**Project No: 3**

**Project Title: Transit Time Prediction**

**Prepared by: Rose Mary Jose**

**Date: 16-07-23**

Prepared for “Thoughtware Training Private Limited”, under Guidance of its CEO Mr. Pattabhi Raman. The project will be subject to further research, modification and exclusive use of “Thoughtware Training Private Limited”

**Objective of the project**

Network optimization refers to the process of enhancing the efficiency and performance of interconnected systems, such as transportation, communication, and supply chain networks. It involves the utilization of mathematical techniques and algorithms to strategically allocate resources, minimize costs, maximize throughput, and optimize routing within these networks. By leveraging optimization methods, organizations can achieve optimal utilization of resources, improved service quality, reduced operational expenses, and better decision-making. Network optimization encompasses a wide range of applications, including optimizing the flow of goods, managing data transmission, balancing workloads, and designing optimal transportation routes. Ultimately, network optimization aims to align network operations with organizational goals, resulting in enhanced overall system performance and competitiveness.

For the project, airline network optimization dataset is used. The objective is to find the shortest Path Problem: aims to find the shortest path between two nodes in an airline network, along with the time taken in air and distance travelled.

**Dataset**

Source: <https://www.kaggle.com/datasets/tanmoyie/airlines-network-optimization-cleaned>

For network optimization an airline network optimization dataset is considered from Kaggle. The considered dataset consists of the following columns:

year: year in which the data was collected

month: month of the flight departure

day: day of the flight departure

dep\_time: time at which the flight departed

sched\_dep\_time: time which was scheduled for the departure

dep\_delay: time difference between departure time and scheduled departure time

arr\_time: time at which the flight arrived at the destination

sched\_arr\_time: scheduled time for the arrival

arr\_delay: time difference between arrival time and the scheduled arrival time

carrier: the name of the carrier

flight: the number assigned for the flight

talinum: the number given to the tail of the plane

origin: place from which the plane starts its journey

destination: place to which the plane is going

air\_time: time taken by the plane in air

diatnace: time taken by the flight to reach destination.

From the columns it is clear that we need to find the optimal route for a plane to travel in order to reach the destination. Also, we calculate the air time taken and the distance travelled.

import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns  
import io

df = pd.read\_csv('Airlines\_timeprediction.csv')  
pd.set\_option('display.max\_rows',None)  
df

year month day dep\_time sched\_dep\_time dep\_delay arr\_time \  
0 2013 2 26 1807.0 1630 97.0 1956.0   
1 2013 8 17 1459.0 1445 14.0 1801.0   
2 2013 2 13 1812.0 1815 -3.0 2055.0   
3 2013 4 11 2122.0 2115 7.0 2339.0   
4 2013 8 5 1832.0 1835 -3.0 2145.0   
5 2013 6 30 1500.0 1505 -5.0 1751.0 sched\_arr\_time arr\_delay carrier flight tailnum origin dest air\_time \   
0 1837 79.0 EV 4411 N13566 EWR MEM 144.0   
1 1747 14.0 B6 1171 N661JB LGA FLL 147.0   
2 2125 -30.0 AS 7 N403AS EWR SEA 315.0   
3 2353 -14.0 B6 97 N656JB JFK DEN 221.0   
4 2155 -10.0 AA 269 N3EYAA JFK SEA 358.0   
5 1650 61.0 UA 685 N424UA LGA ORD 116.0   
 distance   
0 946   
1 1076   
2 2402   
3 1626   
4 2422   
5 733

**EDA**

The shape of a dataset, accessed using the ‘shape’ attribute in pandas or NumPy, provides insights into the structure of the data in terms of rows and columns. It's a fundamental piece of information that helps you understand the dimensions and size of your dataset.

The dataset considered consist of **100 rows and 16 columns**.

The dataset's info returns details about the data frame. It contains information on the total number of columns, column labels, datatypes, the number of columns that aren't null, memory utilization, and range index. The information makes it obvious that the dataset under consideration consists of 4 features of object dtype, 5 features of int64, and 7 features of float64. Missing value’s presence can be detected from the non-null count.

