Activity_ Course 5 Automatidata project lab

May 10, 2023

1 Automatidata project

Course 5 - Regression Analysis: Simplify complex data relationships

The data consulting firm Automatidata has recently hired you as the newest member of their data analytics team. Their newest client, the NYC Taxi and Limousine Commission (New York City TLC), wants the Automatidata team to build a multiple linear regression model for ride durations based on a variety of variables. The team is getting closer to completing the project, having completed an initial plan of action, initial Python coding work, EDA, and A/B testing.

The Automatidata team has reviewed the results of the A/B testing. Now it's time to work on predicting the taxi/ride share trip durations. You've impressed your Automatidata colleagues with your hard work and attention to detail. The data team believes that you are ready to build the regression model and update the client New York City TLC about your progress.

2 Course 5 End-of-course project: Build a multiple linear regression model

In this activity, you will build a multiple linear regression model

The purpose of this project is to demostrate knowledge of EDA and a multiple linear regression model

The goal is to build a multiple linear regression model and evaluate the model *This activity has three parts:*

Part 1: EDA & Checking Model Assumptions * What are some purposes of EDA before constructing a multiple linear regression model?

Part 2: Model Building and evaluation * What resources do you find yourself using as you complete this stage?

Part 3: Interpreting Model Results

- What key insights emerged from your model(s)?
- What business recommendations do you propose based on the models built?

Recall that you have a helpful tool at your disposal! Refer to the PACE strategy document to help apply your learning, apply new problem-solving skills, and guide your approach to this project.

3 Build a multiple linear regression model

4 PACE stages

- [Plan](#scrollTo=psz51YkZVwtN&line=3&uniqifier=1)
- [Analyze](#scrollTo=mA7Mz_SnI8km&line=4&uniqifier=1)
- [Construct](#scrollTo=Lca9c8XON8lc&line=2&uniqifier=1)
- [Execute](#scrollTo=401PgchTPr4E&line=2&uniqifier=1)

5 Pace: Plan

5.1 Task 1. Imports and loading

Import the packages that you've learned are needed for building linear regression models.

[89]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from datetime import datetime
from datetime import date
from datetime import timedelta


from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import sklearn.metrics as metrics
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error,r2_score,mean_squared_error
lr=LinearRegression()
```

Pandas is used to load the NYC TLC dataset. 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.

[90]:
```
# Read in data from NYC TLC dataset provided and load into dataframe
df=pd.read_csv("2017_Yellow_Taxi_Trip_Data.csv") # index_col parameter␣
↪specified to avoid "Unnamed: 0" column when reading in data from csv
print('Data loaded')
```

```
Data loaded
```

6 PACE: Analyze

In this stage, consider the following question where applicable to complete your code response:

- What are some purposes of EDA before constructing a multiple linear regression model?

Improve the performance and accuracy of the model while reducing the possibility of bias in the distribution of the dataset.

6.1 Task 2a. Explore data with EDA

Analyze and discover data, looking for correlations, missing data, outliers, and duplicates.

Start with `.shape` and `.info()`.

[91]:
```
print(df.shape)
print(df.info())
```

```
(22699, 18)
<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
None
```

Use `.head()`.

[92]:
```
df.head()
```

```
[92]:    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

         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

         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

         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
```

Create trip_duration.

```
[93]: df["drop_off_converted"] = pd.to_datetime(df["tpep_dropoff_datetime"],␣
       ↪format="%m/%d/%Y %I:%M:%S %p")

      df["pick_up_converted"] = pd.to_datetime(df["tpep_pickup_datetime"], format="%m/
       ↪%d/%Y %I:%M:%S %p")

      df['trip_duration']=(df['drop_off_converted']-df['pick_up_converted'])/np.
       ↪timedelta64(1,"m")
      df.head(10)
```

