# Airline_Data_Project| NY-DC/MD/VA Flights

Zarak Mahmood

## Introduction

In this project we are going to work on flight data sets, our main priority is get the data report from the flights flying from New York airport to the local airports such as Dulles International Airport (IAD), Baltimore/Washington International Thurgood Marshall Airport (BWI), Ronald Reagan Washington National Airport (DCA).

```
flights_planes = pd.merge(flights2DCMDVA, planes, how='inner',on='tailnum')
flights_planes.head(1)
```

| | date | dep_time | dep_delay | arr_time | arr_delay | carrier | tailnum | flight | origin | dest | ... | hour | minute | year | type | manufacturer | model | engines | seats | speed | engine |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2013-01-01 | 629.0 | -1.0 | 721.0 | -19.0 | WN | N273WN | 4646 | LGA | BWI | ... | 6.0 | 29.0 | 2007.0 | Fixed wing multi engine | BOEING | 737-7H4 | 2 | 140 | NaN | Turbo-fan |

1 rows × 22 columns

Firstly, we created a new data frame named “flight_planes” by merging flights and planes dataset on column tailnum.

## Task 1

### 1a. Total number of seats for all flights planned

```
#1.a
df1=flights_planes.groupby('dest')['seats'].sum().sort_values(ascending=False).reset_index(name='Seats')
df1
```

| | dest | Seats |
|---|---|---|
| 0 | DCA | 906225.0 |
| 1 | IAD | 296004.0 |
| 2 | BWI | 96135.0 |

For 1a we created a new data frame named df1. In 1a we are calculating the total number of seats planned for the three airports from New York. DCA has the highest number of seats with 906,225, followed by IAD with 296,004, and BWI has the least number of seats with only 96,135

### 1b. Day of the year with the highest number of flights

```
#1.b
df2 = flights2DCMDVA.groupby(['date']).size().nlargest(5).reset_index(name='Number of Flights')
print("DAY OF YEAR:", df2['date'][0].dayofyear,df2['date'][1].dayofyear)
print(df2)
#So,here 17th January 2013,11th January 2013, has the highest number of flights with count of 61.
```

```
DAY OF YEAR: 11 17
        date  Number of Flights
0 2013-01-11                 61
1 2013-01-17                 61
2 2013-01-07                 60
3 2013-01-08                 60
4 2013-01-09                 60
```

For part 1b we are calculating for what date of the year we are experiencing the highest number of flights. In order to calculate we created a new data frame named df2 and used the groupby function to get the answer we were seeking. According to our analysis we found that on 1/11/2013 is the busiest day of the year where we will be seeing the highest number of flights.

### 1c. Day of the year with highest number of seats available

```
#1.c
df3 = flights_planes.groupby(['date'])['seats'].sum().sort_values(ascending=False).reset_index(name='Seats Available')
print("DAY OF YEAR:", df3['date'][0].dayofyear)
print(df3.head())
#So,here 28th February 2013 has the highest number of seats with the count of 5379.
```

```
DAY OF YEAR: 59
        date  Seats Available
0 2013-02-28           5379.0
1 2013-01-11           5318.0
2 2013-01-07           5272.0
3 2013-01-24           5268.0
4 2013-02-07           5267.0
```

For part 1c we are calculating the date of the year where the highest number of seats are available. In order to do that we have created a new data frame name df3, and after that we applied group by on it. According to our analysis on 2/28/2013 is that day when the highest number of seats were available

### Task 2

### 2a. Day of the year most cancellations happened

```
#2.a
cancel = flights_planes['dep_time'].isnull() & flights_planes['dep_delay'].isnull() & flights_planes['arr_time'].isnull() & flights_planes['arr_delay'].isnull()
 & flights_planes['air_time'].isnull() & flights_planes['hour'].isnull() & flights_planes['minute'].isnull()
df_cancellations=flights_planes[cancel]
df_cancellations_count=df_cancellations['date'].value_counts().rename_axis('Date').reset_index(name='No.of.Cancellations')
df_cancellations_count
#On 6th March 2013,most of the of flights got cancelled with count of 46.
```

| | Date | No.of.Cancellations |
|---|---|---|
| 0 | 2013-03-06 | 46 |
| 1 | 2013-02-08 | 33 |
| 2 | 2013-09-12 | 25 |
| 3 | 2013-03-08 | 23 |
| 4 | 2013-05-23 | 21 |
| ... | ... | ... |
| 219 | 2013-08-06 | 1 |
| 220 | 2013-08-02 | 1 |
| 221 | 2013-03-03 | 1 |
| 222 | 2013-03-11 | 1 |
| 223 | 2013-12-29 | 1 |

