Activity_Course 2 Automatidata project lab

March 23, 2024

1 Automatidata project

Course 2 - Get Started with Python

Welcome to the Automatidata Project!

You have just started as a data professional in a fictional data consulting firm, Automatidata. Their client, the New York City Taxi and Limousine Commission (New York City TLC), has hired the Automatidata team for its reputation in helping their clients develop data-based solutions.

The team is still in the early stages of the project. Previously, you were asked to complete a project proposal by your supervisor, DeShawn Washington. You have received notice that your project proposal has been approved and that New York City TLC has given the Automatidata team access to their data. To get clear insights, New York TLC's data must be analyzed, key variables identified, and the dataset ensured it is ready for analysis.

A notebook was structured and prepared to help you in this project. Please complete the following questions.

2 Course 2 End-of-course project: Inspect and analyze data

In this activity, you will examine data provided and prepare it for analysis. This activity will help ensure the information is,

1. Ready to answer questions and yield insights
2. Ready for visualizations
3. Ready for future hypothesis testing and statistical methods

The purpose of this project is to investigate and understand the data provided.

The goal is to use a dataframe contructed within Python, perform a cursory inspection of the provided dataset, and inform team members of your findings.
This activity has three parts:

Part 1: Understand the situation * Prepare to understand and organize the provided taxi cab dataset and information.

Part 2: Understand the data

- Create a pandas dataframe for data learning, future exploratory data analysis (EDA), and statistical activities.
- Compile summary information about the data to inform next steps.

Part 3: Understand the variables

- Use insights from your examination of the summary data to guide deeper investigation into specific variables.

Follow the instructions and answer the following questions to complete the activity. Then, you will complete an Executive Summary using the questions listed on the PACE Strategy Document.

Be sure to complete this activity before moving on. The next course item will provide you with a completed exemplar to compare to your own work.

3 Identify data types and relevant variables using Python

4 PACE stages

Throughout these project notebooks, you'll see references to the problem-solving framework PACE. The following notebook components are labeled with the respective PACE stage: Plan, Analyze, Construct, and Execute.

4.1 PACE: Plan

Consider the questions in your PACE Strategy Document and those below to craft your response:

4.1.1 Task 1. Understand the situation

- How can you best prepare to understand and organize the provided taxi cab information?

I will be understanding the data provided in the data dictionary. The first step is to read the information about the data and understand each column. Also keep in mind about the background information provided by the client. For this task, we need to load the data into python and inspect it.

4.2 PACE: Analyze

Consider the questions in your PACE Strategy Document to reflect on the Analyze stage.

4.2.1 Task 2a. Build dataframe

Create a pandas dataframe for data learning, and future exploratory data analysis (EDA) and statistical activities.

Code the following,

- import pandas as `pd`. pandas is used for buidling dataframes.
- import numpy as `np`. numpy is imported with pandas
- `df = pd.read_csv('Datasets\NYC taxi data.csv')`

Note: pair the data object name `df` with pandas functions to manipulate data, such as `df.groupby()`.

Note: As shown in this cell, the dataset has been automatically loaded in for you. You do not need to download the .csv file, or provide more code, in order to access the dataset and proceed with this lab. Please continue with this activity by completing the following instructions.

[1]:
```
#Import libraries and packages listed above
### YOUR CODE HERE ###
import pandas as pd
import numpy as np
# Load dataset into dataframe
df = pd.read_csv('2017_Yellow_Taxi_Trip_Data.csv')
print("done")
```

```
done
```

4.2.2 Task 2b. Understand the data - Inspect the data

View and inspect summary information about the dataframe by coding the following:

1. `df.head(10)`
2. `df.info()`
3. `df.describe()`

Consider the following two questions:

Question 1: When reviewing the `df.info()` output, what do you notice about the different variables? Are there any null values? Are all of the variables numeric? Does anything else stand out?

Question 2: When reviewing the `df.describe()` output, what do you notice about the distributions of each variable? Are there any questionable values?

1. There are 22699 entries, 18 columns in the dataframe. There are 3 data types - Float, Int, Object(datetime). There are no null values.
2. Comparing ro the 25-75 percent range of values, the maximum fare amount is a very big value($1000). There are negative values for fare amount(price is always positive). The maximum trip distance is 33 miles whereas most of the rides are in the range of 1-3 miles.