```
[93]:    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  …  fare_amount  extra  mta_tax  tip_amount  \
0           100           231  …         13.0    0.0      0.5        2.76
1           186            43  …         16.0    0.0      0.5        4.00
2           262           236  …          6.5    0.0      0.5        1.45
3           188            97  …         20.5    0.0      0.5        6.39
4             4           112  …         16.5    0.5      0.5        0.00
5           161           236  …          9.0    0.5      0.5        2.06
6            79           241  …         47.5    1.0      0.5        9.86
7           237           114  …         16.0    1.0      0.5        1.78
8           234           249  …          9.0    0.0      0.5        0.00
9           239           237  …         13.0    0.0      0.5        2.75

   tolls_amount  improvement_surcharge  total_amount  drop_off_converted  \
0           0.0                    0.3         16.56 2017-03-25 09:09:47
1           0.0                    0.3         20.80 2017-04-11 15:19:58
2           0.0                    0.3          8.75 2017-12-15 07:34:08
3           0.0                    0.3         27.69 2017-05-07 13:48:14
4           0.0                    0.3         17.80 2017-04-15 23:49:03
5           0.0                    0.3         12.36 2017-03-25 20:42:11
6           0.0                    0.3         59.16 2017-05-03 20:03:47
7           0.0                    0.3         19.58 2017-08-15 18:03:05
8           0.0                    0.3          9.80 2017-02-04 16:29:14