224 rows × 2 columns

For part 2a we are finding what day of the year had the most flights cancellations and for to do that we created a new data frame and did a value count function on one of the new dataframe to calculate what date of the year had the highest cancellations and according to our analysis on the march 6th of 2023 we saw the highest cancellation with count of 46

### 2b. Finding relationship between the weather datasets and cancellations

```
#2.b
df_weathermd = weatherMDdaily
df_weatherny= weatherNYdaily

mergedf['date'] = mergedf.apply(lambda x: str(x['year_x'])+'-'+str(x['month'])+'-'+str(x['day']), axis=1)
mergedf['date'] = pd.to_datetime(mergedf['date'])
mergedf['cancellation'] = mergedf['dep_time'].apply(lambda x: 1 if math.isnan(x) else 0)
date_cancel = mergedf.groupby('date')['cancellation'].sum().reset_index().sort_values (by ='cancellation', ascending=False)
mergedf_cancel = pd.merge(date_cancel, df_weathermd, left_on='date', right_on='Date')

mergedf_cancel[ 'Precipitation'].replace('T',0,inplace=True)
mergedf_cancel[ 'Snowfall' ].replace('T',0, inplace=True)
mergedf_cancel['Snow Depth' ].replace('T',0, inplace=True)
```

```
[24] corr, temp = pearsonr (mergedf_cancel[ 'Precipitation'], mergedf_cancel['cancellation'])
     print("Pearson Corr b/w Precipitation & Cancellation:", round (corr, 2))

     corr, temp = pearsonr (mergedf_cancel[ 'Snowfall'], mergedf_cancel['cancellation'])
     print("Pearson Corr b/w Snowfall & Cancellation:", round (corr, 2))

     corr, temp = pearsonr (mergedf_cancel[ 'Snow Depth'], mergedf_cancel['cancellation'])
     print("Pearson Corr b/w Snow Depth & Cancellation:", round (corr, 2))
```

```
Pearson Corr b/w Precipitation & Cancellation: 0.25
Pearson Corr b/w Snowfall & Cancellation: 0.12
Pearson Corr b/w Snow Depth & Cancellation: 0.03
```

For part 2b we are trying to find if there is a relationship between the weather and cancellations, In order to to do that we have done various calculations on our data set in order to the correlation between weather and cancellations, after cleaning the data applying various function such as groupby, sum, and correlation. After our analysis we have reached to the conclusion that correlation coefficient between weather and cancellation is relatively low, so we could safely say that is no relationship between weather and flight cancellations.

### 2c. Finding relationship between the Federal Holiday Schedule and cancellations

```
#2.c
holidaydf=pd.read_excel('/content/gdrive/Shareddrives/601-Project2/federal-holidays-2013.xlsx')
holidaydf.rename(columns={'Federal holidays 2013':'Date','Unnamed: 1':'Holiday','https://www.calendarpedia.com/':'Day'},inplace=True)
holidaydf = holidaydf[1:-1]
holidaydf['Date'] = pd.to_datetime(holidaydf['Date'])

holidaydf_cancel = pd.merge(holidaydf, mergedf_cancel, how='right')
holidaydf_cancel.fillna(0,inplace=True)
holidaydf_cancel.loc[holidaydf_cancel.Holiday!=0,'Holiday']=1
holidaydf_cancel.head()
```

