**PROJECT DEVELOPMENT PHASE**

# Sprint - II

|  |  |
| --- | --- |
| **Date** | 05-Nov-2022 |
| **Team ID** | PNT2022TMID12521 |
| **Project Name** | Developing a Flight Delay Model Using Machine Learning |
| **Maximum Marks** | 8 Marks |

Data Pre-processing

# Importing Libraries

import numpy as np import pandas as pd import seaborn as sns

import matplotlib.pyplot as plt import matplotlib as mpl

import matplotlib.patches as patches

from matplotlib.patches import ConnectionPatch from collections import OrderedDict

from matplotlib.gridspec import GridSpec

%matplotlib inline

pd.set\_option('display.max\_columns', None) pd.set\_option('display.max\_rows', None)

import os os.getcwd()

os.chdir("C:/Users/administrator.DOMAIN-01/Documents/GitHub/Abc")

# Importing the necessary files

df = pd.read\_csv("Data/flight\_data.csv") planes = pd.read\_csv("Data/planes.csv") airports = pd.read\_csv("Data/airports.csv") carriers = pd.read\_csv("Data/carriers.csv")

df.head(15)

# Checking the dimensions of the 'flight\_data' dataset

df.shape

# Checking whether the dataset contains the NULL values or not

df.isnull().sum()

# Dropping the rows

df = df.dropna() df.head(10) df.tail(10)

# Dimension after dropping the rows containing NULL values

df.shape

# Now again checking whether the dataset till contains any NULL values

df.isnull().sum()

# Before type casting of 'dep\_time', 'dep\_delay', 'arr\_time', 'arr\_delay'

df.info()

# Type casting

df['dep\_time'] = df['dep\_time'].astype('int64') df['dep\_delay'] = df['dep\_delay'].astype('int64') df['arr\_time'] = df['arr\_time'].astype('int64') df['arr\_delay'] = df['arr\_delay'].astype('int64')

# After type casting of 'dep\_time', 'dep\_delay', 'arr\_time', 'arr\_delay'

df.info() df.head(10)

**Exploratory Data Analysis**

plt.figure(figsize = (18, 6)) sns.countplot(df['month']) plt.title('Month Distribution', size = 25) plt.xticks(size = 15)

plt.yticks(size = 15) plt.xlabel("Months", size = 20) plt.ylabel("Frequency", size = 20) plt.show()

# Market share of each Airline(carrier)

plt.figure(figsize = (20, 6)) sns.countplot(df['carrier']) plt.title('Various Carriers in US') plt.xticks(size = 15)

plt.yticks(size = 15) plt.xlabel("Carriers", size = 20) plt.ylabel("Frequency", size = 20) plt.show()

df['carrier'].value\_counts().to\_frame()

# Extracting statistical parameters from a group by object:

def get\_stats(group):

return {'min': group.min(), 'max': group.max(),

'count': group.count(), 'mean': group.mean()}

# Creation of a dataframe with statitical infos on each airline:

global\_stats = df['dep\_delay'].groupby(df['carrier']).apply(get\_stats).unstack() global\_stats = global\_stats.sort\_values('count')

global\_stats

# Graphs on flights, airports & delays

global\_stats1 = global\_stats global\_stats = global\_stats1.head(14) codes = global\_stats.index.tolist()

carriers1 = carriers[carriers['IATA\_CODE'].isin(codes)]

abbr\_companies = carriers1.set\_index('IATA\_CODE')['AIRLINE'].to\_dict()

font = {'family' : 'DejaVu Sans', 'weight' : 'bold', 'size' : 15} mpl.rc('font', \*\*font)

import matplotlib.patches as mpatches

# Extraction of a subset of columns and redefine the airlines labelling

df2 = df.loc[:, ['carrier', 'dep\_delay']]

df2['carrier'] = df2['carrier'].replace(abbr\_companies)

colors = ['royalblue', 'grey', 'wheat', 'c', 'firebrick', 'seagreen', 'lightskyblue', 'lightcoral', 'yellowgreen', 'gold', 'tomato', 'violet', 'aquamarine', 'chartreuse']

fig = plt.figure(1, figsize=(22,17)) gs=GridSpec(2,2) ax1=fig.add\_subplot(gs[0,0]) ax2=fig.add\_subplot(gs[0,1]) ax3=fig.add\_subplot(gs[1,:])

