flight_data_analysis

February 12, 2021

1 Performing data analysis on a large dataset using Python

I have had the opportunity to study Python and data analysis as part of my MBA coursework this semester at Columbia Business School. For my Business Analytics II class, we were asked to analyze airline flight delays using Tableau. However, I thought that performing this analysis within Python would also be an interesting challenge. This notebook shows how I converted a dataset containing millions of flights.

To operate on this data, I'll first import the Pandas library and then read the data from my 305-megabyte CSV file into a dataframe. Next, I'll apply various Pandas functions to the dataset to understand what this dataset contains and how it's structured.

[2]:
```
import pandas as pd
df = pd.read_csv("../flights.csv")
```

[3]:
```
df.head(10) #Displays first 10 rows
```

[3]:

	Unnamed: 0	YEAR	MONTH	DAY	DAY_OF_WEEK	AIRLINE	ORIGIN_AIRPORT	DESTINATION_AIRPORT	DEPARTURE_DELAY	DISTANCE	ARRIVAL_DELAY	DIVERTED
0	0	2015	1	1	4	AS	ANC	SEA	-11.0	1448	-22.0	0
1	1	2015	1	1	4	AA	LAX	PBI	-8.0	2330	-9.0	0
2	2	2015	1	1	4	US	SFO	CLT	-2.0	2296	5.0	0
3	3	2015	1	1	4	AA	LAX	MIA	-5.0	2342	-9.0	0
4	4	2015	1	1	4	AS	SEA	ANC	-1.0	1448	-21.0	0
5	5	2015	1	1	4	DL	SFO	MSP	-5.0	1589	8.0	0
6	6	2015	1	1	4	NK	LAS	MSP	-6.0	1299	-17.0	0
7	7	2015	1	1	4	US	LAX	CLT	14.0	2125	-10.0	0
8	8	2015	1	1	4	AA	SFO					
9	9	2015	1	1	4	DL	LAS					

```
8                 DFW           -11.0      1464          -13.0         0
9                 ATL             3.0      1747          -15.0         0

   CANCELLED  AIR_SYSTEM_DELAY  SECURITY_DELAY  AIRLINE_DELAY  \
0          0               NaN             NaN            NaN
1          0               NaN             NaN            NaN
2          0               NaN             NaN            NaN
3          0               NaN             NaN            NaN
4          0               NaN             NaN            NaN
5          0               NaN             NaN            NaN
6          0               NaN             NaN            NaN
7          0               NaN             NaN            NaN
8          0               NaN             NaN            NaN
9          0               NaN             NaN            NaN

   LATE_AIRCRAFT_DELAY  WEATHER_DELAY  DELAYED
0                  NaN            NaN        0
1                  NaN            NaN        0
2                  NaN            NaN        0
3                  NaN            NaN        0
4                  NaN            NaN        0
5                  NaN            NaN        0
6                  NaN            NaN        0
7                  NaN            NaN        0
8                  NaN            NaN        0
9                  NaN            NaN        0
```