| | Date | Holiday | Day | date | cancellation | Max Temp | Min Temp | Precipitation | Snowfall | Snow Depth |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2013-03-06 | 0 | 0 | 2013-03-06 | 25 | 40 | 33 | 0.75 | 0.0 | 0 |
| 1 | 2013-02-08 | 0 | 0 | 2013-02-08 | 18 | 42 | 33 | 0.24 | 0.0 | 0 |
| 2 | 2013-03-08 | 0 | 0 | 2013-03-08 | 13 | 49 | 33 | 0.00 | 0.0 | 0 |
| 3 | 2013-05-23 | 0 | 0 | 2013-05-23 | 11 | 80 | 65 | 0.99 | 0.0 | 0 |
| 4 | 2013-09-12 | 0 | 0 | 2013-09-12 | 10 | 89 | 70 | 0.65 | 0.0 | 0 |

```
corr, temp = pearsonr (holidaydf_cancel[ 'Holiday'], holidaydf_cancel['cancellation'])

print('Pearson Corr b/w Holiday & Cancellation:', round(corr,2))
```

```
Pearson Corr b/w Holiday & Cancellation: -0.06
```

For part 2c we are creating a new dataframe to read the new xlsx file, after creating the new dataframe we renamed the column and added a new column naming date, after cleaning and organinzing the data we tried to find the correlation between holiday and cancellations, after applying the correlation function we got -0.06. Correlation coefficient is relativity low, so we could safely conclude that there is no relationship between holidays and flight cancellations.

### 2d. Calculating total number of seats for the cancelled flights and economic loss

```
#2.d
print(f"Total number of seats for the cancelled flights: {df_cancellations['seats'].sum()}")
print(f" Total Economic Loss is: {df_cancellations['seats'].sum()*50}")
```

```
Total number of seats for the cancelled flights: 24032.0
 Total Economic Loss is: 1201600.0
```

---

For part 2d we are calculating the total number of flights and the economic loss, in order to do that we did a sum of total cancelled seats and then multiplied the sum with 50(being the price of each seat). According to our analysis we had 24032 cancelled flights and total economic loss of $1,201,600

### 2e. Ratio of cancelled flights/planned flights for each airline company, and determine the most and least reliable airline

```
[ ]  d1 = df_planned['carrier'].value_counts().to_dict()
     d2 = df_cancellations['carrier'].value_counts().to_dict()
```

```
for i in d1:
    if i in d2:
        print(f"Ratio of cancelled/planned flights for {i} is {d2[i]/d1[i]}")
    else:
        print(f"Ratio of cancelled/planned flights for {i} is {0/d1[i]}")
```

```
Ratio of cancelled/planned flights for EV is 0.06427451663473263
Ratio of cancelled/planned flights for US is 0.051104664137469315
Ratio of cancelled/planned flights for 9E is 0.058823529411764705
Ratio of cancelled/planned flights for MQ is 0.06195426195426196
Ratio of cancelled/planned flights for B6 is 0.011994002998850075
Ratio of cancelled/planned flights for YV is 0.11469534050179211
Ratio of cancelled/planned flights for WN is 0.04
Ratio of cancelled/planned flights for UA is 0.0
Ratio of cancelled/planned flights for DL is 0.0
Ratio of cancelled/planned flights for OO is 0.0
```

```
l1=df_planned['carrier'].value_counts().to_list()
l1
```

```
[5741, 4481, 2448, 2405, 667, 279, 200, 3, 2, 1]
```

```
l2=df_cancellations['carrier'].value_counts().to_list()
l2
#So,here the most reliable airline company is UA,DL,OO as they have 0 cancellations and the least reliable airline company is YV as it has the cancellation rate of 11.4%
```

```
[369, 229, 149, 144, 32, 8, 8]
```

For part 2e we converted d1 and d2 to dictionary and if/else function on d1 and d2, and according to our analysis we were able to find out UA,DL, and OO are most reliable with 0% cancellation ratio, and least reliable carrier is YV with cancellation ratio of 11.4%

**Task 3**

**3a. Calculating average arrival delay for flights that took place in the same day**

```
#3.a
df3a = flights_planes.groupby(['date'])['arr_delay'].mean().reset_index(name='average_delay')
df3a.index=df3a.index+1
df3a.head()
```