9           0.0                    0.3         16.55 2017-11-10 15:40:55

    pick_up_converted trip_duration
0 2017-03-25 08:55:43     14.066667
1 2017-04-11 14:53:28     26.500000
2 2017-12-15 07:26:56      7.200000
3 2017-05-07 13:17:59     30.250000
4 2017-04-15 23:32:20     16.716667
```

```
5 2017-03-25 20:34:11      8.000000
6 2017-05-03 19:04:09     59.633333
7 2017-08-15 17:41:06     21.983333
8 2017-02-04 16:17:07     12.116667
9 2017-11-10 15:20:29     20.433333

[10 rows x 21 columns]
```

Check for missing data and duplicates using `.isna()` and `.drop_duplicates()`.

[94]:
```
# Check for missing data and duplicates using .isna() and .drop_duplicates()
print('Shape before removing duplicates',df.shape)
print('Shape after removing duplicates',df.drop_duplicates().shape)
df.isna().sum().sum()
```

```
Shape before removing duplicates (22699, 21)
Shape after removing duplicates (22699, 21)
```

[94]:
```
0
```

[95]:
```
df.describe()
```

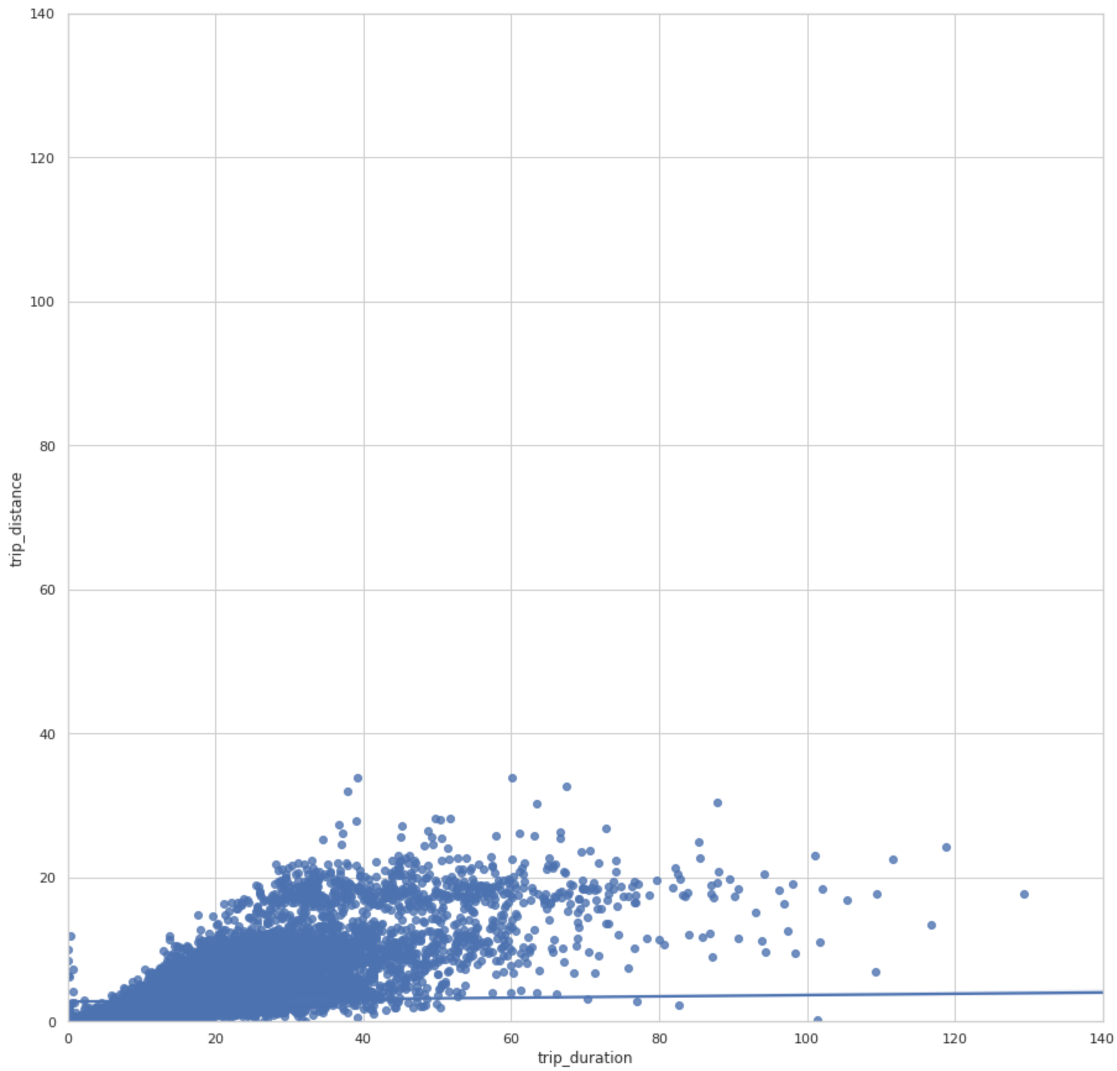
[95]:
```
         Unnamed: 0      VendorID  passenger_count  trip_distance  \
count  2.269900e+04  22699.000000     22699.000000   22699.000000
mean   5.675849e+07      1.556236         1.642319       2.913313
std    3.274493e+07      0.496838         1.285231       3.653171
min    1.212700e+04      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	trip_duration
count	22699.000000	22699.000000	22699.000000
mean	0.299551	16.310502	17.013777
std	0.015673	16.097295	61.996482
min	-0.300000	-120.300000	-16.983333
25%	0.300000	8.750000	6.650000
50%	0.300000	11.800000	11.183333
75%	0.300000	17.800000	18.383333
max	0.300000	1200.290000	1439.550000