[4]:
```
df.tail(10) # Displays final 10 rows
```

[4]:
```
         Unnamed: 0  YEAR  MONTH  DAY  DAY_OF_WEEK AIRLINE ORIGIN_AIRPORT  \
5332904     5819069  2015     12   31            4      B6            LAS
5332905     5819070  2015     12   31            4      B6            RNO
5332906     5819071  2015     12   31            4      B6            SLC
5332907     5819072  2015     12   31            4      B6            DEN
5332908     5819073  2015     12   31            4      B6            ABQ
5332909     5819074  2015     12   31            4      B6            LAX
5332910     5819075  2015     12   31            4      B6            JFK
5332911     5819076  2015     12   31            4      B6            JFK
5332912     5819077  2015     12   31            4      B6            MCO
5332913     5819078  2015     12   31            4      B6            JFK

        DESTINATION_AIRPORT  DEPARTURE_DELAY  DISTANCE  ARRIVAL_DELAY  \
5332904                 JFK            159.0      2248          159.0
5332905                 JFK              0.0      2411          -21.0
5332906                 MCO             16.0      1931           17.0
5332907                 JFK              7.0      1626          -11.0
5332908                 JFK             16.0      1826            3.0
```

```
5332909                 BOS            -4.0      2611          -26.0
5332910                 PSE            -4.0      1617          -16.0
5332911                 SJU            -9.0      1598           -8.0
5332912                 SJU            -6.0      1189          -10.0
5332913                 BQN            15.0      1576            2.0

         DIVERTED  CANCELLED  AIR_SYSTEM_DELAY  SECURITY_DELAY  AIRLINE_DELAY  \
5332904         0          0               0.0             0.0          159.0
5332905         0          0               NaN             NaN            NaN
5332906         0          0               1.0             0.0           16.0
5332907         0          0               NaN             NaN            NaN
5332908         0          0               NaN             NaN            NaN
5332909         0          0               NaN             NaN            NaN
5332910         0          0               NaN             NaN            NaN
5332911         0          0               NaN             NaN            NaN
5332912         0          0               NaN             NaN            NaN
5332913         0          0               NaN             NaN            NaN

         LATE_AIRCRAFT_DELAY  WEATHER_DELAY  DELAYED
5332904                  0.0            0.0        1
5332905                  NaN            NaN        0
5332906                  0.0            0.0        1
5332907                  NaN            NaN        0
5332908                  NaN            NaN        0
5332909                  NaN            NaN        0
5332910                  NaN            NaN        0
5332911                  NaN            NaN        0
5332912                  NaN            NaN        0
5332913                  NaN            NaN        0
```

[5]:
```
df.shape # Returns # of rows and # of columns as a tuple
```

[5]:
```
(5332914, 19)
```

[6]:
```
df.size # Prints the number of values (rows * columns)
```

[6]:
```
101325366
```

[7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5332914 entries, 0 to 5332913
Data columns (total 19 columns):
 #   Column               Dtype
---  ------               -----
 0   Unnamed: 0           int64
 1   YEAR                 int64
 2   MONTH                int64
```

```
 3   DAY                  int64
 4   DAY_OF_WEEK          int64
 5   AIRLINE              object
 6   ORIGIN_AIRPORT       object
 7   DESTINATION_AIRPORT  object
 8   DEPARTURE_DELAY      float64
 9   DISTANCE             int64
 10  ARRIVAL_DELAY        float64
 11  DIVERTED             int64
 12  CANCELLED            int64
 13  AIR_SYSTEM_DELAY     float64
 14  SECURITY_DELAY       float64
 15  AIRLINE_DELAY        float64
 16  LATE_AIRCRAFT_DELAY  float64
 17  WEATHER_DELAY        float64
 18  DELAYED              int64
dtypes: float64(7), int64(9), object(3)
memory usage: 773.1+ MB
```

As shown by the shape() function, this dataset contains over 5.3 million rows, which well exceeds the number of rows that Excel can display. Therefore, it makes sense to analyze this data using Python instead.

In order to analyze flight delays by airline, I'll first want to figure out which airlines are in our dataset, then store those names into an array. Having an array of airline names will make it easier to perform analyses on each airline. After creating the array, I'll loop through it to make sure everything looks correct.

[8]:
```
airlines_array = df['AIRLINE'].unique()
for airline in airlines_array:
    print(airline)
```

