**Project Documentation**

Note:

- All time is LOCAL time for departures and arrivals.

- If Single or 2-Digit Values are present for a Time Value, they represent 00:MM (e.g. 3 represents 00:03 or 12:03 AM).

**Column Description**

|  |  |
| --- | --- |
| Column Name | Column Description |
| DAY\_OF\_MONTH | Date of Month |
| DAY\_OF\_WEEK | Day of Week (1: Monday, 7: Sunday) |
| FL\_DATE | Full flight date (M/DD/YYYY) |
| MKT\_CARRIER\_FL\_NUM | Flight Number |
| TAIL\_NUM | Aircraft Tail Number (Usually starts with 'N') |
| ORIGIN | Flight Departure 3-Letter Airport Abbreviation |
| ORIGIN\_CITY\_NAME | Flight Departure City, State Names |
| ORIGIN\_STATE\_NM | Flight Departure State Name |
| DEST | Flight Arrival 3-Letter Airport Abbreviation |
| DEST\_CITY\_NAME | Flight Arrival City, State Names |
| DEST\_STATE\_NM | Flight Arrival State Name |
| CRS\_DEP\_TIME | Scheduled Departure Time (HHMM) (Single or 2-Digit Values Represent 00:MM, e.g. 3 represents 00:03 or 12:03 AM) |
| DEP\_TIME | Actual Departure Time (HHMM) |
| DEP\_DELAY | Departure Delay (Difference Between Actual Departure Time and Scheduled Departure Time in Minutes) |
| DEP\_DELAY\_NEW | Departure Delay Ignoring Early Departures (Listed as 0) |
| DEP\_TIME\_BLK | Scheduled Departure Time in Hourly Block (HHMM) |
| TAXI\_OUT | Time between Airplane Taxi from Gate and Takeoff (WHEELS\_OFF) Time (in Minutes) |
| WHEELS\_OFF | Time of Airplane Takeoff (HHMM) |
| WHEELS\_ON | Time of Airplane Landing (HHMM) |
| TAXI\_IN | Time between Airplane Taxi to Gate and Landing (WHEELS\_ON) Time (in Minutes) |
| CRS\_ARR\_TIME | Scheduled Arrival Time (HHMM) (Single or 2-Digit Values Represent 00:MM, e.g. 3 represents 00:03 or 12:03 AM) |
| ARR\_TIME | Actual Arrival Time (HHMM) |
| ARR\_DELAY | Arrival Delay (Difference Between Actual Arrival Time and Scheduled Arrival Time in Minutes) |
| ARR\_DELAY\_NEW | Arrival Delay Ignoring Early Arrivals (Listed as 0) |
| ARR\_TIME\_BLK | Scheduled Arrival Time in Hourly Block (HHMM) |
| CANCELLED | 0: Flight Not Cancelled, 1: Flight Cancelled |
| CANCELLATION\_CODE | Reason for Cancellation - if Cancelled, Letter Present (A: Carrier, B: Weather, C: National Aviation System, D: Security) |
| CRS\_ELAPSED\_TIME | Scheduled Total Flight Time (in Minutes) |
| ACTUAL\_ELAPSED\_TIME | Actual Total Elapsed Flight Time (in Minutes) |
| AIR\_TIME | Actual Total Elapsed Time Airplane in the Air (in Minutes) |
| DISTANCE | Distance Between Departure and Arrival Airports (in Miles) |
| CARRIER\_DELAY | Carrier Delay (in Minutes) |
| WEATHER\_DELAY | Weather Delay (in Minutes) |
| NAS\_DELAY | National Aviation System Delay (in Minutes) |
| SECURITY\_DELAY | Security Delay (in Minutes) |
| LATE\_AIRCRAFT\_DELAY | Late Aircraft Delay (in Minutes) |

**Data Source:**

[**https://www.kaggle.com/akulbahl/covid19-airline-flight-delays-and-cancellations**](https://www.kaggle.com/akulbahl/covid19-airline-flight-delays-and-cancellations)