# Pie chart nº1: nb of flights

labels = [s for s in global\_stats.index] sizes = global\_stats['count'].values

explode = [0.3 if sizes[i] < 20000 else 0.0 for i in range(len(abbr\_companies))] patches, texts, autotexts = ax1.pie(sizes, explode = explode,

labels=labels, colors = colors, autopct='%1.0f%%', shadow=False, startangle=0)

for i in range(len(abbr\_companies)): texts[i].set\_fontsize(14)

ax1.axis('equal')

ax1.set\_title('% of flights per company', bbox={'facecolor':'midnightblue', 'pad':5}, color = 'w',fontsize=18

# Setting the legend: abreviation -> airline name

comp\_handler = []

for i in range(len(abbr\_companies)): comp\_handler.append(mpatches.Patch(color=colors[i],

label = global\_stats.index[i] + ': ' + abbr\_companies[global\_stats.index[i]])) ax1.legend(handles=comp\_handler, bbox\_to\_anchor=(0.2, 0.9),

fontsize = 13, bbox\_transform=plt.gcf().transFigure)

# Pie chart nº2: mean delay at departure

sizes = global\_stats['mean'].values sizes = [max(s,0) for s in sizes]

explode = [0.0 if sizes[i] < 20000 else 0.01 for i in range(len(abbr\_companies))] patches, texts, autotexts = ax2.pie(sizes, explode = explode, labels = labels,

colors = colors, shadow=False, startangle=0,

autopct = lambda p : '{:.0f}'.format(p \* sum(sizes) / 100)) for i in range(len(abbr\_companies)):

texts[i].set\_fontsize(14) ax2.axis('equal')

ax2.set\_title('Mean delay at origin', bbox={'facecolor':'midnightblue', 'pad':5}, color='w', fontsize=18)

# Redefine the colors for correspondance with the pie charts

codes = global\_stats1.index.tolist()

carriers1 = carriers[carriers['IATA\_CODE'].isin(codes)]

abbr\_companies = carriers1.set\_index('IATA\_CODE')['AIRLINE'].to\_dict()

colors = ['firebrick', 'gold', 'lightcoral', 'aquamarine', 'c', 'yellowgreen', 'grey', 'seagreen', 'tomato', 'violet', 'wheat', 'chartreuse', 'lightskyblue', 'royalblue', 'black', 'grey', 'white', 'silver', 'black', 'pink']

ax3 = sns.stripplot(y="carrier", x="dep\_delay", size = 4, palette = colors, data=df2, linewidth = 0.5, jitter=True)

plt.setp(ax3.get\_xticklabels(), fontsize=14) plt.setp(ax3.get\_yticklabels(), fontsize=14)

ax3.set\_xticklabels(['{:2.0f}h{:2.0f}m'.format(\*[int(y) for y in divmod(x,60)]) for x in ax3.get\_xticks()])

plt.xlabel('Departure delay', fontsize=18, bbox={'facecolor':'midnightblue', 'pad':5}, color='w', labelpad=20)

ax3.yaxis.label.set\_visible(False)

plt.tight\_layout(w\_pad=3)

# Plot Mean Delay of various Airline(carrier)

carrier\_code=carriers.set\_index('IATA\_CODE')['AIRLINE'].to\_dict() mpl.rc('patch', edgecolor = 'dimgray', linewidth = 1) mpl.rcParams.update(mpl.rcParamsDefault) mpl.rcParams['hatch.linewidth'] = 2.0

fig = plt.figure(1, figsize = (11, 6))

ax = sns.barplot(x = 'dep\_delay', y = 'carrier', data = df, color = 'lightskyblue', ci = None)

ax = sns.barplot(x = 'arr\_delay', y = 'carrier', data = df, color = 'r', hatch = '///', alpha = 0.0, ci

