add_edge(my_dataset['Origin'][flight],
my_dataset['Dest'][flight],
l=my_dataset['ActualElapsedTime'][flight],flight=my_data
set['FlightNum'][flight])
    elif
G[my_dataset['Origin'][flight]][my_dataset['Dest'][fligh
t]][0]["l"] > my_dataset['ActualElapsedTime'][flight]:

G[my_dataset['Origin'][flight]][my_dataset['Dest'][fligh
t]][0]["l"] = my_dataset['ActualElapsedTime'][flight]

G[my_dataset['Origin'][flight]][my_dataset['Dest'][fligh
t]][0]["flight"] = my_dataset['FlightNum'][flight]
G.number_of_edges()
G.number_of_nodes()
```

```
13]  G.number_of_edges()

··   5127
```

```
14]  G.number_of_nodes()

··   303
```

Fig 6.1.6: Number of Nodes and Edges.

Code:

```
nx.draw(G, with_labels=False , node_size=0.1)
```

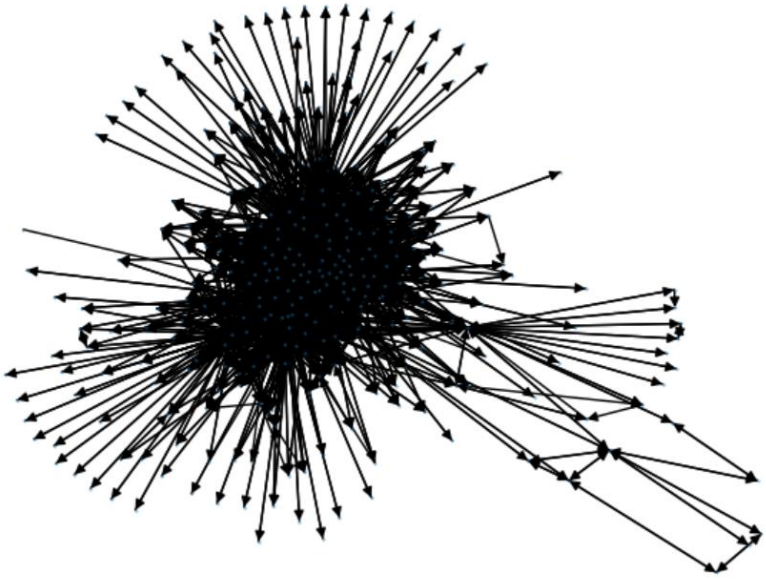

Fig 6.1.7: Graphical Representation.

Code:

nx.degree_centrality(G)

```
Output exceeds the size limit. Open the full output data in a text editor
{'IAD': 0.46357615894039733,
 'IND': 0.25165562913907286,
 'ISP': 0.059602649006622516,
 'JAN': 0.08609271523178808,
 'JAX': 0.24172185430463577,
 'LAS': 0.6026490066225165,
 'LAX': 0.5794701986754967,
 'LBB': 0.046357615894039736,
 'LIT': 0.12582781456953643,
 'MAF': 0.046357615894039736,
 'MCI': 0.33112582781456956,
 'MCO': 0.5728476821192053,
 'MDW': 0.3576158940397351,
 'MHT': 0.11589403973509933,
 'MSY': 0.25165562913907286,
 'OAK': 0.2185430463576159,
 'OKC': 0.19205298013245034,
 'OMA': 0.1490066225165563,
 'ONT': 0.2119205298013245,
 'ORF': 0.16887417218543047,
 'PBI': 0.17218543046357615,
 'PDX': 0.2913907284768212,
 'PHL': 0.3609271523178808,
 'PHX': 0.5827814569536424,
 'PIT': 0.2152317880794702,
```

Fig 6.1.8: Degree Centrality.

Code:

nx.betweenness_centrality(G)

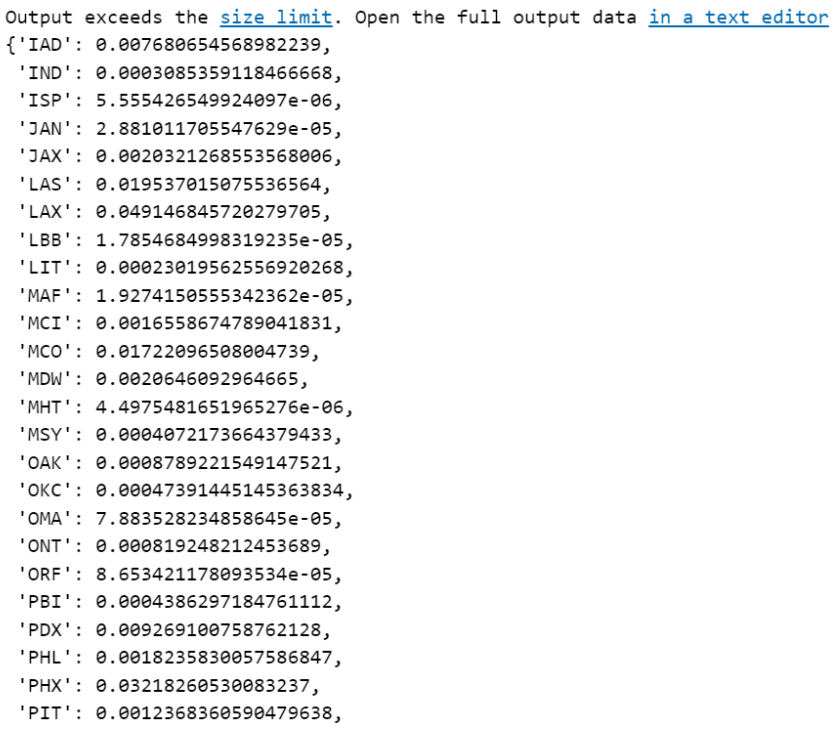

Fig 6.1.9: Betweenness Centrality.

Code:

```
l=list(G.degree(list(G.nodes())))
l.sort(key=lambda x: x[1], reverse=True)

l[:10] #top 10 nodes with highest degree (stations with most number of flights)
```

```
[('ATL', 345),
 ('ORD', 295),
 ('DFW', 268),
 ('MSP', 251),
 ('DEN', 242),
 ('DTW', 235),
 ('IAH', 226),
 ('CVG', 222),
 ('SLC', 220),
 ('LAS', 182)]
```

Fig 6.1.10: Top 10 busiest airports.

Code:

```
df  =pd.DataFrame(nx.adjacency_matrix(G,  weight="l").todense(),
index=G.nodes(), columns=G.nodes())
df
```

	IAD	IND	ISP	JAN	JAX	LAS	LAX	LBB	LIT	MAF	...	WYS	BJI	INL	GST	PUB	OTH	LMT	ITH	HTS	PIR
IAD	0.0	0.0	0.0	0.0	97.0	269.0	284.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
IND	0.0	0.0	0.0	0.0	90.0	212.0	245.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ISP	0.0	0.0	0.0	0.0	0.0	298.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
JAN	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
JAX	88.0	104.0	0.0	0.0	0.0	258.0	324.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	213.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
OTH	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
LMT	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ITH	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
HTS	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
PIR	0.0	0.0	0.0	0.0	199.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

303 rows × 303 columns

Fig 6.1.11: Adjacency Matrix.

Code:

```
shortestPathByDistanceTimeDF =pd.DataFrame( index=G.nodes(),
columns=G.nodes())
for i in G.nodes():
```

```
    for j in G.nodes():
        if i==j:
            continue
        lst=[]
        # try:
        #     lst.append(nx.shortest_path_length(G, source=i, target=j,
weight="distance"))
        # except nx.NetworkXNoPath:
        #     lst.append(None)
        try:
            lst.append(nx.shortest_path_length(G, source=i, target=j,
weight="l"))
        except nx.NetworkXNoPath:
            lst.append(None)

        shortestPathByDistanceTimeDF.loc[i,j] = lst
shortestPathByDistanceTimeDF
```

	IAD	IND	ISP	JAN	JAX	LAS	LAX	LBB	LIT	MAF	...	WYS	BJI	INL	GST	PUB	OTH	LMT	ITH	HTS
IAD	NaN	[111.0]	[113.0]	[128.0]	[97.0]	[260.0]	[236.0]	[202.0]	[155.0]	[205.0]	...	[307.0]	[190.0]	[198.0]	[437.0]	[None]	[322.0]	[332.0]	[141.0]	[110.0]
IND	[98.0]	NaN	[118.0]	[114.0]	[90.0]	[212.0]	[207.0]	[161.0]	[107.0]	[164.0]	...	[255.0]	[140.0]	[148.0]	[411.0]	[None]	[295.0]	[305.0]	[121.0]	[62.0]
ISP	[129.0]	[136.0]	NaN	[169.0]	[146.0]	[298.0]	[301.0]	[259.0]	[178.0]	[262.0]	...	[358.0]	[233.0]	[241.0]	[492.0]	[None]	[376.0]	[386.0]	[184.0]	[166.0]
JAN	[139.0]	[114.0]	[153.0]	NaN	[106.0]	[187.0]	[163.0]	[121.0]	[83.0]	[124.0]	...	[273.0]	[178.0]	[186.0]	[429.0]	[None]	[293.0]	[277.0]	[180.0]	[126.0]
JAX	[88.0]	[104.0]	[132.0]	[94.0]	NaN	[227.0]	[203.0]	[184.0]	[122.0]	[186.0]	...	[324.0]	[198.0]	[206.0]	[469.0]	[None]	[333.0]	[317.0]	[183.0]	[118.0]
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
OTH	[305.0]	[273.0]	[343.0]	[293.0]	[296.0]	[144.0]	[149.0]	[224.0]	[267.0]	[229.0]	...	[192.0]	[260.0]	[268.0]	[234.0]	[None]	NaN	[96.0]	[337.0]	[303.0]
LMT	[309.0]	[277.0]	[347.0]	[297.0]	[300.0]	[144.0]	[137.0]	[228.0]	[271.0]	[233.0]	...	[196.0]	[264.0]	[272.0]	[238.0]	[None]	[90.0]	NaN	[341.0]	[307.0]
ITH	[138.0]	[128.0]	[189.0]	[189.0]	[191.0]	[305.0]	[311.0]	[258.0]	[184.0]	[261.0]	...	[344.0]	[214.0]	[222.0]	[483.0]	[None]	[369.0]	[379.0]	NaN	[158.0]
HTS	[98.0]	[63.0]	[118.0]	[120.0]	[111.0]	[250.0]	[239.0]	[187.0]	[107.0]	[190.0]	...	[290.0]	[142.0]	[150.0]	[431.0]	[None]	[317.0]	[327.0]	[138.0]	NaN
PIR	[197.0]	[153.0]	[240.0]	[214.0]	[199.0]	[239.0]	[260.0]	[237.0]	[178.0]	[240.0]	...	[276.0]	[123.0]	[131.0]	[412.0]	[None]	[298.0]	[308.0]	[221.0]	[193.0]

303 rows × 303 columns

Fig 6.1.12: Shortest Path.

Code:

```
#CSV To Dataframe
import pandas as pd
station_code = pd.read_csv('US-Airport-Codes.csv',
low_memory=False)
station_code
import csv
airport_name={}
with open ('US-Airport-Codes.csv', mode='r') as f:
    data = csv.reader(f)
    airport_name={rows[1].strip():rows[0].strip() for rows in data}
print(type(airport_name))
```

	Airport	Code	State	State Name	Code.1
0	Birmingham International Airport	BHM	Alabama	Alabama	AL
1	Dothan Regional Airport	DHM	Alabama	Alaska	AK
2	Huntsville International Airport	HSV	Alabama	Arizona	AZ
3	Mobile	MOB	Alabama	Arkansas	AR
4	Montgomery	MGM	Alabama	California	CA
...	...	...	...	...	...
181	Milwaukee	MKE	Wisconsin	NaN	NaN
182	Casper	CPR	Wyoming	NaN	NaN
183	Cheyenne	CYS	Wyoming	NaN	NaN
184	Jackson Hole	JAC	Wyoming	NaN	NaN
185	Rock Springs	RKS	Wyoming	NaN	NaN

186 rows × 5 columns

Fig 6.1.13: Airport names and their Codes.

Code:

```
import csv
airport_name={}
with open ('US-Airport-Codes.csv', mode='r') as f:
   data = csv.reader(f)
   state_name={rows[1].strip():rows[2].strip() for rows in data}
print(state_name)
state_name.values()
temp={}
for z in state_name:
   if state_name[z]=='Alabama':
      temp[z]=G.degree(z)
temp
#Store Unique States
uniqueValues = set(state_name.values())
type(uniqueValues)
type(list(uniqueValues))
temp={}
for z in state_name:
   if state_name[z]=='Alabama':
      temp[z]=nod.degree(z)
temp
st.write(temp)
# Put Dictionary in Dataframe
df = pd.DataFrame.from_dict(temp, orient='index',
columns=['Degree'])
```

```
# Print table
st.write(df)
import csv
airport_name={}
with open ('US-STATE.csv', mode='r') as f:
   data = csv.reader(f)
   state_name2={rows[1].strip():rows[0].strip() for rows in data}
print(state_name2)
list(state_name2.values())
```

```
Output exceeds the size limit. Open the full output data in a text editor
['State Name',
 'Alabama',
 'Alaska',
 'Arizona',
 'Arkansas',
 'California',
 'Colorado',
 'Connecticut',
 'District of Columbia',
 'Florida',
 'Georgia',
 'Hawaii',
 'Idaho',
 'Illinois',
 'Indiana',
 'Iowa',
 'Kansas',
 'Kentucky',
 'Louisiana',
 'Maine',
 'Maryland',
 'Massachusetts',
 'Michigan',
 'Minnesota',
 'Mississippi',
```

Fig 6.1.14: States of USA.

Code:

```
state_name2
```

```
Output exceeds the size limit. Open the full output data in a text editor
{'Code': 'State Name',
 'AL': 'Alabama',
 'AK': 'Alaska',
 'AZ': 'Arizona',
 'AR': 'Arkansas',
 'CA': 'California',
 'CO': 'Colorado',
 'CT': 'Connecticut',
 'DC': 'District of Columbia',
 'FL': 'Florida',
 'GA': 'Georgia',
 'HI': 'Hawaii',
 'ID': 'Idaho',
 'IL': 'Illinois',
 'IN': 'Indiana',
 'IA': 'Iowa',
 'KS': 'Kansas',
 'KY': 'Kentucky',
 'LA': 'Louisiana',
 'ME': 'Maine',
 'MD': 'Maryland',
 'MA': 'Massachusetts',
 'MI': 'Michigan',
 'MN': 'Minnesota',
 'MS': 'Mississippi',
...
 'VA': 'Virginia',
```

Fig 6.1.15: States of USA and short forms.

6.2 VISUALISATION

Code:

```
import pickle
import streamlit as st
import pandas as pd
import numpy as np
import networkx as nx
page_bg_img = '''
<style>
[data-testid="stAppViewContainer"] {
    background-image:
url('https://www.devdiscourse.com/remote.axd?https://devdiscours
e.blob.core.windows.net/devnews/17_08_2022_14_11_33_798511
```

```
5.jpg?width=920&format=webp');
    background-size: cover;
}
[data-testid="stSidebar"] {
    background-color: rgba(0,0,0,0.0);
}
h1{
    color: indianred;
}
table{
    background-color: #00425A;
    text-align: center;
}

</style>
'''
st.markdown(page_bg_img, unsafe_allow_html=True)
st.title("Airlines Route Optimization System")
with open('airport.pkl', 'rb') as c:
    airport = pickle.load(c)
with open('graph.pkl', 'rb') as c:
    nod = pickle.load(c)
with open('airport_code.pkl', 'rb') as f:
    airportname = pickle.load(f)
with open('state_code.pkl', 'rb') as f:
    statename = pickle.load(f)
with open('nearairport.pkl', 'rb') as f:
    nearairport = pickle.load(f)

# print no of Airports and Flights
st.header("Number Of Airports: ")
```

st.write(nod.number_of_nodes())

st.header("Number Of Flights: ")

st.write(nod.number_of_edges())

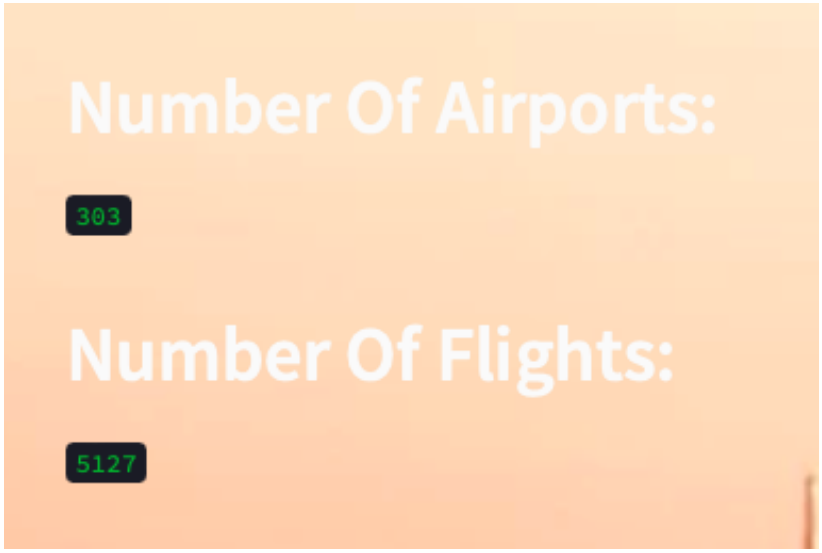

Fig 6.2.1: Number of Airports and Flights.

Code:

```
# listing busiest Airports
st.header("Busiest Airports ")
air1 = st.slider('Enter The No Of Airports Required', 0, 130, 25)
if st.button('Find Stations'):
    l = list(nod.degree(list(nod.nodes())))
    l.sort(key=lambda x: x[1], reverse=True)
    # top 10 nodes with highest degree (Airports with most number
of Flights)
    busy=l[:air1]
    bstation = pd.DataFrame(busy, columns=['Airports', 'No Of
Flights'])
    st.table(bstation)
```

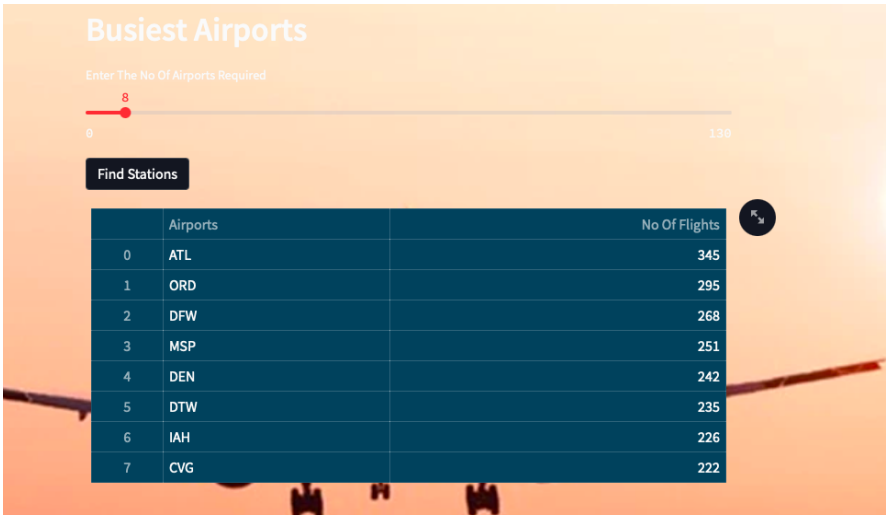

Fig 6.2.2: Busiest Airports.

Code:

```
# Centrality Measure Calculation
G2 = nx.Graph(nod)
st.header("Centrality Measure Calculation")
centrality = st.selectbox('Select Centrality Type', ('Degree
Centrality',
                    'Closeness Centrality', 'Betweenness Centrality',
'Eigenvector Centrality'))
if st.button('Calculate'):
    if centrality == 'Degree Centrality':
        st.write("Degree Centrality: ", nx.degree_centrality(nod))
    elif centrality == 'Closeness Centrality':
        st.write("Closeness Centrality: ",
nx.closeness_centrality(nod))
    elif centrality == 'Betweenness Centrality':
        st.write("Betweenness Centrality: ",
nx.betweenness_centrality(nod))
    elif centrality == 'Eigenvector Centrality':
        st.write("Eigenvector Centrality: ",
nx.eigenvector_centrality(G2))
```

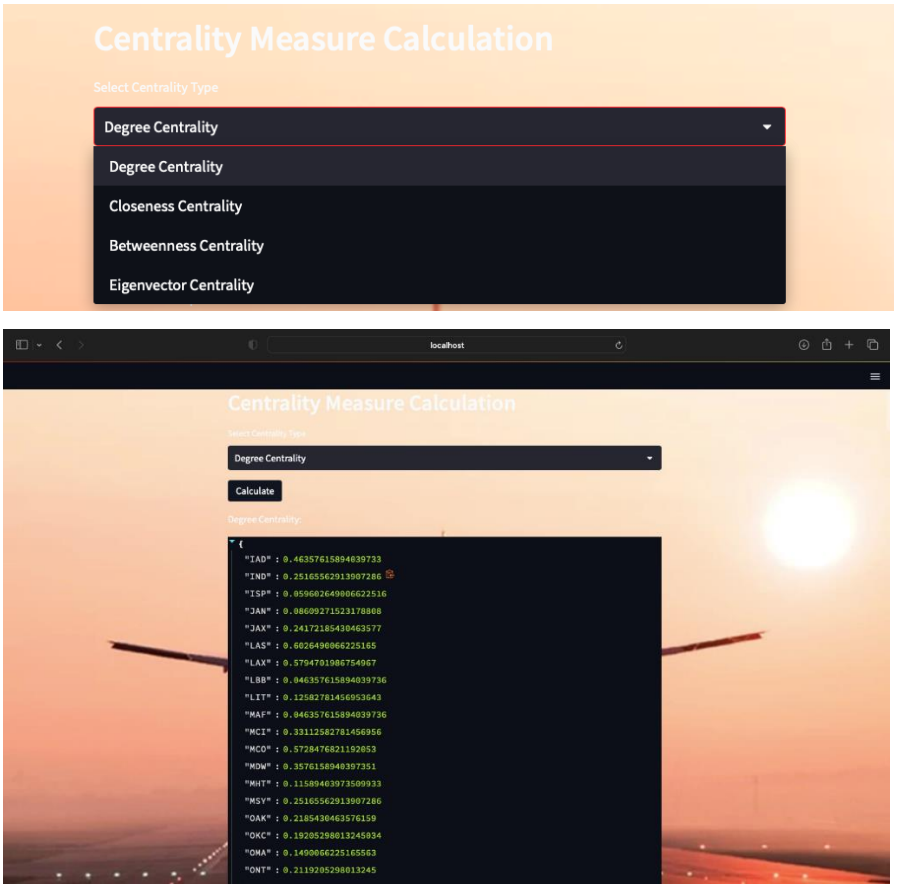

Fig 6.2.3: Centrality Measures.

Code:

```
# Shortest Path Calculation
st.header("Shortest Path Calculation For Flight Route")
option1 = st.selectbox('Select Source Airport', airportname.values())
option2 = st.selectbox('Select Destination Airport', airportname.values())

if st.button('Find Route'):
    # route()
    st.write("Route Found")
    st.write("Route is:")
    p = nx.shortest_path(nod,source=[i for i in airportname if airportname[i]==option1][0],target=[i for i in airportname if
```