Create a scatterplot to visualize the relationship between variables of interest.

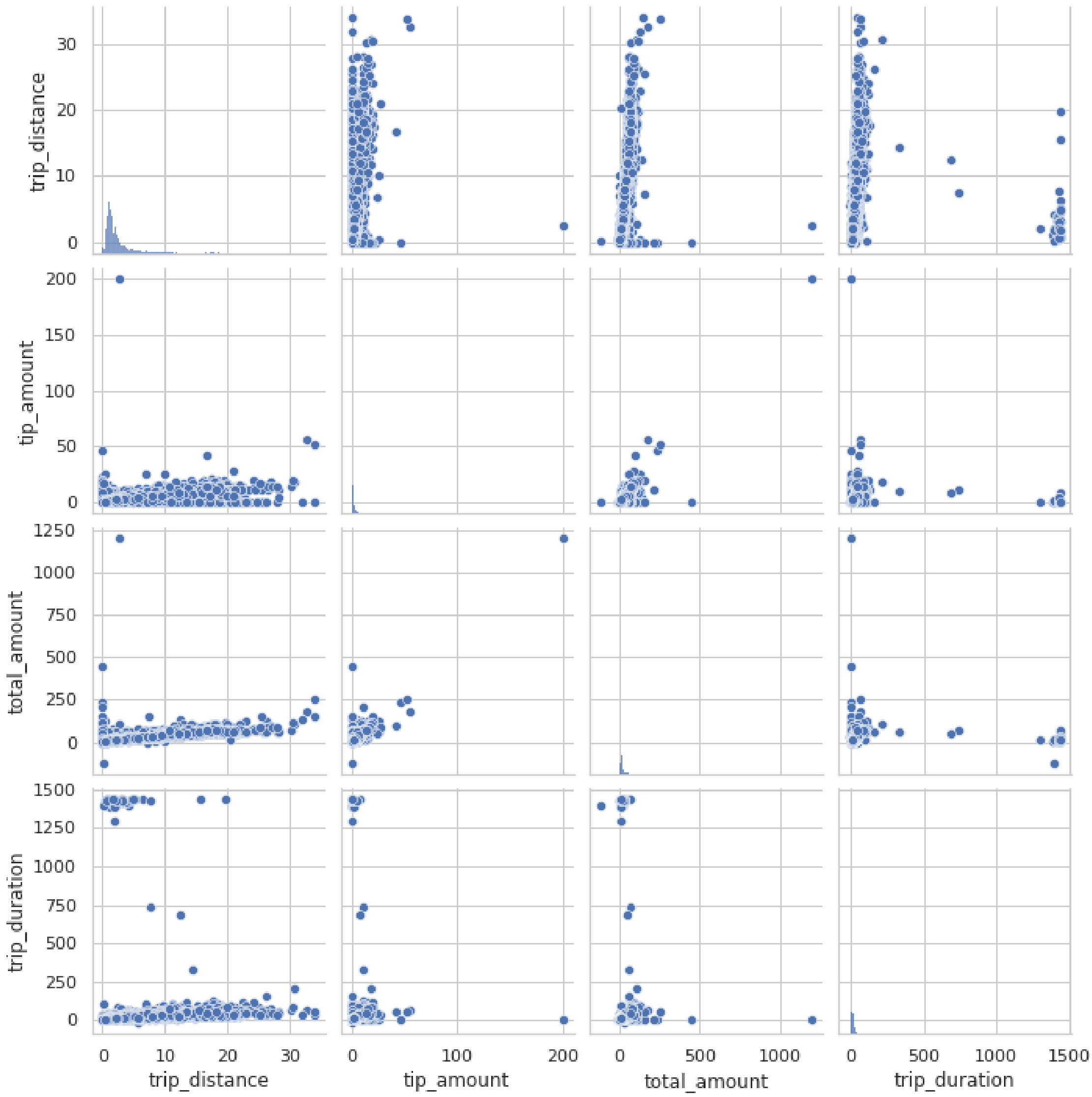
[96]:
```
# Create a scatterplot to visualize the relationship between variables of␣
↪interest
sns.set(style='whitegrid')
f = plt.figure()
f.set_figwidth(15)
f.set_figheight(15)
sns.regplot(x=df["trip_duration"], y=df["trip_distance"])
plt.ylim(0, 140)
plt.xlim(0,140)
plt.show()
```



Create a pairplot to visualize pairwise relationships between relevant variables.

[97]:
```
# Create a pairplot to visualize pairwise relationships between variables in
↪the data
sns.pairplot(df[['trip_distance', 'tip_amount', 'total_amount',
↪'trip_duration']])
```