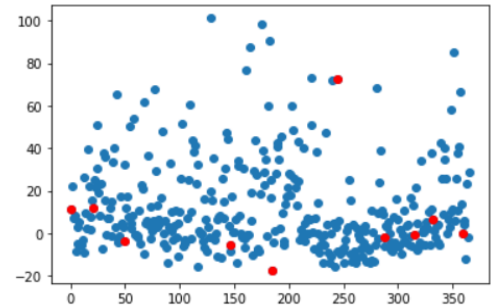
| | date | average_delay |
|---|---|---|
| 1 | 2013-01-01 | 34.075000 |
| 2 | 2013-01-02 | 23.702128 |
| 3 | 2013-01-03 | 8.040816 |
| 4 | 2013-01-04 | 5.326531 |
| 5 | 2013-01-05 | -8.538462 |

For part 3a we are collecting average arrival delay for flights thak took place in the same day and for to do that we created a new dataframe df3a, and after doing that we used groupby function to find the average delay for flights

**Creating scatter plot and marking the federal holidays**

```
plt.scatter(y=df3a['average_delay'],x=df3a.index)
plt.scatter(x=q3b.index,y=q3b['average_delay'], color="red")
```

```
<matplotlib.collections.PathCollection at 0x7f8caf424790>
```




For the second part of question 3a we created a scatter plot of the data that shows that average delay of flights, and in order ro show the Federal Holiday we have colored them in red, they could be distinguished easily when comparing them to the regular days.

### 3b. Correlation between the weather datasets and daily average arrival delay

```
correlations_weather
```

| | date | arr_delay | Date | Max Temp | Min Temp | Precipitation | Snowfall | Snow Depth |
|---|---|---|---|---|---|---|---|---|
| **0** | 2013-01-01 | 34.075000 | 2013-01-01 | 44 | 34 | 0.00 | 0.0 | 0 |
| **1** | 2013-01-02 | 23.702128 | 2013-01-02 | 37 | 26 | 0.00 | 0.0 | 0 |
| **2** | 2013-01-03 | 8.040816 | 2013-01-03 | 38 | 22 | 0.00 | 0.0 | 0 |
| **3** | 2013-01-04 | 5.326531 | 2013-01-04 | 42 | 23 | 0.00 | 0.0 | 0 |
| **4** | 2013-01-05 | -8.538462 | 2013-01-05 | 43 | 31 | 0.00 | 0.0 | 0 |

```
[ ]  convert_dict = {'Snowfall': float,
                     'Snow Depth': float }
     correlations_weather = correlations_weather.astype(convert_dict)
```

```
from scipy.stats import pearsonr
corr, _= pearsonr(correlations_weather['Max Temp'], correlations_weather['arr_delay'])
print('Pearsons correlation Max Temp: %.3f' % _)

corr, _ = pearsonr(correlations_weather['Min Temp'], correlations_weather['arr_delay'])
print('Pearsons correlation Min Temp: %.3f' % _)


corr, _ = pearsonr(correlations_weather['Snowfall'],correlations_weather['arr_delay'])
print('Pearsons correlation Snowfall: %.3f' % _)

corr, _ = pearsonr(correlations_weather['Snow Depth'], correlations_weather['arr_delay'])
print('Pearsons correlation Snow Depth: %.3f' % _)
```

```
Pearsons correlation Max Temp: 0.331
Pearsons correlation Min Temp: 0.001
Pearsons correlation Snowfall: 0.008
Pearsons correlation Snow Depth: 0.054
```

For 3b we have created a statistical method to show our calculations that is there a correlation between weather and daily average arrival delay. According to our analysis we have calculated that correlation with Max temperature is 0.331, correlation with minimum temperature is 0.001, correlation with snowfall is 0.008, and correlation with snow depth is 0.054.

**3c. Finding correlation between the Federal Holiday Schedule and daily average arrival delay**

```
mergedftemp = mergedf
pltdata = mergedftemp.groupby('date')['arr_delay'].mean().reset_index()
pltdata['day'] = pltdata['date'].apply(lambda x:x.timetuple().tm_yday)
pltdata.rename(columns={'date':'Date'},inplace=True)

pltdata_final = pd.merge(pltdata, holidaydf, on='Date', how='left')
pltdata_final.fillna(0,inplace=True)
pltdata_final.loc[pltdata_final.Holiday!=0,'Holiday']=1
pltdata_final.drop(columns=['Day','Date'],inplace=True)
```

```
hol_delay_corr=pltdata_final.groupby('Holiday')['arr_delay'].mean().reset_index()
hol_delay_corr.corr()
```

| | Holiday | arr_delay |
|---|---|---|
| **Holiday** | 1.0 | -1.0 |
| **arr_delay** | -1.0 | 1.0 |