[2]:
```
#==> ENTER YOUR CODE HERE
df.head(10)
```

[2]:

	Unnamed: 0	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	\
0	24870114	2	03/25/2017 8:55:43 AM	03/25/2017 9:09:47 AM	
1	35634249	1	04/11/2017 2:53:28 PM	04/11/2017 3:19:58 PM	

```
2   106203690         1   12/15/2017 7:26:56 AM   12/15/2017 7:34:08 AM
3    38942136         2   05/07/2017 1:17:59 PM   05/07/2017 1:48:14 PM
4    30841670         2  04/15/2017 11:32:20 PM  04/15/2017 11:49:03 PM
5    23345809         2   03/25/2017 8:34:11 PM   03/25/2017 8:42:11 PM
6    37660487         2   05/03/2017 7:04:09 PM   05/03/2017 8:03:47 PM
7    69059411         2   08/15/2017 5:41:06 PM   08/15/2017 6:03:05 PM
8     8433159         2   02/04/2017 4:17:07 PM   02/04/2017 4:29:14 PM
9    95294817         1   11/10/2017 3:20:29 PM   11/10/2017 3:40:55 PM

   passenger_count  trip_distance  RatecodeID store_and_fwd_flag  \
0                6           3.34           1                  N
1                1           1.80           1                  N
2                1           1.00           1                  N
3                1           3.70           1                  N
4                1           4.37           1                  N
5                6           2.30           1                  N
6                1          12.83           1                  N
7                1           2.98           1                  N
8                1           1.20           1                  N
9                1           1.60           1                  N

   PULocationID  DOLocationID  payment_type  fare_amount  extra  mta_tax  \
0           100           231             1         13.0    0.0      0.5
1           186            43             1         16.0    0.0      0.5
2           262           236             1          6.5    0.0      0.5
3           188            97             1         20.5    0.0      0.5
4             4           112             2         16.5    0.5      0.5
5           161           236             1          9.0    0.5      0.5
6            79           241             1         47.5    1.0      0.5
7           237           114             1         16.0    1.0      0.5
8           234           249             2          9.0    0.0      0.5
9           239           237             1         13.0    0.0      0.5

   tip_amount  tolls_amount  improvement_surcharge  total_amount
0        2.76           0.0                    0.3         16.56
1        4.00           0.0                    0.3         20.80
2        1.45           0.0                    0.3          8.75
3        6.39           0.0                    0.3         27.69
4        0.00           0.0                    0.3         17.80
5        2.06           0.0                    0.3         12.36
6        9.86           0.0                    0.3         59.16
7        1.78           0.0                    0.3         19.58
8        0.00           0.0                    0.3          9.80
9        2.75           0.0                    0.3         16.55
```

[3]:
```
#==> ENTER YOUR CODE HERE
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22699 entries, 0 to 22698
Data columns (total 18 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Unnamed: 0             22699 non-null  int64
 1   VendorID               22699 non-null  int64
 2   tpep_pickup_datetime   22699 non-null  object
 3   tpep_dropoff_datetime  22699 non-null  object
 4   passenger_count        22699 non-null  int64
 5   trip_distance          22699 non-null  float64
 6   RatecodeID             22699 non-null  int64
 7   store_and_fwd_flag     22699 non-null  object
 8   PULocationID           22699 non-null  int64
 9   DOLocationID           22699 non-null  int64
 10  payment_type           22699 non-null  int64
 11  fare_amount            22699 non-null  float64
 12  extra                  22699 non-null  float64
 13  mta_tax                22699 non-null  float64
 14  tip_amount             22699 non-null  float64
 15  tolls_amount           22699 non-null  float64
 16  improvement_surcharge  22699 non-null  float64
 17  total_amount           22699 non-null  float64
dtypes: float64(8), int64(7), object(3)
memory usage: 3.1+ MB
```

[4]:
```
#==> ENTER YOUR CODE HERE
df.describe()
```

[4]:

	Unnamed: 0	VendorID	passenger_count	trip_distance	\
count	2.269900e+04	22699.000000	22699.000000	22699.000000	
mean	5.675849e+07	1.556236	1.642319	2.913313	
std	3.274493e+07	0.496838	1.285231	3.653171	
min	1.212700e+04	1.000000	0.000000	0.000000	
25%	2.852056e+07	1.000000	1.000000	0.990000	
50%	5.673150e+07	2.000000	1.000000	1.610000	
75%	8.537452e+07	2.000000	2.000000	3.060000	
max	1.134863e+08	2.000000	6.000000	33.960000	

	RatecodeID	PULocationID	DOLocationID	payment_type	fare_amount	\
count	22699.000000	22699.000000	22699.000000	22699.000000	22699.000000	
mean	1.043394	162.412353	161.527997	1.336887	13.026629	
std	0.708391	66.633373	70.139691	0.496211	13.243791	
min	1.000000	1.000000	1.000000	1.000000	-120.000000	
25%	1.000000	114.000000	112.000000	1.000000	6.500000	
50%	1.000000	162.000000	162.000000	1.000000	9.500000	
75%	1.000000	233.000000	233.000000	2.000000	14.500000	

```
max       99.000000    265.000000    265.000000      4.000000    999.990000

              extra       mta_tax    tip_amount  tolls_amount  \
count  22699.000000  22699.000000  22699.000000  22699.000000
mean       0.333275      0.497445      1.835781      0.312542
std        0.463097      0.039465      2.800626      1.399212
min       -1.000000     -0.500000      0.000000      0.000000
25%        0.000000      0.500000      0.000000      0.000000
50%        0.000000      0.500000      1.350000      0.000000
75%        0.500000      0.500000      2.450000      0.000000
max        4.500000      0.500000    200.000000     19.100000

       improvement_surcharge  total_amount
count           22699.000000  22699.000000
mean                0.299551     16.310502
std                 0.015673     16.097295
min                -0.300000   -120.300000
25%                 0.300000      8.750000
50%                 0.300000     11.800000
75%                 0.300000     17.800000
max                 0.300000   1200.290000
```

4.2.3 Task 2c. Understand the data - Investigate the variables

Sort and interpret the data table for two variables:`trip_distance` and `total_amount`.

Answer the following three questions:

Question 1: Sort your first variable (`trip_distance`) from maximum to minimum value, do the values seem normal? The values are normal.

Question 2: Sort by your second variable (`total_amount`), are any values unusual? The values are normal. the highest total_amount is 1200 whereas the remaining all the amounts are less than or equal to 120 which is giving us a huge difference.

Question 3: Are the resulting rows similar for both sorts? Why or why not?

==> ENTER YOUR RESPONSES TO QUESTION 1-3 HERE

[11]:
```
# ==> ENTER YOUR CODE HERE

# Sort the data by trip distance from maximum to minimum value
sor_tripdistance = df.sort_values(by='trip_distance', ascending=False)
```

[13]:
```
#==> ENTER YOUR CODE HERE

# Sort the data by total amount and print the top 20 values
sort_totalamount = df.sort_values(by='total_amount', ascending=False)
sort_totalamount.tail(20)
```

[13]:

	Unnamed: 0	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime
14283	37675840	1	05/03/2017 7:44:28 PM	05/03/2017 7:44:38 PM
19067	58713019	1	07/10/2017 2:40:09 PM	07/10/2017 2:40:59 PM
10506	26005024	2	03/30/2017 3:14:26 AM	03/30/2017 3:14:28 AM
5722	49670364	2	06/12/2017 12:08:55 PM	06/12/2017 12:08:57 PM
4402	108016954	2	12/20/2017 4:06:53 PM	12/20/2017 4:47:50 PM
22566	19022898	2	03/07/2017 2:24:47 AM	03/07/2017 2:24:50 AM
1646	57337183	2	07/05/2017 11:02:23 AM	07/05/2017 11:03:00 AM
18565	43859760	2	05/22/2017 3:51:20 PM	05/22/2017 3:52:22 PM
314	105454287	2	12/13/2017 2:02:39 AM	12/13/2017 2:03:08 AM
5758	833948	2	01/03/2017 8:15:23 PM	01/03/2017 8:15:39 PM
5448	28459983	2	04/06/2017 12:50:26 PM	04/06/2017 12:52:39 PM
4423	97329905	2	11/16/2017 8:13:30 PM	11/16/2017 8:14:50 PM
10281	55302347	2	06/05/2017 5:34:25 PM	06/05/2017 5:36:29 PM
8204	91187947	2	10/28/2017 8:39:36 PM	10/28/2017 8:41:59 PM
20317	75926915	2	09/09/2017 10:59:51 PM	09/09/2017 11:02:06 PM
11204	58395501	2	07/09/2017 7:20:59 AM	07/09/2017 7:23:50 AM
14714	109276092	2	12/24/2017 10:37:58 PM	12/24/2017 10:41:08 PM
17602	24690146	2	03/24/2017 7:31:13 PM	03/24/2017 7:34:49 PM
20698	14668209	2	02/24/2017 12:38:17 AM	02/24/2017 12:42:05 AM
12944	29059760	2	04/08/2017 12:00:16 AM	04/08/2017 11:15:57 PM