```
AS
AA
US
DL
NK
UA
HA
B6
OO
EV
MQ
F9
WN
VX
```

Note that these airline values are actually IATA airline codes, which don't always resemble the airlines' names. For instance, NK corresponds to Spirit Airlines, and F9 corresponds to Frontier Airlines. Wikipedia has a list of which airline codes correspond to which airlines:

https://en.wikipedia.org/wiki/List_of_airline_codes

I now plan to evaluate the average arrival delays for each of these airlines. My goal will be to create a DataFrame (i.e. a table within Pandas) that contains four columns:

1. The airline code
2. The mean arrival delay (in minutes) for all of that airline's flights that had a delay greater than 0. (I only want to include flights with a delay greater than 0 because, as shown by the head() and tail() functions above, early arrivals are represented as negative arrival delays. Since those 'negative' delays offset the actual delays, I want to limit my analysis to delays with positive values. (This column will be labeled "mean_arrival_delay".) This value serves to answer the question: "Among all the airline's flights that had a delay, how bad was the delay on average?"
3. The proportion of the airline's flights that had a delay greater than 0. (This column will be labeled "proportion_delayed".)
4. The product of the above two columns, which will represent the expected value of an airline's flight delays (in minutes). For example, if the mean arrival delay is 30 minutes, and 0.5 (50%) of the airline's flights are delayed, this will equal an expected value of 15 minutes. This column will make it easier to compare delay times across airlines. (This column will be labeled "delay_expected_value".)

To build this DataFrame, I will first create an empty dictionary of dictionaries called "airline_arrival_delay_data". The keys of this dictionary will be airline codes, and the values will themselves be dictionaries. Each dictionary within airline_arrival_delay_data will have three key-value pairs that correspond to the data points in the above description (e.g. the mean arrival delay; the proportion of flights that were delayed; and the expected value of the delay).

For each airline, the code block below builds a dictionary (airline_dict); calculates the three values described above; and stores them into airline_dict under their respective keys. Once this process is complete, airline_dict itself is stored into airline_arrival_delay_data with the airline code as the key. At the end of the block, I print out the airline_arrival_delay_data dictionary.

[9]:
```
airline_arrival_delay_data = {}
for airline in airlines_array:
    airline_dict = {}
    mean_arrival_delay = df[(df.AIRLINE==airline) & (df.ARRIVAL_DELAY > 0)].
↪ARRIVAL_DELAY.mean()
    # The above line finds all rows within df, our original DataFrame, that (1)␣
↪correspond to the current airline within our for loop and (2) have an␣
↪arrival delay greater than 0. It then finds the mean arrival delay of all␣
↪these rows and stores it in the mean_arrival_delay variable.
    proportion_delayed = df[(df.AIRLINE==airline) & (df.ARRIVAL_DELAY > 0)].
↪AIRLINE.count()/df[(df.AIRLINE==airline)].AIRLINE.count()
    # The above line finds, for the current airline within our for loop, the␣
↪number of flights that had an arrival day greater than 0. It then divides␣
↪this number by the total number of flights and stores this proportion within␣
↪the proportion_delayed variable.
    delay_expected_value = mean_arrival_delay*proportion_delayed
```

```
    print("Average arrival delay (among delays greater than 0) for" , airline,␣
 ↪mean_arrival_delay) # Since this block of code takes a while to execute for␣
 ↪my computer, this print statement helps me keep track of the code's progress.
    # The following three lines of code store the variables we calculated above␣
 ↪within airline_dict.
    airline_dict ['mean_arrival_delay'] = mean_arrival_delay
    airline_dict ['proportion_delayed'] = proportion_delayed
    airline_dict ['delay_expected_value'] = delay_expected_value
    # Finally, we store this instance of airline_dict as a value within␣
 ↪airline_arrival_delay_data, with the current airline within our for loop as␣
 ↪the key.
    airline_arrival_delay_data[airline] = airline_dict
airline_arrival_delay_data # Prints out our dictionary of dictionaries so that␣
 ↪we can confirm the code ran correctly
```