For part 3c we created new dateframe and after creating a new data frame we cleaned and organized the data, after doing that we created another dataframe hol_delay_corr and applied groupby function to it to find the mean of the data and after doing that we find the correlation between the holidays and the average delays. After analyzing we came to the conclusion that there is no correlation between holidays and average delays

**3d. Average arrival delay for all the flights for each arrival airport**

```
#3d
df3d = flights_planes.groupby(['dest'])['arr_delay'].mean().sort_values(ascending=False).reset_index(name='average_delay')
df3d
#Most reliable airport is DCA and least reliable airport is IAD
```

| | dest | average_delay |
|---|---|---|
| **0** | IAD | 13.866071 |
| **1** | BWI | 10.745552 |
| **2** | DCA | 9.069106 |

For part 3d we are calculating average delay for each 3 of airports so we could find out which airport is the most reliable and which airport is the least reliable. According to our analysis

Ronald Reagan Washington National Airport (DCA) is the most reliable airport with the least amount of average delay of 9.06, and the Dulles International Airport (IAD) is the least reliable airport with the highest number of average delays 13.86

### 3e. Calculating average delay for all airlines

```
#3e
df3e = flights_planes.groupby(['carrier'])['arr_delay'].mean().sort_values(ascending=False).reset_index(name='average_delay')
df3e
#Most reliable airport is DL and least reliable airport is YV
```

| | carrier | average_delay |
|---|---|---|
| 0 | YV | 18.917266 |
| 1 | EV | 17.359776 |
| 2 | B6 | 12.805097 |
| 3 | MQ | 10.995401 |
| 4 | US | 5.829000 |
| 5 | WN | 4.915000 |
| 6 | 9E | 3.612890 |
| 7 | OO | 3.000000 |
| 8 | UA | -7.666667 |
| 9 | DL | -8.000000 |

For part 3e we are calculating average delay for all airline and trying to find out which airline is most reliable and which one is the least reliable. We created a new dataframe df3e and applied groupby function to it in order to find the average delay for each carrier. After finding the mean for all flights we have reached the conclusion that Airline MQ is the least reliable airline with the highest average delay, and airline DL is the most reliable airline among all the carriers with least amount of delays.

### 3f. Day of the week wih highest average delay

```
#3f
flights_planes['Day']=flights_planes['date'].dt.day_name()
df3f = flights_planes.groupby(['Day'])['arr_delay'].mean().sort_values(ascending=False).reset_index(name='average_delay')
df3f
#We had the highest delay on Monday
```

| | Day | average_delay |
|---|---|---|
| 0 | Monday | 15.763542 |
| 1 | Friday | 13.205007 |
| 2 | Thursday | 12.697419 |
| 3 | Wednesday | 11.991179 |
| 4 | Tuesday | 9.946862 |
| 5 | Sunday | 6.344271 |
| 6 | Saturday | 2.234884 |

For part 3f we are trying to find the week of the day with the highest average delay. In order to do part 3f we created a new data frame df3f and applied group by function to it to find the average delay among all the day of the week. According to our analysis we see that Monday has the highest average of among all day with 15.76, and Sunday has the least amount of average with 2.23

**3g. Calculating higher average delay: flights that took off in the morning (6 am to 10 am), noon (11 am to 2 pm), afternoon (3 pm to 5pm), or evening (6 pm to 10 pm).**

```
time_groups=pd.cut(df_planned['dep_time'],ordered=False,bins=[600, 1000, 1100,1400,1500,1700,1800,2200],
labels=['morning','','noon','','afternoon','','evening'])
```

```
df_day_timings=df_planned.groupby(time_groups)['arr_delay'].mean().reset_index()
df_day_timings.iloc[1:]
#Flights that took off in afternoon has the highest average delay.
```

| | dep_time | arr_delay |
|---|---|---|
| 1 | afternoon | 18.990246 |
| 2 | evening | 16.761868 |
| 3 | morning | -1.723660 |
| 4 | noon | 10.229515 |

For part 3g we are calculating which time of the day has seen the highest number of flight delays and in order to that we created a new dataframe df_day_timings and applied group by on it. According to our analysis we have calculated that afternoon time has has seen the highest number of average delays with 18.99, and the morning time with the least amount of average delay -1.72.