```
airportname[i]==option2][0], weight="l")
    q = nx.shortest_path_length(nod,source=[i for i in airportname if
airportname[i]==option1][0],target=[i for i in airportname if
airportname[i]==option2][0], weight="l")
    # p = nx.shortest_path(nod,source=option1,target=option2,
weight="l")
    # q =
nx.shortest_path_length(nod,source=option1,target=option2,
weight="l")
    #Convert List To Table And Display
    temp=[]
    for z in p:
        temp.append(z)
    shrt_path = pd.DataFrame(temp)
    st.table(shrt_path)
    st.write('Distance travel is: ', q, 'km')
```

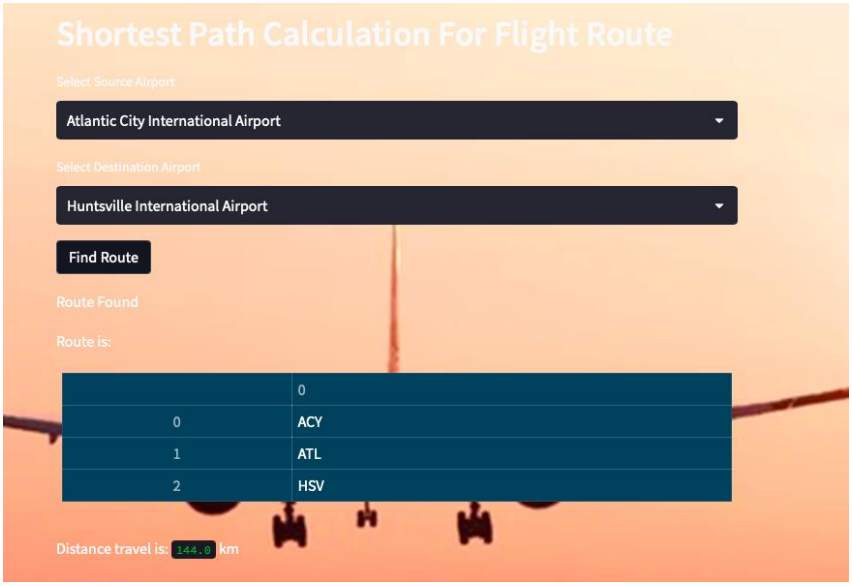

Fig 6.2.4: Shortest Path.

Code:

```
# Articulation Points Calculation
st.header("Articulation Points Calculation")
if st.button('Calculate Articulation Points'):
    st.write("Articulation Points: ", list(nx.articulation_points(G2)))
    st.write("Number of Articulation Points: ",
            len(list(nx.articulation_points(G2))))
```

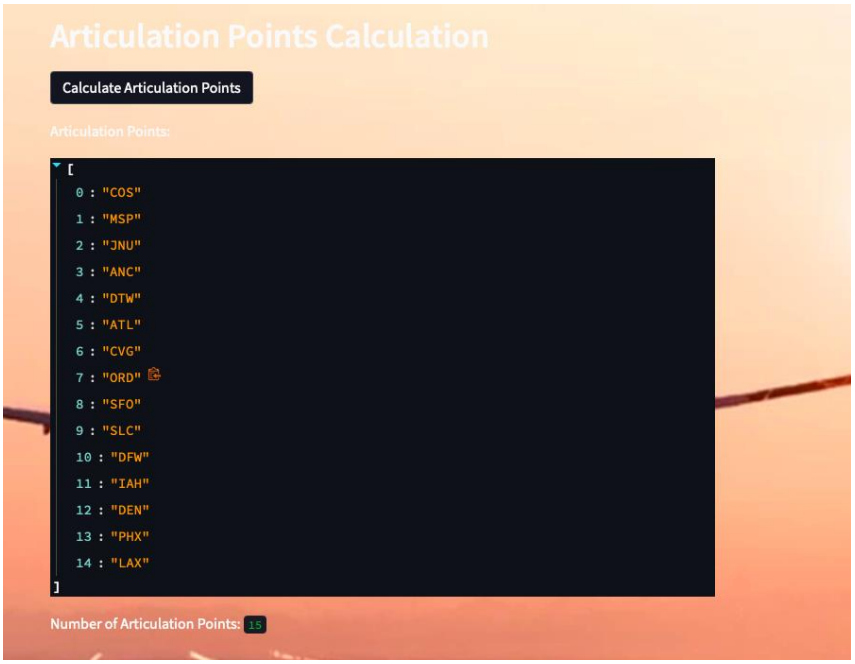

Fig 6.2.5: Articulation Points.

Code:

```
#Efficiency Calculation
st.header("Efficiency Calculation")
if st.button('Calculate Global Efficiency'):
    st.write("Efficiency: ", nx.global_efficiency(G2))
G0 = nx.Graph(nod)
su1 = st.selectbox('Select Source Airport:', airport)
des1 = st.selectbox('Select Destination1 Airport:', airport)
des2 = st.selectbox('Select Destination2 Airport:', airport)
```

