S02_T05_Data_Exploration

February 2, 2022

1 IT Academy - Data Science Itinerary

1.1 S02 T05: Data_Exploration:

1 :

```
Download the "Airlines Delay: Airline on-time statistics and delay causes"
data set and upload it to a Dataframe pandas.
Explore the data it contains, and keep only the columns you consider relevant
```

[154]:
```
# Importing required library
import pandas as pd
```

[155]:
```
#data files are available in the "./data/" directory.
path = "./data/DelayedFlights.csv"
```

[156]:
```
#Reading the Data
df = pd.read_csv(path)
```

- Let's take a quick look at what the data looks like:

[218]:
```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1936125 entries, 0 to 1936757
Data columns (total 30 columns):
 #   Column             Dtype
---  ------             -----
 0   Unnamed: 0         int64
 1   Year               int64
 2   Month              int64
 3   DayofMonth         int64
 4   DayOfWeek          int64
 5   DepTime            float64
 6   CRSDepTime         int64
 7   ArrTime            float64
 8   CRSArrTime         int64
 9   UniqueCarrier      object
 10  FlightNum          int64
 11  TailNum            object
```

```
 12  ActualElapsedTime  float64
 13  CRSElapsedTime     float64
 14  AirTime            float64
 15  ArrDelay           float64
 16  DepDelay           float64
 17  Origin             object
 18  Dest               object
 19  Distance           int64
 20  TaxiIn             float64
 21  TaxiOut            float64
 22  Cancelled          int64
 23  CancellationCode   object
 24  Diverted           int64
 25  CarrierDelay       float64
 26  WeatherDelay       float64
 27  NASDelay           float64
 28  SecurityDelay      float64
 29  LateAircraftDelay  float64
dtypes: float64(14), int64(11), object(5)
memory usage: 457.9+ MB
None
```

1.1.1 This dataset is composed by the following variables:

1. **Year** = 2008
2. **Month** = 1-12
3. **DayofMonth** = 1-31
4. **DayOfWeek** = 1 (Monday) - 7 (Sunday)
5. **DepTime** = actual departure time (local, hhmm)
6. **CRSDepTime** = scheduled departure time (local, hhmm)
7. **ArrTime** = actual arrival time (local, hhmm)
8. **CRSArrTime** = scheduled arrival time (local, hhmm)
9. **UniqueCarrier** = unique carrier code
10. **FlightNum** = flight number
11. **TailNum** = plane tail number: aircraft registration, unique aircraft identifier
12. **ActualElapsedTime** = in minutes
13. **CRSElapsedTime** = in minutes
14. **AirTime** = in minutes
15. **ArrDelay** = arrival delay, in minutes: A flight is counted as "on time" if it operated less than 15 minutes later the scheduled time shown in the carriers' Computerized Reservations Systems (CRS).
16. **DepDelay** = departure delay, in minutes
17. **Origin** = origin IATA airport code
18. **Dest** = destination IATA airport code
19. **Distance** = in miles
20. **TaxiIn** = taxi in time, in minutes
21. **TaxiOut** = taxi out time in minutes
22. **Cancelled** = *was the flight cancelled

23. **CancellationCode** = reason for cancellation (A = carrier, B = weather, C = NAS, D = security)
24. **Diverted** = 1 = yes, 0 = no
25. **CarrierDelay** = in minutes: Carrier delay is within the control of the air carrier. Examples of occurrences that may determine carrier delay are: aircraft cleaning, aircraft damage, awaiting the arrival of connecting passengers or crew, baggage, bird strike, cargo loading, catering, computer, outage-carrier equipment, crew legality (pilot or attendant rest), damage by hazardous goods, engineering inspection, fueling, handling disabled passengers, late crew, lavatory servicing, maintenance, oversales, potable water servicing, removal of unruly passenger, slow boarding or seating, stowing carry-on baggage, weight and balance delays.
26. **WeatherDelay** = in minutes: Weather delay is caused by extreme or hazardous weather conditions that are forecasted or manifest themselves on point of departure, enroute, or on point of arrival.
27. **NASDelay** = in minutes: Delay that is within the control of the National Airspace System (NAS) may include: non-extreme weather conditions, airport operations, heavy traffic volume, air traffic control, etc.
28. **SecurityDelay** = in minutes: Security delay is caused by evacuation of a terminal or concourse, re-boarding of aircraft because of security breach, inoperative screening equipment and/or long lines in excess of 29 minutes at screening areas.
29. **LateAircraftDelay** = in minutes: Arrival delay at an airport due to the late arrival of the same aircraft at a previous airport. The ripple effect of an earlier delay at downstream airports is referred to as delay propagation.