**3h. Number of airplanes used in these flights manufactured by BOEING, EMBRAER, and AURBUS separately.**

```
#3.h
df_manufacturer=flights2DCMDVA.merge(planes[['tailnum','manufacturer']],on='tailnum', how='left')
print("BOEING",(df_manufacturer['manufacturer']=='BOEING').sum())
print("EMBRAER",(df_manufacturer['manufacturer']=='EMBRAER').sum())
print("AIRBUS",(df_manufacturer['manufacturer']=='AIRBUS').sum())
#BOEING has manufactured 208,EMBRAER manufactured 4606 and AIRBUS manufactured 4 airplanes
```

```
BOEING 208
EMBRAER 4606
AIRBUS 4
```

For part 3h we are finding the number of planes used in these flights manufactured by BOEING, EMBRAER, and AURBUS separately. We created a new dataframe q3h and applied groupby function to it to find the count of number of flights. If you take a look at the image above you could easily see the number of flights by each manufacturer. According to our analysis conclude that EMBRAER has the highest number of flights among all the manufacturers.

**Task 4**

**Creating Linear Regression Model Step by Step**

**Linear Regression:**

A linear regression model is a statistical model that is used to predict a continuous target variable using a linear relationship between the independent variables and the dependent variable. It is one of the most commonly used techniques in machine learning and is used to estimate the relationship between one or more independent variables and a continuous target variable. The linear regression model assumes that the relationship between the dependent and independent variables is linear, which means that the change in the dependent variable is directly proportional to the change in the independent variables.

**Problem statement approach workflow:**

**Step 1:**

Gathered the data related to the problem and stored it in multiple data frames.

```
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
le= LabelEncoder()
df_planned=pd.read_excel('/content/gdrive/Shareddrives/601-Project2/flights2DCMDVA.xlsx')
X=df_planned[['year','month','day','carrier','origin','dest','distance']]
y=df_planned[['arr_delay']]

flights_test_data=pd.read_excel('/content/gdrive/Shareddrives/601-Project2/flights_test_data.xlsx')
```

**Step 2:**

we checked and removed all the null values in the training data and grabbed the necessary data columns

**Step 3:**

The Dataset consists of Numerical and categorical data columns

Categorical data columns = “carrier”, “origin”, “dest”

Numerical data columns = “year”, “month”, “day”, “distance”

To Transform the categorical data to numerics we used OneHotEncoder library with the help of scikit-learn

```
categorical_col=['carrier','origin','dest']
X[categorical_col] = X[categorical_col].apply(le.fit_transform)
y=y.fillna(y.mean())
```

```
/usr/local/lib/python3.8/dist-packages/pandas/core/frame.py:3641: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self[k1] = value[k2]
```

**Step 4:** predicting the average delay of testing data using the linear regression model.

```
regressor = LinearRegression()
regressor.fit(X_train, y_train)

LinearRegression()
```

```
[ ] flights_test_data[categorical_col]=flights_test_data[categorical_col].apply(le.fit_transform)
```

```
[ ] y_pred = regressor.predict(flights_test_data)
```

```
[ ] flights_test_data['arr_delay']=y_pred
    flights_test_data.head()
```

| | year | month | day | carrier | origin | dest | distance | arr_delay |
|---|---|---|---|---|---|---|---|---|
| **0** | 2013 | 1 | 6 | 2 | 1 | 1 | 213 | 9.683063 |
| **1** | 2013 | 1 | 25 | 1 | 2 | 2 | 229 | 10.718957 |
| **2** | 2013 | 2 | 11 | 2 | 1 | 1 | 213 | 9.527239 |
| **3** | 2013 | 4 | 14 | 3 | 2 | 1 | 214 | 5.735312 |
| **4** | 2013 | 4 | 29 | 1 | 2 | 2 | 229 | 9.677370 |