```
if st.button('Calculate Efficiency'):
    st.write("Efficiency: ", nx.efficiency(G0, su1, des1))
    st.write("Efficiency: ", nx.efficiency(G0, su1, des2))
# efficiency = nx.efficiency(G0, "ATL", "YUM")
```

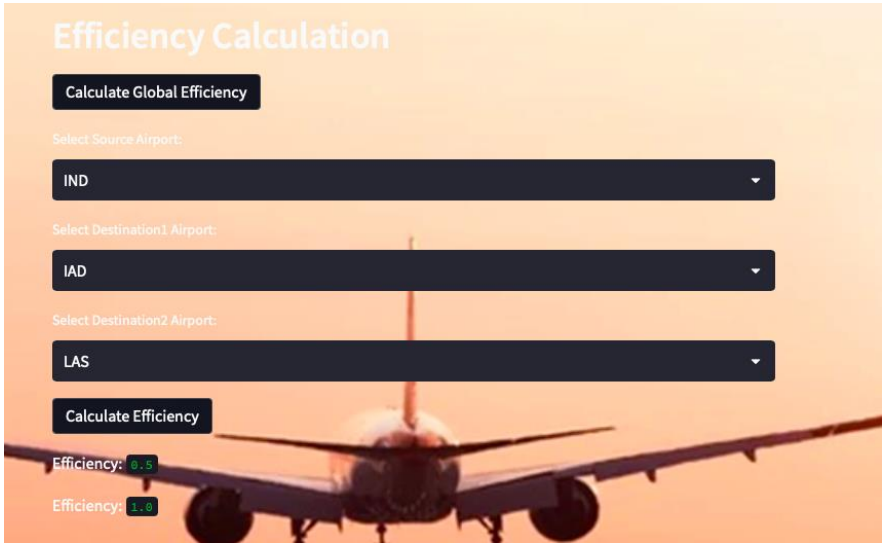

Fig 6.2.6: Efficiency.

Code:

```
#Stations In A State
st.header("Find Airports In A State")
option = st.selectbox('Select State',statename.values())
if st.button('Find Airports:'):
    # route()
    st.write("Airports Found")
    st.write("Airports are:")
    temp={}
    for z in statename:
        if statename[z]==option:
            temp[airportname[z]]=nod.degree(z)
    #Remove non numeric values from temp values
    temp = {k: v for k, v in temp.items() if isinstance(v, int)}
```

#Sort temp based on values

temp = {k: v for k, v in sorted(temp.items(), key=lambda item: item[1],reverse=True)}

st.write(temp)

state_airport = pd.DataFrame.from_dict(temp, orient='index', columns=['Airport','No Of Flights'])

st.table(state_airport)

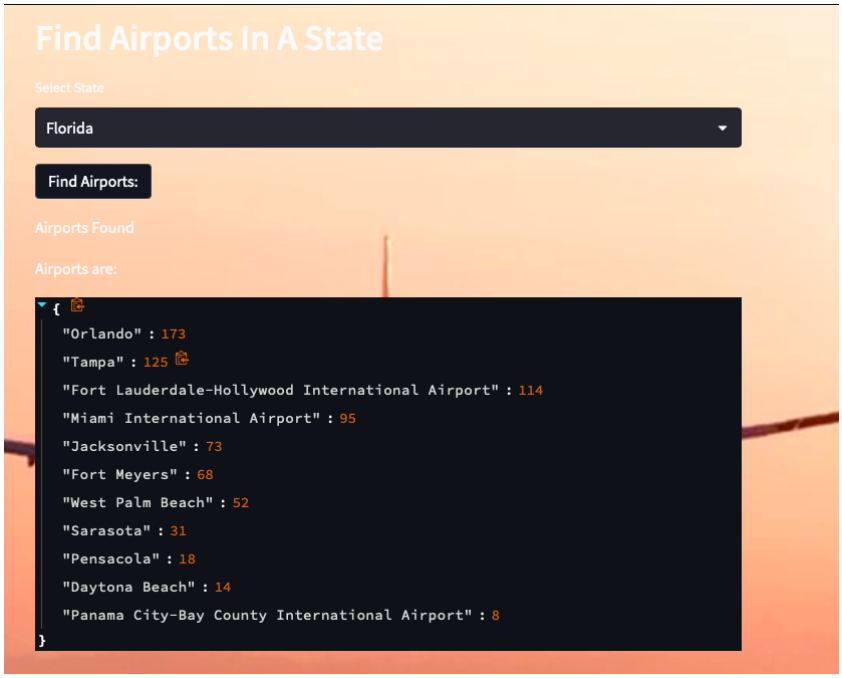

Fig 6.2.7: Airports in a State.

Code:

#Airport withing a distance from a particular airport

st.header("Find Airports Nearer From A Particular Airport")

option8 = st.selectbox('Select Airport',airportname.values())

dist = st.slider('Enter The No Of Airports', 0, 20, 5)

if st.button('Find Airports Nearer:'):

route()

st.write("Airports Found")