	passenger_count	trip_distance	RatecodeID	store_and_fwd_flag
14283	1	0.00	5	N
19067	1	0.10	5	N
10506	1	0.00	1	N
5722	1	0.00	1	N
4402	1	7.06	1	N
22566	1	0.00	1	N
1646	1	0.04	1	N
18565	1	0.10	1	N
314	6	0.12	1	N
5758	1	0.02	1	N
5448	1	0.25	1	N
4423	2	0.06	1	N
10281	2	0.00	1	N
8204	1	0.41	1	N
20317	1	0.24	1	N
11204	1	0.64	1	N
14714	5	0.40	1	N
17602	1	0.46	1	N
20698	1	0.70	1	N
12944	1	0.17	5	N

	PULocationID	DOLocationID	payment_type	fare_amount	extra	mta_tax
14283	146	146	3	0.01	0.0	0.0
19067	261	13	3	0.00	0.0	0.0

10506	264	193	1	0.00	0.0	0.0
5722	264	193	1	0.00	0.0	0.0
4402	263	169	2	0.00	0.0	0.0
22566	264	193	1	0.00	0.0	0.0
1646	79	79	3	-2.50	0.0	-0.5
18565	230	163	3	-3.00	0.0	-0.5
314	161	161	3	-2.50	-0.5	-0.5
5758	170	170	3	-2.50	-0.5	-0.5
5448	90	68	3	-3.50	0.0	-0.5
4423	237	237	4	-3.00	-0.5	-0.5
10281	238	238	4	-2.50	-1.0	-0.5
8204	236	237	3	-3.50	-0.5	-0.5
20317	116	116	4	-3.50	-0.5	-0.5
11204	50	48	3	-4.50	0.0	-0.5
14714	164	161	4	-4.00	-0.5	-0.5
17602	87	45	4	-4.00	-1.0	-0.5
20698	65	25	4	-4.50	-0.5	-0.5
12944	138	138	4	-120.00	0.0	0.0

	tip_amount	tolls_amount	improvement_surcharge	total_amount
14283	0.0	0.0	0.3	0.31
19067	0.0	0.0	0.3	0.30
10506	0.0	0.0	0.0	0.00
5722	0.0	0.0	0.0	0.00
4402	0.0	0.0	0.0	0.00
22566	0.0	0.0	0.0	0.00
1646	0.0	0.0	-0.3	-3.30
18565	0.0	0.0	-0.3	-3.80
314	0.0	0.0	-0.3	-3.80
5758	0.0	0.0	-0.3	-3.80
5448	0.0	0.0	-0.3	-4.30
4423	0.0	0.0	-0.3	-4.30
10281	0.0	0.0	-0.3	-4.30
8204	0.0	0.0	-0.3	-4.80
20317	0.0	0.0	-0.3	-4.80
11204	0.0	0.0	-0.3	-5.30
14714	0.0	0.0	-0.3	-5.30
17602	0.0	0.0	-0.3	-5.80
20698	0.0	0.0	-0.3	-5.80
12944	0.0	0.0	-0.3	-120.30

[12]:
```
#==> ENTER YOUR CODE HERE

# Sort the data by total amount and print the bottom 20 values
sort_totalamount.tail(20)
```

[12]:

	Unnamed: 0	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime
14283	37675840	1	05/03/2017 7:44:28 PM	05/03/2017 7:44:38 PM
19067	58713019	1	07/10/2017 2:40:09 PM	07/10/2017 2:40:59 PM
10506	26005024	2	03/30/2017 3:14:26 AM	03/30/2017 3:14:28 AM
5722	49670364	2	06/12/2017 12:08:55 PM	06/12/2017 12:08:57 PM
4402	108016954	2	12/20/2017 4:06:53 PM	12/20/2017 4:47:50 PM
22566	19022898	2	03/07/2017 2:24:47 AM	03/07/2017 2:24:50 AM
1646	57337183	2	07/05/2017 11:02:23 AM	07/05/2017 11:03:00 AM
18565	43859760	2	05/22/2017 3:51:20 PM	05/22/2017 3:52:22 PM
314	105454287	2	12/13/2017 2:02:39 AM	12/13/2017 2:03:08 AM
5758	833948	2	01/03/2017 8:15:23 PM	01/03/2017 8:15:39 PM
5448	28459983	2	04/06/2017 12:50:26 PM	04/06/2017 12:52:39 PM
4423	97329905	2	11/16/2017 8:13:30 PM	11/16/2017 8:14:50 PM
10281	55302347	2	06/05/2017 5:34:25 PM	06/05/2017 5:36:29 PM
8204	91187947	2	10/28/2017 8:39:36 PM	10/28/2017 8:41:59 PM
20317	75926915	2	09/09/2017 10:59:51 PM	09/09/2017 11:02:06 PM
11204	58395501	2	07/09/2017 7:20:59 AM	07/09/2017 7:23:50 AM
14714	109276092	2	12/24/2017 10:37:58 PM	12/24/2017 10:41:08 PM
17602	24690146	2	03/24/2017 7:31:13 PM	03/24/2017 7:34:49 PM
20698	14668209	2	02/24/2017 12:38:17 AM	02/24/2017 12:42:05 AM
12944	29059760	2	04/08/2017 12:00:16 AM	04/08/2017 11:15:57 PM

	passenger_count	trip_distance	RatecodeID	store_and_fwd_flag
14283	1	0.00	5	N
19067	1	0.10	5	N
10506	1	0.00	1	N
5722	1	0.00	1	N
4402	1	7.06	1	N
22566	1	0.00	1	N
1646	1	0.04	1	N
18565	1	0.10	1	N
314	6	0.12	1	N
5758	1	0.02	1	N
5448	1	0.25	1	N
4423	2	0.06	1	N
10281	2	0.00	1	N
8204	1	0.41	1	N
20317	1	0.24	1	N
11204	1	0.64	1	N
14714	5	0.40	1	N
17602	1	0.46	1	N
20698	1	0.70	1	N
12944	1	0.17	5	N

	PULocationID	DOLocationID	payment_type	fare_amount	extra	mta_tax
14283	146	146	3	0.01	0.0	0.0
19067	261	13	3	0.00	0.0	0.0

10506	264	193	1	0.00	0.0	0.0
5722	264	193	1	0.00	0.0	0.0
4402	263	169	2	0.00	0.0	0.0
22566	264	193	1	0.00	0.0	0.0
1646	79	79	3	-2.50	0.0	-0.5
18565	230	163	3	-3.00	0.0	-0.5
314	161	161	3	-2.50	-0.5	-0.5
5758	170	170	3	-2.50	-0.5	-0.5
5448	90	68	3	-3.50	0.0	-0.5
4423	237	237	4	-3.00	-0.5	-0.5
10281	238	238	4	-2.50	-1.0	-0.5
8204	236	237	3	-3.50	-0.5	-0.5
20317	116	116	4	-3.50	-0.5	-0.5
11204	50	48	3	-4.50	0.0	-0.5
14714	164	161	4	-4.00	-0.5	-0.5
17602	87	45	4	-4.00	-1.0	-0.5
20698	65	25	4	-4.50	-0.5	-0.5
12944	138	138	4	-120.00	0.0	0.0