**Note: We conside only January data for Southwest Airlines**

**Code:**

**First will start by importing the libraries that we need Pandas, Numpy, matplotlib & seaborn**

import pandas as pd

import numpy as np

from pandas import DataFrame, Series

import matplotlib.pyplot as plt

from pylab import rcParams

import seaborn as sns

from sklearn.datasets import load\_iris

from sklearn.neighbors import KNeighborsClassifier

from sklearn import metrics

from datetime import date

import warnings

%matplotlib inline

warnings.filterwarnings('ignore')

**Then we need to read our data**

df = pd.read\_csv("C:\\Users\\mrnaz\\Desktop\\BootCamp\\My\_Project\\WN-JAN-2020.csv")

**let's have a look at what just we read**

df.shape

df.head(5)

**First, I will rename some columns**

df.rename({'MKT\_CARRIER\_FL\_NUM':'FL\_NUM',

'ORIGIN\_CITY\_NAME':'ORIGIN\_CITY',

'ORIGIN\_CITY\_NAME':'ORIGIN\_CITY',

'ORIGIN\_STATE\_NM':'ORIGIN\_STATE',

'DEST\_CITY\_NAME':'DEST\_CITY',

'DAY\_OF\_MONTH':'MONTH\_DAY',

'DAY\_OF\_WEEK':'WEEK\_DAY',

'DEST\_STATE\_NM':'DEST\_STATE'},

axis=1, inplace=True)

**Second, I will remove the Time in Hourly Block and the 3 letters city code it is not familiar for most people that are not working in the aviation let's just remove it and save some space**

df.drop(['ORIGIN','DEST','DEP\_TIME\_BLK','ARR\_TIME\_BLK'],axis=1,inplace=True)

**we decided that we will not include canceled flights in our study**

**so we will remove the row that have tha value of 1 in the column CANCELLED then we will remove the columns CANCELLED and CANCELLATION\_CODE**

df = df[df.CANCELLED == 0]

df.drop(['CANCELLED','CANCELLATION\_CODE'],axis=1,inplace=True)

**let's have a look on the table after we do these changes**

df.head(100)# or df.sample(5)

**O.K we need to know more about our table such as number of non-null and data type for each column**

df.corr() # View the correlations

df.info()

**Let's convert the column FL\_DATE from object to Date**

df["FL\_DATE"] = pd.to\_datetime(df["FL\_DATE"])

**how still have nulls?**

df.isna().any()

**we need to know number of NaN in each column**

df.isna().sum() # or df.isnull().sum()

**In the columns CARRIER\_DELAY, WEATHER\_DELAY, NAS\_DELAY, SECURITY\_DELAY and LATE\_AIRCRAFT\_DELAY**

**The null means that there is no delay we will just replace the nulls with 0**

#replac null with 0

df['CARRIER\_DELAY'] = df['CARRIER\_DELAY'].fillna(0)

df['WEATHER\_DELAY'] = df['WEATHER\_DELAY'].fillna(0)

df['NAS\_DELAY'] = df['NAS\_DELAY'].fillna(0)

df['SECURITY\_DELAY'] = df['SECURITY\_DELAY'].fillna(0)

**df['LATE\_AIRCRAFT\_DELAY'] = df['LATE\_AIRCRAFT\_DELAY'].fillna(0)**

**We found nulls in columns WHEELS\_ON, TAXI\_IN, ARR\_TIME, ACTUAL\_ELAPSED\_TIME and AIR\_TIME**

**But this column should not have nulls so we will consider it un complete data and we will exclude it from the study**

#remove raw with null

df = df[df.WHEELS\_ON.notnull()]

df = df[df.TAXI\_IN.notnull()]

df = df[df.ARR\_TIME.notnull()]

df = df[df.ACTUAL\_ELAPSED\_TIME.notnull()]

df = df[df.AIR\_TIME.notnull()]