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 100 entries, 0 to 99  
Data columns (total 16 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 year 100 non-null int64   
 1 month 100 non-null int64   
 2 day 100 non-null int64   
 3 dep\_time 99 non-null float64  
 4 sched\_dep\_time 100 non-null int64   
 5 dep\_delay 99 non-null float64  
 6 arr\_time 99 non-null float64  
 7 sched\_arr\_time 100 non-null int64   
 8 arr\_delay 99 non-null float64  
 9 carrier 100 non-null object   
 10 flight 100 non-null int64   
 11 tailnum 100 non-null object   
 12 origin 100 non-null object   
 13 dest 100 non-null object   
 14 air\_time 99 non-null float64  
 15 distance 100 non-null int64   
dtypes: float64(5), int64(7), object(4)  
memory usage: 12.6+ KB

To obtain the dataset's summary statistics, describe function is used. The Describe function of the numerical column provides an overview of the central tendency, dispersion, and distributional form of the dataset. The 25th, 50th, and 75th percentile values are returned, together with the count, mean, standard deviation, minimum value, and maximum value. From the descriptive statistics it is observable that the data considered is of the year 2013. Missing values can be detected from the count of the descriptive statistics. By comparing the mean and median it can be concluded that the columns dep\_delay, arr\_delay, flight, air\_time, distance follows a right-skewed distribution since their mean is greater than median. Features such as month and day follows a normal distribution. Rest columns follows left-skewed distribution.

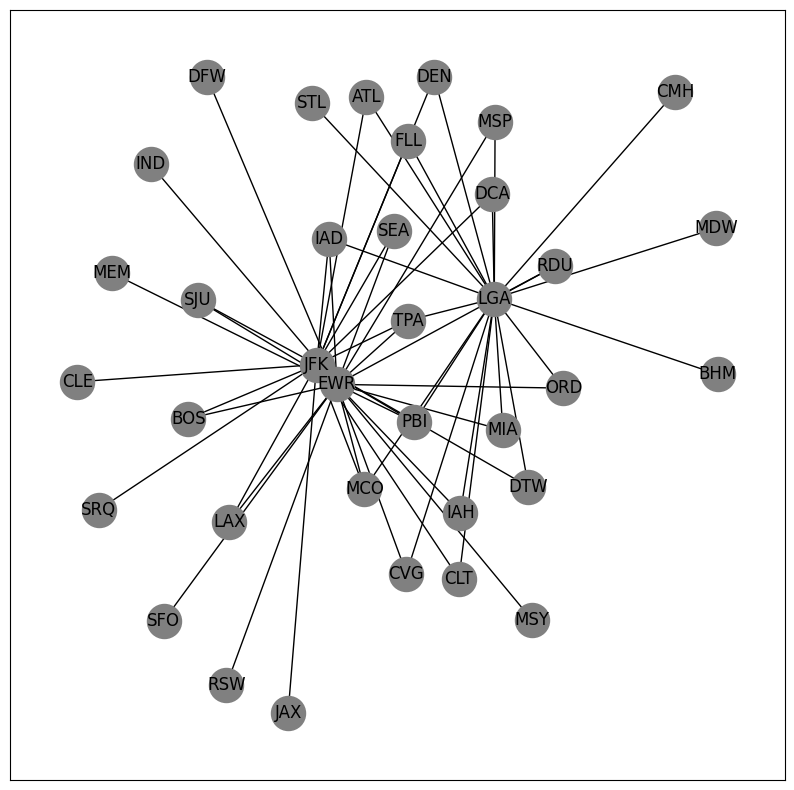
count mean std min 25% 50%  
year 100.0 2013.000000 0.000000 2013.0 2013.00 2013.0   
month 100.0 5.770000 3.475324 1.0 2.00 5.0   
day 100.0 15.950000 8.848186 1.0 10.00 16.0   
dep\_time 99.0 1333.828283 520.292991 511.0 824.50 1354.0   
sched\_dep\_time 100.0 1317.710000 501.604200 515.0 822.75 1350.0   
dep\_delay 99.0 8.262626 37.000437 -11.0 -5.00 -3.0   
arr\_time 99.0 1499.141414 561.939807 9.0 1035.50 1556.0   
sched\_arr\_time 100.0 1540.820000 529.796023 43.0 1043.00 1613.5   
arr\_delay 99.0 1.111111 38.698175 -53.0 -19.50 -9.0   
flight 100.0 1943.980000 1614.206544 7.0 526.75 1573.5   
air\_time 99.0 151.181818 79.341083 39.0 101.00 136.0   
distance 100.0 1037.010000 618.106468 187.0 630.00 973.5   
  