	tip_amount	tolls_amount	improvement_surcharge	total_amount
14283	0.0	0.0	0.3	0.31
19067	0.0	0.0	0.3	0.30
10506	0.0	0.0	0.0	0.00
5722	0.0	0.0	0.0	0.00
4402	0.0	0.0	0.0	0.00
22566	0.0	0.0	0.0	0.00
1646	0.0	0.0	-0.3	-3.30
18565	0.0	0.0	-0.3	-3.80
314	0.0	0.0	-0.3	-3.80
5758	0.0	0.0	-0.3	-3.80
5448	0.0	0.0	-0.3	-4.30
4423	0.0	0.0	-0.3	-4.30
10281	0.0	0.0	-0.3	-4.30
8204	0.0	0.0	-0.3	-4.80
20317	0.0	0.0	-0.3	-4.80
11204	0.0	0.0	-0.3	-5.30
14714	0.0	0.0	-0.3	-5.30
17602	0.0	0.0	-0.3	-5.80
20698	0.0	0.0	-0.3	-5.80
12944	0.0	0.0	-0.3	-120.30

[14]:
```
#==> ENTER YOUR CODE HERE

# How many of each payment type are represented in the data?
df['payment_type'].value_counts()
```

[14]:
```
1    15265
2     7267
3      121
4       46
Name: payment_type, dtype: int64
```

According to the data dictionary, the payment method was encoded as follows:

1 = Credit card
2 = Cash
3 = No charge
4 = Dispute
5 = Unknown
6 = Voided trip

[19]:
```
#==> ENTER YOUR CODE HERE

# What is the average tip for trips paid for with credit card?
Tipamount_creditcard_df = df[df['payment_type'] == 1]
averagetip_creditcard = Tipamount_creditcard_df['tip_amount'].mean()
print(averagetip_creditcard)

#==> ENTER YOUR CODE HERE

# What is the average tip for trips paid for with cash?
Tipamount_cash_df = df[df['payment_type'] == 2]
averagetip_cash = Tipamount_cash_df['tip_amount'].mean()
print(averagetip_cash)
```

```
2.7298001965279934
0.0
```

[20]:
```
#==> ENTER YOUR CODE HERE

# How many times is each vendor ID represented in the data?
df['VendorID'].value_counts()
```

[20]:
```
2    12626
1    10073
Name: VendorID, dtype: int64
```

[21]:
```
#==> ENTER YOUR CODE HERE

# What is the mean total amount for each vendor?
df.groupby(['VendorID']).mean(numeric_only=True)[['total_amount']]
```

[21]:
```
           total_amount
VendorID
```

```
1            16.298119
2            16.320382
```

[24]:
```
#==> ENTER YOUR CODE HERE

# Filter the data for credit card payments only
Filtered_credit_card = df[df['payment_type']==1]

#==> ENTER YOUR CODE HERE

# Filter the credit-card-only data for passenger count only
Filtered_credit_card['passenger_count'].value_counts()
```

[24]:
```
1    10977
2     2168
5      775
3      600
6      451
4      267
0       27
Name: passenger_count, dtype: int64
```

[27]:
```
#==> ENTER YOUR CODE HERE

# Calculate the average tip amount for each passenger count (credit card␣
 ↪payments only)
Filtered_credit_card.groupby(['passenger_count']).
 ↪mean(numeric_only=True)[['tip_amount']]
```

[27]:
```
                 tip_amount
passenger_count
0                  2.610370
1                  2.714681
2                  2.829949
3                  2.726800
4                  2.607753
5                  2.762645
6                  2.643326
```

4.3 PACE: Construct

Note: The Construct stage does not apply to this workflow. The PACE framework can be adapted to fit the specific requirements of any project.

4.4 PACE: Execute

Consider the questions in your PACE Strategy Document and those below to craft your response.

4.4.1 Given your efforts, what can you summarize for DeShawn and the data team?

Note for Learners: Your notebook should contain data that can address Luana's requests. Which two variables are most helpful for building a predictive model for the client: NYC TLC?

total_amount and trip_distance are the two variables that are most helpful for building a predictive model for the client: NYC TLC.

Congratulations! You've completed this lab. However, you may not notice a green check mark next to this item on Coursera's platform. Please continue your progress regardless of the check mark. Just click on the "save" icon at the top of this notebook to ensure your work has been logged.