[158]:
```
nRow, nCol = df.shape
print(f'There are {nRow} rows and {nCol} columns')
```

```
There are 1936758 rows and 30 columns
```

[220]:
```
# showing head data
df.head()
```

[220]:

	Unnamed: 0	Year	Month	DayofMonth	DayOfWeek	DepTime	CRSDepTime	\
0	0	2008	1	3	4	2003.0	1955	
1	1	2008	1	3	4	754.0	735	
2	2	2008	1	3	4	628.0	620	
3	4	2008	1	3	4	1829.0	1755	
4	5	2008	1	3	4	1940.0	1915	

	ArrTime	CRSArrTime	UniqueCarrier	…	TaxiIn	TaxiOut	Cancelled	\
0	2211.0	2225	WN	…	4.0	8.0	0	
1	1002.0	1000	WN	…	5.0	10.0	0	
2	804.0	750	WN	…	3.0	17.0	0	
3	1959.0	1925	WN	…	3.0	10.0	0	
4	2121.0	2110	WN	…	4.0	10.0	0	

	CancellationCode	Diverted	CarrierDelay	WeatherDelay	NASDelay	\
0	N	0	NaN	NaN	NaN	
1	N	0	NaN	NaN	NaN	

```
2                  N         0           NaN           NaN       NaN
3                  N         0           2.0           0.0       0.0
4                  N         0           NaN           NaN       NaN

  SecurityDelay  LateAircraftDelay
0           NaN                NaN
1           NaN                NaN
2           NaN                NaN
3           0.0               32.0
4           NaN                NaN

[5 rows x 30 columns]
```

- **The first thing to keep in mind is that if a flight has been canceled it is impossible for it to reach its destination, so we will not consider those flights for any analysis:**

[160]:
```
df = df.loc[df['Cancelled'] == 0]
```

- Let's use the following list of columns:

[161]:
```
selec_col = ["Month","DayofMonth","DayOfWeek","DepTime",\
             "CRSDepTime","UniqueCarrier","FlightNum","TailNum",\
            ␣
 ↪"AirTime","ArrDelay","DepDelay","Origin","Dest","Distance","CarrierDelay",\
             "WeatherDelay","NASDelay","SecurityDelay","LateAircraftDelay"]
```

- Create new dataframe with only columns in "selec_col":

[162]:
```
new_df = df[selec_col].copy()
```

[221]:
```
print(new_df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1901331 entries, 0 to 1936757
Data columns (total 22 columns):
 #   Column             Dtype
---  ------             -----
 0   Month              int64
 1   DayofMonth         int64
 2   DayOfWeek          int64
 3   DepTime            float64
 4   CRSDepTime         int64
 5   UniqueCarrier      object
 6   FlightNum          int64
 7   TailNum            object
 8   AirTime            float64
 9   ArrDelay           float64
 10  DepDelay           float64
 11  Origin             object
```

```
 12  Dest               object
 13  Distance           int64
 14  CarrierDelay       float64
 15  WeatherDelay       float64
 16  NASDelay           float64
 17  SecurityDelay      float64
 18  LateAircraftDelay  float64
 19  AvgFlightSpeed     float64
 20  Delay              bool
 21  Total_Delay        float64
dtypes: bool(1), float64(11), int64(6), object(4)
memory usage: 320.9+ MB
None
```

[164]:
```
new_df.head()
```

[164]:

	Month	DayofMonth	DayOfWeek	DepTime	CRSDepTime	UniqueCarrier	FlightNum
0	1	3	4	2003.0	1955	WN	335
1	1	3	4	754.0	735	WN	3231
2	1	3	4	628.0	620	WN	448
3	1	3	4	1829.0	1755	WN	3920
4	1	3	4	1940.0	1915	WN	378

	TailNum	AirTime	ArrDelay	DepDelay	Origin	Dest	Distance	CarrierDelay
0	N712SW	116.0	-14.0	8.0	IAD	TPA	810	NaN
1	N772SW	113.0	2.0	19.0	IAD	TPA	810	NaN
2	N428WN	76.0	14.0	8.0	IND	BWI	515	NaN
3	N464WN	77.0	34.0	34.0	IND	BWI	515	2.0
4	N726SW	87.0	11.0	25.0	IND	JAX	688	NaN

	WeatherDelay	NASDelay	SecurityDelay	LateAircraftDelay
0	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN
3	0.0	0.0	0.0	32.0
4	NaN	NaN	NaN	NaN

2 :

- Make a complete report of the data set:
  - Summarize the columns of interest statistically
  - Find how many missing data are per column
  - Create new columns (average flight speed, whether late or not …)
  - Table of airlines with the most accumulated delays
  - What are the longest flights? And the most delayed?
  - Etc.
- Summarize columns of interest statistically:

[165]:
```
new_df.describe().round()
```

[165]:

	Month	DayofMonth	DayOfWeek	DepTime	CRSDepTime	FlightNum	\
count	1936125.0	1936125.0	1936125.0	1936125.0	1936125.0	1936125.0	
mean	6.0	16.0	4.0	1519.0	1467.0	2184.0	
std	3.0	9.0	2.0	450.0	425.0	1945.0	
min	1.0	1.0	1.0	1.0	0.0	1.0	
25%	3.0	8.0	2.0	1203.0	1135.0	610.0	
50%	6.0	16.0	4.0	1545.0	1510.0	1543.0	
75%	9.0	23.0	6.0	1900.0	1815.0	3422.0	
max	12.0	31.0	7.0	2400.0	2359.0	9742.0	

	AirTime	ArrDelay	DepDelay	Distance	CarrierDelay	WeatherDelay	\
count	1928371.0	1928371.0	1936125.0	1936125.0	1247488.0	1247488.0	
mean	108.0	42.0	43.0	766.0	19.0	4.0	
std	69.0	57.0	53.0	574.0	44.0	21.0	
min	0.0	-109.0	6.0	11.0	0.0	0.0	
25%	58.0	9.0	12.0	338.0	0.0	0.0	
50%	90.0	24.0	24.0	606.0	2.0	0.0	
75%	137.0	56.0	53.0	998.0	21.0	0.0	
max	1091.0	2461.0	2467.0	4962.0	2436.0	1352.0	

	NASDelay	SecurityDelay	LateAircraftDelay
count	1247488.0	1247488.0	1247488.0
mean	15.0	0.0	25.0
std	34.0	2.0	42.0
min	0.0	0.0	0.0
25%	0.0	0.0	0.0
50%	2.0	0.0	8.0
75%	15.0	0.0	33.0
max	1357.0	392.0	1316.0

[166]:
```
new_df.groupby("FlightNum")[["ArrDelay","DepDelay"]].agg([min, max, sum])
```

[166]:

	ArrDelay			DepDelay		
	min	max	sum	min	max	sum
FlightNum						
1	-40.0	454.0	32220.0	6.0	452.0	34862.0
2	-32.0	909.0	34407.0	6.0	921.0	35189.0
3	-38.0	544.0	46207.0	6.0	550.0	47440.0
4	-34.0	376.0	30495.0	6.0	371.0	30522.0
5	-38.0	664.0	35744.0	6.0	675.0	36213.0
…	…	…	…	…	…	…
8403	17.0	17.0	17.0	16.0	16.0	16.0
9002	12.0	12.0	12.0	11.0	11.0	11.0
9740	-1.0	94.0	333.0	6.0	284.0	748.0
9741	49.0	49.0	49.0	7.0	58.0	89.0

```
9742           NaN    NaN      0.0      9.0    9.0      9.0

[7499 rows x 6 columns]
```

