Activity_Course 2 Automatidata project lab

April 18, 2025

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?

To best prepare to understand and organize the provided taxi cab information, I should take the following steps:

Review the Project Brief: Carefully reread the project description and goals to ensure a clear understanding of the objectives and expected outcomes.

Examine the PACE Strategy Document: Refer to the PACE strategy document for guidance on planning, analysis, construction, and execution. This will provide a structured approach to the project.

Understand the Data Source: Recognize that the data comes from the New York City Taxi and Limousine Commission (New York City TLC).

This context is important for understanding the potential variables and their significance. Identify the Goal: The primary goal is to investigate and understand the provided dataset to determine its readiness for analysis, visualizations, hypothesis testing, and statistical methods.

Outline the Steps: Based on the project description, the activity has three parts:

Part 1: Understand the situation (this step). Part 2: Understand the data (creating a Pandas DataFrame and compiling summary information). Part 3: Understand the variables (deeper investigation into specific variables based on summary data). Prepare the Environment: Ensure that the necessary Python libraries, particularly pandas, are available and ready to be used.

Anticipate Data Characteristics: Based on the source (taxi trip data), I can anticipate potential data points such as pickup/dropoff times and locations, passenger count, trip distance, fare information, payment types, and potentially vehicle or driver identifiers.

Formulate Initial Questions: Consider initial questions about the data, such as: What is the time period covered by the data? What are the key variables present in the dataset? Are there any obvious data quality issues to be aware of? What kind of insights might this data provide to the New York City TLC?

By taking these preparatory steps, I can establish a solid foundation for effectively analyzing and organizing the taxi cab data.

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.

[10]:
```
import pandas as pd                #library exercise for buidling dataframes
import numpy as np                 #numpy is imported with pandas

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?

==> ENTER YOUR RESPONSE TO QUESTIONS 1 & 2 HERE

[11]:
```
df.head(10)
```

[11]:
```
   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
```

[12]:
```python
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
```

[13]:
```
df.describe()
```

[13]:
```
         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?

Question 2: Sort by your second variable (`total_amount`), are any values unusual?

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

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

[14]:
```python
# Sort trip_distance from highest to lowest
df_sorted_distance = df.sort_values(by='trip_distance', ascending=False)
df_sorted_distance[['trip_distance', 'total_amount']].head(10)
```

[14]:

	trip_distance	total_amount
9280	33.96	150.30
13861	33.92	258.21
6064	32.72	179.06
10291	31.95	131.80
29	30.83	111.38
18130	30.50	119.31
5792	30.33	73.20
15350	28.23	62.96
10302	28.20	70.27
2592	27.97	63.06

[15]:
```python
# Sort total_amount from highest to lowest
df_sorted_total = df.sort_values(by='total_amount', ascending=False)
df_sorted_total[['trip_distance', 'total_amount']].head(10)
```

[15]:

	trip_distance	total_amount
8476	2.60	1200.29
20312	0.00	450.30
13861	33.92	258.21
12511	0.00	233.74
15474	0.00	211.80
6064	32.72	179.06
16379	25.50	157.06
3582	7.30	152.30
11269	0.00	151.82
9280	33.96	150.30

[16]:
```python
# Compare top rows of both sorts
df_sorted_distance.head(5)
df_sorted_total.head(5)
```

[16]:

	Unnamed: 0	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	\
8476	11157412	1	02/06/2017 5:50:10 AM	02/06/2017 5:51:08 AM	
20312	107558404	2	12/19/2017 9:40:46 AM	12/19/2017 9:40:55 AM	
13861	40523668	2	05/19/2017 8:20:21 AM	05/19/2017 9:20:30 AM	

```
12511   107108848         2  12/17/2017 6:24:24 PM  12/17/2017 6:24:42 PM
15474    55538852         2  06/06/2017 8:55:01 PM  06/06/2017 8:55:06 PM

       passenger_count  trip_distance  RatecodeID store_and_fwd_flag  \
8476                 1           2.60           5                  N
20312                2           0.00           5                  N
13861                1          33.92           5                  N
12511                1           0.00           5                  N
15474                1           0.00           5                  N

       PULocationID  DOLocationID  payment_type  fare_amount  extra  mta_tax  \
8476            226           226             1       999.99    0.0      0.0
20312           265           265             2       450.00    0.0      0.0
13861           229           265             1       200.01    0.0      0.5
12511           265           265             1       175.00    0.0      0.0
15474           265           265             1       200.00    0.0      0.5

       tip_amount  tolls_amount  improvement_surcharge  total_amount
8476       200.00          0.00                    0.3       1200.29
20312        0.00          0.00                    0.3        450.30
13861       51.64          5.76                    0.3        258.21
12511       46.69         11.75                    0.3        233.74
15474       11.00          0.00                    0.3        211.80
```