```
    st.write("Airports are:")
    n=nearairport[[i for i in airportname if airportname[i]==option8][0]].sort_values().head(dist)

    #convert n to dictionary
    n=n.to_dict()
    blank={}
    for i in n:
        blank[i]=int(n[i][0])
    state_airport = pd.DataFrame.from_dict(blank, orient='index', columns=['No Of Flights'])
    st.table(state_airport)
```

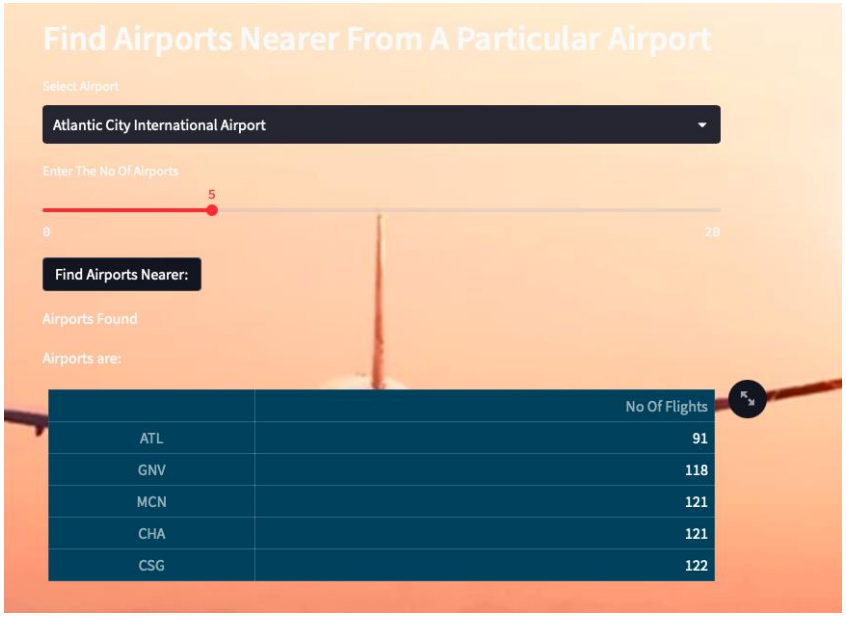

Fig 6.2.8: Airports nearer from a particular airport.

Chapter 7

CONCLUSION & FUTURE WORK

7.1 CONCLUSION & FUTURE WORK

The use of machine learning techniques in processing and analyzing the USA flight dataset has shown great potential for providing insights and making informed decisions in the aviation industry. Through data pre-processing, extraction, and processing, we were able to derive useful information such as unique airport names, centrality measures for each node, and busiest airports with the highest degree. The use of pickle files also proved to be a useful tool in storing and retrieving Python objects such as airport names, codes, graphs, and state information. One future work that could be explored is the identification of clusters among the states using the nearest airport feature. By doing this, airlines and government entities could better understand where new airports should be established and if the existing infrastructure is sufficient to handle the traffic. This could potentially lead to more efficient travel, reduced congestion, and improved airport services. Another future work that could be explored is the development of predictive models to forecast flight delays and cancellations. By analyzing historical data, machine learning models could be trained to identify patterns and predict when delays or cancellations are likely to occur. This information could be used by airlines and travelers to better plan and manage their schedules, leading to reduced disruptions and increased customer satisfaction. In conclusion, the use of machine learning techniques in processing and analyzing the USA flight dataset has provided valuable insights and potential applications in the aviation industry. The future work mentioned above and other potential avenues for exploration could lead to even greater benefits and advancements in the field.

Chapter 8

REFERENCES

8.1 REFERENCES

- Gabriella Schoier and Giuseppe Borusso (2014). A network-based analysis of the European air traffic network., International Conference on Computational Science and Its Applications.

- Massimiliano Zanin and Fabrizio Lillo (2013), A complex network analysis of air traffic., The European Physical Journal Special Topics 215(1).

- Oriol Lordon, Jose M Sallan and Pep Simo (2014), Study of the topology and robustness of airline route networks from the complex network approach., Journal of Transport Geography Vol 37, pp. 112.

- Chun-Hsiang Chan (2021), Spatially Characterizing Major Airline Alliances., International Journal of Geo-Information 10(1).

- Weisi Guo, Bogdan Toader, Roxana Feier, Guillem Mosquera, Fabian Ying, Se-Wook Oh, Matthew Price-Williams and Armin Krupp (2019), Global Air transport complex network : multi-scale analysis., SN Applied Sciences volume 1, Article number: 680.