 75% max   
year 2013.00 2013.0   
month 9.00 12.0   
day 23.00 31.0   
dep\_time 1816.50 2311.0   
sched\_dep\_time 1811.25 2155.0   
dep\_delay 2.00 235.0   
arr\_time 2033.00 2359.0   
sched\_arr\_time 2041.50 2359.0   
arr\_delay 5.00 221.0   
flight 3379.75 6101.0   
air\_time 183.00 358.0   
distance 1113.75 2565.0

From the summary statistics of the categorical columns count, number of unique values, the mode of the categorical data and its frequency is given. It provides useful information for understanding the distribution and characteristics of categorical data. From the summary statistics it is clear the dataset consists of flight details that originate from 33 air ports in USA and there are 3 destination that appears in the dataset.

count unique top freq  
carrier 100 11 UA 19  
tailnum 100 99 N656JB 2  
origin 100 3 LGA 42  
dest 100 33 LAX 7

**Data Visualization**

For visualizing the routes of transport of the flights, the NetworkX library from python is used. The NetworkX library is used for analyzing and visualizing networks and graphs. Here creates a graph (network) from a pandas DataFrame using the columns 'origin' and 'dest' as source and target nodes for edges. This plot can provide insights into the structure, connectivity, and centrality of the network. The plot displays nodes (vertices) and edges (connections between nodes). It can be used to observe the distribution of nodes and how they are interconnected by edges. Labels are associated with each node. Nodes with higher degree centrality (more connections) might be more influential or central within the network. Nodes with high betweenness centrality could indicate critical connectors between different parts of the network. The overall density of the network indicates how many potential edges are present compared to the maximum possible edges.

****

**Data Preparation**

**Detecting and handling missing values**: To ensure accuracy, reliability, and robustness of analyses and models, data preparation is essential for detecting and handling missing values and outliers. Due to the fact that missing values result from errors or inadequate recording and outliers represent anomalies or exceptional circumstances, these problems may result in biased conclusions and incorrect predictions. For the sake of maintaining data integrity and arriving with suitable conclusion, these issues must be regularly addressed.

For detecting the missing data is to use Python functions like isnull() and sum(). The isnull().sum() function helps to quickly figure out the amount of data missing from each column. In this step, one can observe which columns have missing values and deal with them. In the dataset considered there is presence of missing value and the missing value is present in one row which is given below. Here missing value is handled by dropping it using dropna function which makes the shape of the dataset as (99,16)

df.isnull().sum()

year 0  
month 0  
day 0  
dep\_time 1  
sched\_dep\_time 0  
dep\_delay 1  
arr\_time 1  
sched\_arr\_time 0  
arr\_delay 1  
carrier 0  
flight 0  
tailnum 0  
origin 0  
dest 0  
air\_time 1  
distance 0  
dtype: int64

df = df.dropna(axis = 0,how= 'any')  
df.shape

(99, 16)

**Detecting and handling duplicated values:** Duplicated values are identical or repeated entries in a dataset, resulting from errors, system glitches, or unintentional repetitions. Identifying and handling duplicates is crucial in data preprocessing to maintain integrity and ensure accurate analyses. Functions like duplicated() in Python can detect duplicates and mark subsequent occurrences as duplicates. Here in the considered dataset no duplicated values are present.

df.duplicated().sum()

0

**Model Selection**

Model selection is the process of choosing the most appropriate mathematical or computational representation (model) to solve a specific problem or analyze a dataset. It involves identifying a model that accurately captures the underlying relationships, patterns, or behaviors of the data, leading to effective decision-making or problem-solving.