[17]:

```
df['payment_type'].value_counts()
```

[17]:

```
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

[18]:

```
# First, filter the dataset by payment type
# 1 = Credit card, 2 = Cash

# Average tip for credit card payments
avg_tip_credit = df[df['payment_type'] == 1]['tip_amount'].mean()
print(f"Average tip (Credit Card): ${avg_tip_credit:.2f}")
```

```python
# Average tip for cash payments
avg_tip_cash = df[df['payment_type'] == 2]['tip_amount'].mean()
print(f"Average tip (Cash): ${avg_tip_cash:.2f}")
```

```
Average tip (Credit Card): $2.73
Average tip (Cash): $0.00
```

[19]:
```python
vendor_counts = df['VendorID'].value_counts()
print("Vendor ID counts:\n")
print(vendor_counts)
```

```
Vendor ID counts:

2    12626
1    10073
Name: VendorID, dtype: int64
```

[20]:
```python
# Group by VendorID and calculate the mean total_amount
mean_total_by_vendor = df.groupby('VendorID')['total_amount'].mean()

print("Mean total amount for each VendorID:")
print(mean_total_by_vendor)
```

```
Mean total amount for each VendorID:
VendorID
1    16.298119
2    16.320382
Name: total_amount, dtype: float64
```

[25]:
```python
# Filter the data for credit card payments only
credit_card = df[df['payment_type']==1]

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

[25]:
```
1    10977
2     2168
5      775
3      600
6      451
4      267
0       27
Name: passenger_count, dtype: int64
```

[26]:
```python
# Filter for credit card payments
df_credit = df[df['payment_type'] == 1]
```

```
# Group by passenger count and calculate the mean tip_amount
avg_tip_by_passenger_count = df_credit.groupby('passenger_count')['tip_amount'].
 ↪mean()

print("Average tip amount for each passenger count (credit card payments only):
 ↪")
print(avg_tip_by_passenger_count)
```

```
Average tip amount for each passenger count (credit card payments only):
passenger_count
0    2.610370
1    2.714681
2    2.829949
3    2.726800
4    2.607753
5    2.762645
6    2.643326
Name: tip_amount, dtype: float64
```

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?

Summary for DeShawn and the Data Team: Key Insights: Data Overview:

The dataset provides details about NYC Yellow Taxi trips, including variables like trip_distance, total_amount, payment_type, passenger_count, tip_amount, and VendorID.

The data also includes several important columns like pickup_datetime, dropoff_datetime, and fare_amount, which could be valuable for further analysis or predictive modeling.

Credit Card vs. Cash Payments:

We calculated the average tip for credit card payments and cash payments, revealing how tips may differ based on payment method. This could be valuable for improving customer experience or operational decisions.

Credit card payments tend to have a higher average tip than cash payments, which might indicate more consistent or higher tips from digital payments.

VendorID Insights:

By analyzing the VendorID, we can see how often each vendor processes trips and the corresponding average total fare. This information could help identify trends or discrepancies between different vendors in terms of pricing or service quality.

Trip Distance & Total Amount:

We found that the trip_distance and total_amount columns could offer key insights into fare pricing models. Sorting these values gave us a better understanding of the range and variability of trips, identifying outliers (extremely high or low values).

Passenger Count and Tips:

We grouped data by passenger_count for credit card payments, showing how average tips vary with different group sizes. This could be useful for predicting tip amounts in future trips, especially if combined with passenger-related features like group size.

Most Helpful Variables for Building a Predictive Model: For the purpose of building a predictive model for NYC TLC (Taxi and Limousine Commission), the two most useful variables for predicting total fare (or any related output, such as tip amount or fare revenue) are:

trip_distance:

Why: It is directly correlated with the fare calculation (the longer the trip, the higher the fare). This will be a key variable for predicting the total amount a passenger will pay.

passenger_count:

Why: This could help predict variations in fare and tip amounts based on group size. It might also help capture the dynamic pricing model for group rides versus solo trips, and could affect both the fare amount and the tip amount.

Further Exploration for DeShawn: Payment Type Analysis: Since the dataset includes different payment types, investigating how payment method influences other variables (e.g., tip amount or vendor choice) could provide additional insights for the predictive model.

Time-based Features: Incorporating pickup and drop-off times could help predict the fare during peak and off-peak hours, which is crucial for demand forecasting.

Location Data: If available, pickup and drop-off locations could be integrated to refine the model, as some areas have higher demand or longer trip durations.

Conclusion: The combination of trip distance and passenger count will likely be the most significant variables for predicting the total fare or other related metrics in future taxi rides. By focusing on these, the predictive model can be better tailored to provide accurate estimations, optimize operations, and support strategic decision-making for 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.