[167]:
```
new_df.groupby(["Origin", "Dest"])[["ArrDelay", "DepDelay"]].mean()
```

[167]:
```
              ArrDelay   DepDelay
Origin Dest
ABE    ATL   50.405622  48.526104
       BHM   -3.000000  11.000000
       CLE   52.088235  44.029412
       CLT   43.937500  40.593750
       CVG   38.666667  39.416667
…                  …          …
YUM    IPL   24.830508  25.932203
       LAS   45.480000  50.680000
       LAX   33.397516  33.534161
       PHX   39.978979  42.435435
       SLC   21.899083  27.513761

[5205 rows x 2 columns]
```

- Missing values by column:

[172]:
```
print(new_df.isnull().sum())
```

```
Month                   0
DayofMonth              0
DayOfWeek               0
DepTime                 0
CRSDepTime              0
UniqueCarrier           0
FlightNum               0
TailNum                 5
AirTime              7754
ArrDelay             7754
DepDelay                0
Origin                  0
Dest                    0
Distance                0
CarrierDelay       688637
WeatherDelay       688637
NASDelay           688637
SecurityDelay      688637
LateAircraftDelay  688637
dtype: int64
```

- Let's replace the missing values using *pd.fillna()* :

[175]:
```
new_df = new_df.fillna(0)
```

- checking the missing values again:

[176]:
```
print(new_df.isnull().sum())
```

```
Month                0
DayofMonth           0
DayOfWeek            0
DepTime              0
CRSDepTime           0
UniqueCarrier        0
FlightNum            0
TailNum              0
AirTime              0
ArrDelay             0
DepDelay             0
Origin               0
Dest                 0
Distance             0
CarrierDelay         0
WeatherDelay         0
NASDelay             0
SecurityDelay        0
LateAircraftDelay    0
dtype: int64
```

- Create new columns (average flight speed, whether late or not …):

**Average speed is calculated by dividing the total distance that something has traveled by the total amount of time it took it to travel that distance*

[177]:
```
# average flight speed

new_df["AvgFlightSpeed"] = new_df["Distance"] / (new_df["AirTime"]/60)
```

[178]:
```
#whether late or not

new_df["Delay"] = (new_df["ArrDelay"] != 0.0) & (new_df["DepDelay"] != 0.0)
```

[179]:
```
#total time Delay:

new_df["Total_Delay"] = new_df["ArrDelay"] + new_df ["DepDelay"]
```

- if we look at the DataFrame again, the new columns will appear:

[186]:
```
new_df.head()
```

[186]:

	Month	DayofMonth	DayOfWeek	DepTime	CRSDepTime	UniqueCarrier	FlightNum	\
0	1	3	4	2003.0	1955	WN	335	

```
1      1           3          4    754.0         735            WN       3231
2      1           3          4    628.0         620            WN        448
3      1           3          4   1829.0        1755            WN       3920
4      1           3          4   1940.0        1915            WN        378

  TailNum  AirTime  ArrDelay  …  Dest Distance CarrierDelay  WeatherDelay  \
0  N712SW    116.0     -14.0  …   TPA      810          0.0           0.0
1  N772SW    113.0       2.0  …   TPA      810          0.0           0.0
2  N428WN     76.0      14.0  …   BWI      515          0.0           0.0
3  N464WN     77.0      34.0  …   BWI      515          2.0           0.0
4  N726SW     87.0      11.0  …   JAX      688          0.0           0.0

   NASDelay  SecurityDelay  LateAircraftDelay  AvgFlightSpeed  Delay  \
0       0.0            0.0                0.0      418.965517   True
1       0.0            0.0                0.0      430.088496   True
2       0.0            0.0                0.0      406.578947   True
3       0.0            0.0               32.0      401.298701   True
4       0.0            0.0                0.0      474.482759   True

   Total_Delay
0         -6.0
1         21.0
2         22.0
3         68.0
4         36.0

[5 rows x 22 columns]
```