```
Average arrival delay (among delays greater than 0) for AS 22.80629539951574
Average arrival delay (among delays greater than 0) for AA 34.612147110454806
Average arrival delay (among delays greater than 0) for US 27.41992528019925
Average arrival delay (among delays greater than 0) for DL 32.58149609147273
Average arrival delay (among delays greater than 0) for NK 41.44997530113615
Average arrival delay (among delays greater than 0) for UA 39.6580342355456
Average arrival delay (among delays greater than 0) for HA 15.51550082336937
Average arrival delay (among delays greater than 0) for B6 38.89763343242869
Average arrival delay (among delays greater than 0) for OO 33.00360595030398
Average arrival delay (among delays greater than 0) for EV 35.700212217674704
Average arrival delay (among delays greater than 0) for MQ 39.74337582390879
Average arrival delay (among delays greater than 0) for F9 42.062584014062665
Average arrival delay (among delays greater than 0) for WN 29.822739130829994
Average arrival delay (among delays greater than 0) for VX 30.85802913145118
```

[9]:
```
{'AS': {'mean_arrival_delay': 22.80629539951574,
  'proportion_delayed': 0.33185493565490276,
  'delay_expected_value': 7.568381692333001},
 'AA': {'mean_arrival_delay': 34.612147110454806,
  'proportion_delayed': 0.3538941319019445,
  'delay_expected_value': 12.2490357549168},
 'US': {'mean_arrival_delay': 27.41992528019925,
  'proportion_delayed': 0.38389150290617213,
  'delay_expected_value': 10.526276325390633},
 'DL': {'mean_arrival_delay': 32.58149609147273,
  'proportion_delayed': 0.29362799548685603,
  'delay_expected_value': 9.566839387301972},
 'NK': {'mean_arrival_delay': 41.44997530113615,
  'proportion_delayed': 0.4911216653758946,
  'delay_expected_value': 20.356980899683684},
```