The objective of this problem is to find the shortest path, it is a network optimization problem, determining the approach and tools used to solve complex interconnected problems. The shortest path problem is a fundamental **optimization problem** in graph theory. It involves finding the shortest path between two nodes in a graph, where the edges have associated weights or costs. The objective is to minimize the total weight or cost of the path while moving from the source node to the destination node. This problem is widely used in various applications, including navigation systems, transportation planning, network routing, and more.

**Model building**

Components of the Model

1) Graph Representation: The graph is represented as a collection of nodes (vertices) and edges (links) connecting these nodes. The nodes represent the origin and destination places.

2) Source and Destination Nodes: The problem requires specifying the origin city (source) and the destination city (destination) between which the shortest path needs to be found. The starting city and the destination city are inputted for the optimization process.

3) Objective Function: The objective is to find the path that minimizes the total distance of traversing the edges along the path.

4) Decision Variables: The decision variable is the selection of edges to form the path from the source to the destination.

5) Constraints: There are no specific constraints in the shortest path problem. However, the selected path must be a valid path in the graph, connecting the source and destination nodes.

Solving the Model

The NetworkX library in Python is used to calculate the shortest path, total air time, and total distance between two cities in a flight network.

**Implementation and Usage**

In Python, the NetworkX library provides built-in functions to solve the shortest path problem. The nx.shortest\_path() function from the NetworkX library is used to calculate the shortest path between the origin and destination cities in a flight network represented by the graph FG. The weight parameter 'air\_time' indicates that the function should consider the air travel time as the weight to find the shortest path. The nx.shortest\_path\_length() function is used to calculates the total air time (total\_air\_time) required for the flight by considering the shortest path calculated and calculates the total distance (total\_distance) traveled for the flight by considering the shortest path and using the 'distance' weight.

origin = input('enter the starting city: ')  
destination = input('enter the destination city: ')  
shortest\_path = nx.shortest\_path(FG, source=origin, target=destination, weight='air\_time')  
total\_air\_time = nx.shortest\_path\_length(FG, source=origin, target=destination, weight='air\_time')  
total\_distance = nx.shortest\_path\_length(FG, source=origin, target=destination, weight='distance')

The output of the optimization problem is found by inputting the starting and destination city.

enter the starting city: BHM  
enter the destination city: EWR  
Shortest path: ['BHM', 'LGA', 'IAD', 'EWR']  
Total air time taken for the flight: 207.0  
Total distance traveled: 1307

**Model’s Summary**

The airline network optimization model employed the NetworkX library to analyze and improve the efficiency of travel routes. The primary goal was to determine the shortest path between two cities while considering airtime and distance. The NetworkX library facilitated the creation of a graph where cities represented nodes and flight routes were edges. The 'shortest\_path' function was used to find the optimal route, and additional functions helped calculate the total airtime and distance. This approach allowed for accurate route planning, contributing to enhanced flight operations and customer experience by minimizing travel time and distance. Ultimately, this model significantly contributed to improving overall operational efficiency and passenger satisfaction by providing airlines with a tool to optimize their flight paths, reduce travel times, and conserve resources.

**Conclusion**

The shortest path problem involves finding the most efficient path between two nodes in a graph, considering the weights or costs associated with the edges. It's a classic optimization problem with wide-ranging applications across various domains. The NetworkX library has proven to be a robust solution, modelling flight routes as graphs and considering airtime and distance. This approach enhances operational efficiency, reduces fuel consumption, lowered emissions, and improves passenger experience. The successful integration of NetworkX demonstrates the power of data-driven decision-making, allowing airlines to strategically plan routes and make informed choices that positively impact their bottom line and sustainability efforts. This optimization model demonstrates the benefits of advanced analytics in real-world challenges.

**References**

<https://www.kaggle.com/code/tanmoyie/network-optimization-example-airlines/notebook>