- Table of airlines with the most accumulated delays:

we can use pivot tables:

[215]:
```
import numpy as np

new_df.pivot_table(values=["ArrDelay","DepDelay","Total_Delay"],\
                   index=["UniqueCarrier"],aggfunc=np.sum).
↪sort_values(by=["Total_Delay"],ascending=False)
```

[215]:
```
                 ArrDelay    DepDelay  Total_Delay
UniqueCarrier
WN             11319092.0  12939239.0   24258331.0
AA              8889066.0   8831063.0   17720129.0
UA              6733013.0   7009850.0   13742863.0
MQ              6396704.0   6140112.0   12536816.0
OO              5978936.0   5874040.0   11852976.0
XE              5176042.0   5139974.0   10316016.0
DL              4535644.0   4418474.0    8954118.0
CO              4045932.0   4273624.0    8319556.0
```

```
EV              3888131.0   3935485.0    7823616.0
YV              3691461.0   3687902.0    7379363.0
US              3571867.0   3782039.0    7353906.0
NW              3462075.0   3244063.0    6706138.0
FL              3100150.0   3006130.0    6106280.0
B6              3025749.0   3008825.0    6034574.0
OH              2675993.0   2558691.0    5234684.0
9E              2420468.0   2434743.0    4855211.0
AS              1406735.0   1475058.0    2881793.0
F9               788549.0    776628.0    1565177.0
HA               255613.0    246587.0     502200.0
AQ                15814.0     19246.0      35060.0
```

or do the same using groupby()

[213]:
```
new_df.groupby(["UniqueCarrier"])[["ArrDelay", "DepDelay","Total_Delay"]].
↪agg(sum)\
                .sort_values(by="Total_Delay",ascending=False)
```

[213]:
```
                 ArrDelay    DepDelay  Total_Delay
UniqueCarrier
WN             11319092.0  12939239.0   24258331.0
AA              8889066.0   8831063.0   17720129.0
UA              6733013.0   7009850.0   13742863.0
MQ              6396704.0   6140112.0   12536816.0
OO              5978936.0   5874040.0   11852976.0
XE              5176042.0   5139974.0   10316016.0
DL              4535644.0   4418474.0    8954118.0
CO              4045932.0   4273624.0    8319556.0
EV              3888131.0   3935485.0    7823616.0
YV              3691461.0   3687902.0    7379363.0
US              3571867.0   3782039.0    7353906.0
NW              3462075.0   3244063.0    6706138.0
FL              3100150.0   3006130.0    6106280.0
B6              3025749.0   3008825.0    6034574.0
OH              2675993.0   2558691.0    5234684.0
9E              2420468.0   2434743.0    4855211.0
AS              1406735.0   1475058.0    2881793.0
F9               788549.0    776628.0    1565177.0
HA               255613.0    246587.0     502200.0
AQ                15814.0     19246.0      35060.0
```

- What are the longest flights?:

[222]:
```
new_df[["UniqueCarrier","FlightNum","Distance"]].\
        sort_values(by="Distance",ascending=False).head()
```

[222]:

	UniqueCarrier	FlightNum	Distance
1420365	CO	14	4962
556364	CO	15	4962
1620824	CO	14	4962
1415175	CO	15	4962
368631	CO	14	4962

- And the most delayed?:

[223]:
```
new_df.groupby(["UniqueCarrier","FlightNum"])[["ArrDelay",␣
↪"DepDelay","Total_Delay"]].agg(max)\
                .sort_values(by="Total_Delay",ascending=False).head()
```

[223]:

		ArrDelay	DepDelay	Total_Delay
UniqueCarrier	FlightNum			
NW	1699	2453.0	2467.0	4920.0
	808	2461.0	2457.0	4918.0
	1107	1951.0	1952.0	3903.0
MQ	3538	1707.0	1710.0	3417.0
NW	357	1655.0	1597.0	3252.0

3 :

Export the data set clean and with the new columns to Excel.

[217]:
```
new_df.to_csv("./data/new_df.csv", index = False)
```

[ ]:
```

```