```
 'UA': {'mean_arrival_delay': 39.6580342355456,
  'proportion_delayed': 0.3714053410921846,
  'delay_expected_value': 14.729205732298347},
 'HA': {'mean_arrival_delay': 15.51550082336937,
  'proportion_delayed': 0.39888619163215766,
  'delay_expected_value': 6.188919034699414},
 'B6': {'mean_arrival_delay': 38.89763343242869,
  'proportion_delayed': 0.3828461867950313,
  'delay_expected_value': 14.891810634956249},
 'OO': {'mean_arrival_delay': 33.00360595030398,
  'proportion_delayed': 0.3813787543207703,
  'delay_expected_value': 12.586874125420493},
 'EV': {'mean_arrival_delay': 35.700212217674704,
  'proportion_delayed': 0.3760767241362929,
  'delay_expected_value': 13.426018861793562},
 'MQ': {'mean_arrival_delay': 39.74337582390879,
  'proportion_delayed': 0.3622483036860444,
  'delay_expected_value': 14.396970474967906},
 'F9': {'mean_arrival_delay': 42.062584014062665,
  'proportion_delayed': 0.4675651175439657,
  'delay_expected_value': 19.666997038738142},
 'WN': {'mean_arrival_delay': 29.822739130829994,
  'proportion_delayed': 0.3802274009603064,
  'delay_expected_value': 11.339422589232715},
 'VX': {'mean_arrival_delay': 30.85802913145118,
  'proportion_delayed': 0.39412463013164656,
  'delay_expected_value': 12.16190931802477}}
```

Now that I have my dictionary of dictionaries, it's easy to convert it to a DataFrame using the Pandas function DataFrame.from_dict(). I'll also transpose the DataFrame so that the airlines show up as rows, not columns, and give the index the name "airline."

[10]:
```
df_airline_delay_data = pd.DataFrame.from_dict(airline_arrival_delay_data).
 ↪transpose()
df_airline_delay_data.rename_axis("airline",inplace=True)
df_airline_delay_data
```

[10]:

airline	mean_arrival_delay	proportion_delayed	delay_expected_value
AS	22.806295	0.331855	7.568382
AA	34.612147	0.353894	12.249036
US	27.419925	0.383892	10.526276
DL	32.581496	0.293628	9.566839
NK	41.449975	0.491122	20.356981
UA	39.658034	0.371405	14.729206
HA	15.515501	0.398886	6.188919
B6	38.897633	0.382846	14.891811
OO	33.003606	0.381379	12.586874

EV	35.700212	0.376077	13.426019
MQ	39.743376	0.362248	14.396970
F9	42.062584	0.467565	19.666997
WN	29.822739	0.380227	11.339423
VX	30.858029	0.394125	12.161909

Next, I'll sort the rows by delay_expected_value to determine which airlines have the longest delays, based on the metrics described above.

[11]:
```
df_airline_delay_data.
 ↪sort_values('delay_expected_value',ascending=False,inplace=True)
df_airline_delay_data
```

[11]:

airline	mean_arrival_delay	proportion_delayed	delay_expected_value
NK	41.449975	0.491122	20.356981
F9	42.062584	0.467565	19.666997
B6	38.897633	0.382846	14.891811
UA	39.658034	0.371405	14.729206
MQ	39.743376	0.362248	14.396970
EV	35.700212	0.376077	13.426019
OO	33.003606	0.381379	12.586874
AA	34.612147	0.353894	12.249036
VX	30.858029	0.394125	12.161909
WN	29.822739	0.380227	11.339423
US	27.419925	0.383892	10.526276
DL	32.581496	0.293628	9.566839
AS	22.806295	0.331855	7.568382
HA	15.515501	0.398886	6.188919

According to this table, Spirit Airlines (NK) and Frontier Airlines (F9) have the longest expected arrival delay values based on the flights in our dataset, and Hawaiian Airlines (HA) and Alaska Airlines (AS) have the shortest. These values are shaped by my decisions about how to calculate mean arrival delays, and the delay figures that actual industry analysts and government agencies use may be lower.

Finally, I will save this DataFrame (not the one I just created above) as a CSV file using the to_csv() function:

[12]:
```
df_airline_delay_data.to_csv('airline_delay_data.csv')
```

Given that there are multiple ways to interpret delays, I'll also compare the airlines in this dataset using a much simpler measure: their mean delay time for all flights, not only those with a delay greater than 0. The following code calculates this version of the mean delay time for each airline, then stores it within a dictionary. (This code will answer the question: "For each airline, what is the average difference between when a flight is supposed to arrive and when it actually arrives?" The larger the value, the higher the average delay.)

[13]:
```
simpler_delay_comparison = {}
for airline in airlines_array:
    mean_arrival_delay = df[(df.AIRLINE==airline)].ARRIVAL_DELAY.mean()
    simpler_delay_comparison[airline]=mean_arrival_delay
simpler_delay_comparison
```

[13]:
```
{'AS': -0.7914090113039325,
 'AA': 3.9352136660833175,
 'US': 3.7062088424131026,
 'DL': 0.6802647653485016,
 'NK': 15.210786320067772,
 'UA': 6.21131564254273,
 'HA': 2.161856334598582,
 'B6': 6.94933917038418,
 'OO': 6.221438197483382,
 'EV': 6.964550126466531,
 'MQ': 7.316540271458018,
 'F9': 13.729467715383208,
 'WN': 4.83783107128739,
 'VX': 4.979538817121459}
```

Next, I'll convert this single dictionary into a dataframe, which requires some extra code (e.g. orient='index' within the first line).

[14]:
```
simpler_delay_table = pd.DataFrame.
 ↪from_dict(simpler_delay_comparison,orient='index',columns=['mean_arrival_delay'])
simpler_delay_table.
 ↪sort_values('mean_arrival_delay',ascending=False,inplace=True)
df_airline_delay_data.rename_axis("airline",inplace=True)
simpler_delay_table
```

[14]:

	mean_arrival_delay
NK	15.210786
F9	13.729468
MQ	7.316540
EV	6.964550
B6	6.949339
OO	6.221438
UA	6.211316
VX	4.979539
WN	4.837831
AA	3.935214
US	3.706209
HA	2.161856
DL	0.680265
AS	-0.791409

This DataFrame's mean arrival delay values are quite lower than the previous one's due to the

inclusion of 0 and negative delay times in our calculation. However, Spirit and Frontier continue to have the two largest mean arrival delay times, and the same 3 airlines appear at the bottom of our list (albeit in a different order).

I am still very new to Python, but I'm impressed by how little code it takes to perform statistical analysis of giant dataframes. I look forward to continuing to learn about both Python and business analytics during my time at CBS.

–Kenneth Burchfiel

First edition of this notebook was completed on Feb. 12, 2021

This exercise was inspired by an assignment written by CBS professors Daniel Guetta and Hongseok Namkoong, with help from Ander Unterga del Orden, and was uploaded to GitHub with permission from Professor Namkoong. The dataset I used also came from this assignment.