= None)

labels = [carrier\_code[item.get\_text()] for item in ax.get\_yticklabels()] ax.set\_yticklabels(labels)

ax.yaxis.label.set\_visible(False)

plt.xlabel("Mean delay [min] (@departure: blue, @arrival: hatch lines)", fontsize = 15, weight = 'bold', labelpad = 10)

mpl.rc('patch', edgecolor = 'dimgray', linewidth = 1) mpl.rcParams.update(mpl.rcParamsDefault) mpl.rcParams['hatch.linewidth'] = 2.0

fig = plt.figure(1, figsize = (10, 6))

# Subset 4 major airlines

ax = sns.barplot(x = 'dep\_delay', y = 'carrier', data = df, order = ['AA', 'DL', 'F9', 'HA', 'B6'], color = 'lightskyblue', ci = None)

ax = sns.barplot(x = 'arr\_delay', y = 'carrier', data = df, order = ['AA', 'DL', 'F9', 'HA', 'B6'], color = 'r', hatch = '///', alpha = 0.0, ci = None)

labels = [carrier\_code[item.get\_text()] for item in ax.get\_yticklabels()]

ax.set\_yticklabels(labels) ax.yaxis.label.set\_visible(False)

plt.xlabel("5 Major Carrier's Mean Delay [min] (@departure: blue, @arrival: hatch lines)", fontsize = 12, weight = 'bold', labelpad = 10)

# Plotting the Market Share of the Airports(origin) of New York

df['origin'].value\_counts().to\_frame() plt.pie(

df['origin'].value\_counts(),

labels = df['origin'].value\_counts().index, explode = (0.1, 0, 0),

startangle = 90, autopct = '%1.1f%%',

colors = ['#52D017', '#F62217', '#43C6DB']

)

plt.tight\_layout()

plt.title("New York City Airport Market share") plt.show()

fig = plt.figure(1, figsize = (12, 6))

df[df['origin'] == 'EWR']['month'].value\_counts().sort\_index().plot(kind = 'line', color = '#52D017')

df[df['origin'] == 'JFK']['month'].value\_counts().sort\_index().plot(kind = 'line', color = '#F62217')

df[df['origin'] == 'LGA']['month'].value\_counts().sort\_index().plot(kind = 'line', color = '#43C6DB')

plt.title("Flights in New York City Area", size = 15) plt.xticks(range(1, 13), size = 12)

plt.yticks(size = 12) plt.xlabel("Month", size = 17) plt.ylabel("Frequency", size = 17) plt.legend(['EWR', 'JFK', 'LGA'])

# Modelling

def map\_labels(delays): if delays > 15:

return 1 else:

return 0

df['delayed'] = ((df['dep\_delay'].map(map\_labels) + df['arr\_delay'].map(map\_labels)) != 0).astype(int)

df['delayed'].value\_counts(normalize = True)

df.head(20)

df.tail(5)

# Feature Omission

columns\_to\_remove = ['dep\_time', 'sched\_dep\_time', 'dep\_delay', 'arr\_time', 'sched\_arr\_time', 'arr\_delay', 'flight', 'tailnum', 'air\_time', 'distance', 'hour', 'minute', 'time\_hour'] df.drop(columns\_to\_remove, axis = 1, inplace = True)

df.head() df['delayed'].value\_counts().to\_frame() df['dest'].value\_counts().to\_frame()

df\_filtered = df[df['dest'].isin(["LEX","TVC","MYR","CHO","BZN","JAC","PSP","EYW","HDN","MTJ", "SBN","ANC"])]

print(df\_filtered.head(15)) print(df\_filtered.shape)

df.drop(df[df['dest'].isin(["LEX","TVC","MYR","CHO","BZN","JAC","PSP","EYW","HDN

","MTJ","SBN","ANC"])].index, inplace = True , axis = 0) print(df.shape)

df['delayed'].value\_counts().to\_frame()

saving\_data = df.to\_csv("Data/Processed\_data15.csv", index = False)