**more descriptions**

df.describe()

**To examine data types and look for cases of missing or potentially wrong data, the describe function helps to get an overview of the data from a statistical summary point of view. This is also useful for spotting any potential errors that may need a closer look.**

df.shape

df.size

df.columns

df.dtypes # This will peovied data type for each column

**I need to list all origin city name**

Print(df.ORIGIN\_CITY.unique())

**To check if there is duplicate rows in the dataset**

sum(df.duplicated())

**This command tells us how many flights in each airport, it will basically calculate city name repeat number**

print (df['ORIGIN\_CITY'].value\_counts())

**We need to how many flights for each plain**

print (df.TAIL\_NUM.value\_counts())

**This will display all flight number 795**

df[df['FL\_NUM'] == 795]

**The average of the departial delay**

av\_DEP\_DELAY = df.DEP\_DELAY.mean()

print (av\_DEP\_DELAY)

**let's Display some Graphs**

**The below figure shows the delay departure time for each city, the minus number means early departure**

plt.figure(figsize =(40,10))

plt.xticks(rotation = 90)

temp = df[['DEP\_DELAY', 'ORIGIN\_CITY']]

temp=pd.DataFrame(temp.groupby('ORIGIN\_CITY')['DEP\_DELAY'].agg('mean')).reset\_index().sort\_values(by= ['DEP\_DELAY'])

temp

sns.barplot(x='ORIGIN\_CITY', y="DEP\_DELAY", data = temp)

Chart, bar chart

Description automatically generated

**Here we can get the same information in a table**

t1=temp

t1

Table

Description automatically generated

**The below figure shows the number of flights time in each day of the week (Air Traffic)**

plt.figure(figsize =(10,5))

plt.xticks(rotation = 45)

temp = df[['AIR\_TIME', 'WEEK\_DAY']]

temp=pd.DataFrame(temp.groupby('WEEK\_DAY')['AIR\_TIME'].agg('sum')).reset\_index().sort\_values(by= ['AIR\_TIME'])

temp

sns.barplot(x='WEEK\_DAY', y="AIR\_TIME", data = temp)

Chart, bar chart

Description automatically generated

**Wednesday has the highest air traffic during the week**

t2=temp

t2

Table

Description automatically generated

**The rate of difference between the scheduled access time and the actual time**

plt.figure(figsize = (10,5))

sns.distplot(x=df['ARR\_DELAY']);

Chart

Description automatically generated

**The figure below gives us an idea of the relationship between flight delays due to civil aviation and the distance between airports**

plt.figure(figsize =(10,10))

temp = df[['NAS\_DELAY', 'DISTANCE']]

temp=pd.DataFrame(temp.groupby('DISTANCE')['NAS\_DELAY'].agg('max')).reset\_index().sort\_values(by= ['NAS\_DELAY'])

temp

sns.kdeplot(x='DISTANCE', y="NAS\_DELAY", data = temp)

Diagram

Description automatically generated

the below graph shows the departial delay for each day of the week

Chart, scatter chart

Description automatically generated

sns.lmplot(x ='DEP\_TIME', y ='CRS\_DEP\_TIME', data = df,ci=None );

# import model and fit

from sklearn.metrics import r2\_score

from sklearn.linear\_model import LinearRegression

# grab the DEP\_TIME and CRS\_DEP\_TIME

#define the independent column

X1 = df.DEP\_TIME.values.reshape(-1, 1)

#then the dependent column?

y1 = df.CRS\_DEP\_TIME.values.reshape(-1, 1)

#1 eate an object to Linear Regression model formula

linreg\_model = LinearRegression()

#2 tting the model == Train Model

linreg\_model.fit(X1, y1)

round(linreg\_model.score(X1, y1)\*100)

import numpy as np

x2 = np.array(100)

x2 = x2.reshape(-1,1)

linreg\_model.predict(x2)