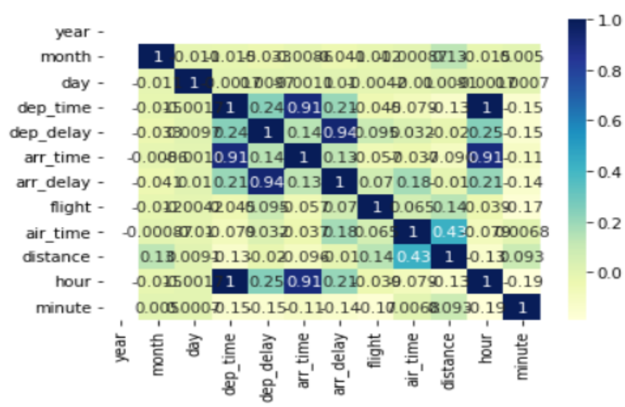
**Step 5:** For step 5 we are trying to find is our model accurate enough to make reliable predictions, our mean squared root comes up to 0.27 which means our model can relatively predict data accurately, (if numbers are between 0.2 - 0.5 than predicting probability is higher)

```
[ ] from sklearn import metrics
    print(f"Root Mean Squared Error:{(np.sqrt(metrics.mean_squared_error(y_test.sample(20), y_pred)))/100}")
    #RMSE values between 0.2 and 0.5 shows that the model can relatively predict the data accurately.

Root Mean Squared Error:0.27782448360905676
```

**Step 6:** For step 6 we created a heat map to give a better understanding for our model

```
[ ] sns.heatmap(flights2DCMDVA.corr(), cmap="YlGnBu", annot = True)
    plt.show()
```




**Task 5**

**Build a logistic regression model to guess the 3 cancelled flights given**

**Step 1**
Gathered the data related to the problem and stored it in multiple data frames.

```
from sklearn.linear_model import LogisticRegression
df_cancellations=df_planned[cancel]
df_flight_test_guess=flights_test_data
df_flight_test_guess.drop('arr_delay', inplace=True, axis=1)
X_CAN=df_cancellations[['year','month','day','carrier','origin','dest','distance']]
y_CAN=df_cancellations[['flight']]
```

**Step 2**
The Dataset consists of Numerical and categorical data columns
Categorical data columns = "carrier", "origin", "dest"
Numerical data columns = "year", "month", "day", "distance"
To Transform the categorical data to numerics we used OneHotEncoder library with the help of scikit-learn

```
categorical_col=['carrier','origin','dest']
X_CAN[categorical_col] = X_CAN[categorical_col].apply(le.fit_transform)
y_CAN[['flight']] = y_CAN[['flight']].apply(le.fit_transform)
```

```
/usr/local/lib/python3.8/dist-packages/pandas/core/frame.py:3641: SettingWithCopyWarning
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
  self[k1] = value[k2]
```

**Step 3** In this step we are predicting and making speculation regarding our model

```
[ ]  y_pred = logregressor.predict(df_flight_test_guess)
```

```
[ ]  df_flight_test_guess['flight']=y_pred
```

```
[ ]  df_flight_test_guess.head(3)
```

| | year | month | day | carrier | origin | dest | distance | flight |
|---|---|---|---|---|---|---|---|---|
| **0** | 2013 | 1 | 6 | 2 | 1 | 1 | 213 | 38 |
| **1** | 2013 | 1 | 25 | 1 | 2 | 2 | 229 | 81 |
| **2** | 2013 | 2 | 11 | 2 | 1 | 1 | 213 | 38 |

**Step 4** For step 4 we are checking how accurate we can can predict the data, we created a new dataframe score and used prediction function on it to check its accuracy and we ended up with

0.05, For a normal eye we converted it into int and multiplied it with 100 so we could get our answer in percentage, and we came up that we have 5% accuracy.

```
from sklearn.metrics import accuracy_score
score = accuracy_score(y_test_1.sample(20).to_numpy(),y_pred)
score
```

```
0.05
```

```
print(f"Accuracy is {int(score*100)}%")
#Accuracy of this prediction is very low.
```

```
Accuracy is 5%
```