[97]: <seaborn.axisgrid.PairGrid at 0x7f51fa0e5c10>

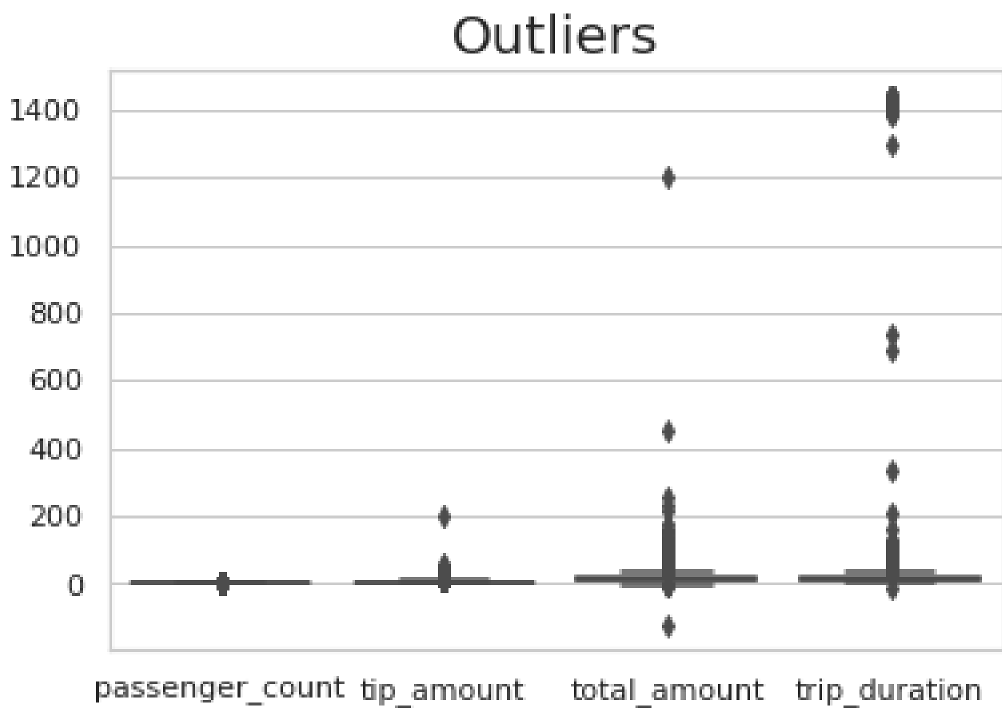


6.2 Task 2b. Address any outliers

Use a boxplot to visualize any outliers.

[98]:
```
# Create boxplot to visualize the outliers
g = sns.boxplot(data=df[["passenger_count","tip_amount","total_amount",␣
↪"trip_duration"]], showfliers=True);
g.set_title("Outliers",fontsize=20)
```

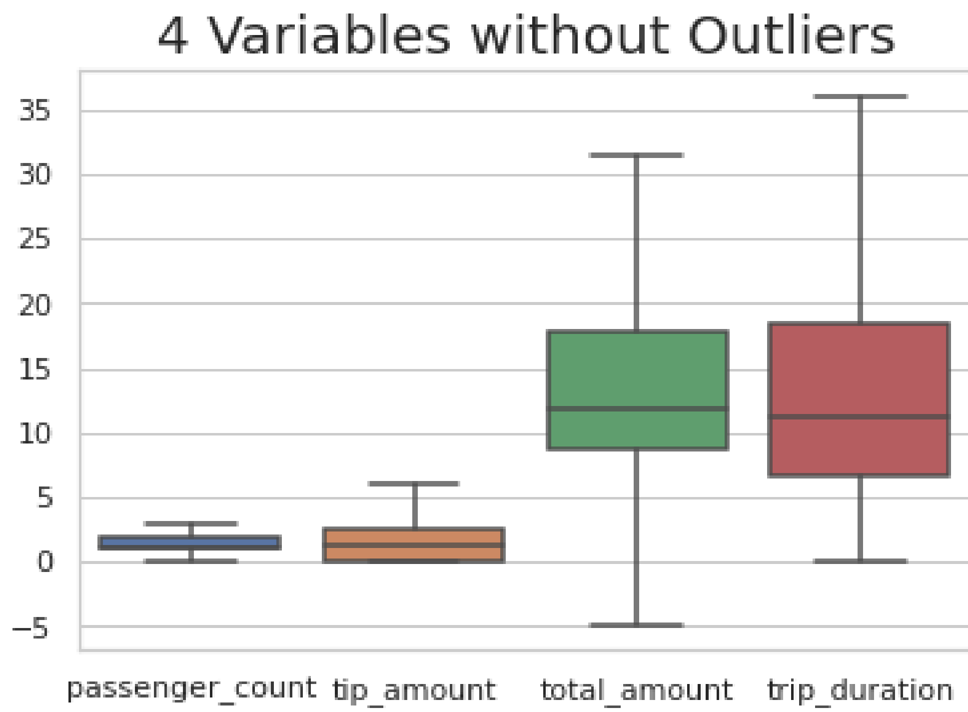
[98]:
```
Text(0.5, 1.0, 'Outliers')
```



Use a boxplot to visualize the distribution of the data without outliers.

[99]:
```
# Create boxplot to visualize distribution of data without outliers
g = sns.boxplot(data=df[["passenger_count","tip_amount","total_amount",␣
↪"trip_duration"]], showfliers=False);
g.set_title("4 Variables without Outliers",fontsize=20)
```

[99]:
```
Text(0.5, 1.0, '4 Variables without Outliers')
```



Remove outliers as needed.

[100]:
```
dpercentile_25 = df['trip_duration'].quantile(0.25)
dpercentile_75 = df['trip_duration'].quantile(0.75)
iqr= dpercentile_75 - dpercentile_25
upper_limit = dpercentile_75 + 1.5 * iqr

df[df["trip_duration"] > upper_limit] = upper_limit
df[df["trip_duration"] < 0] = 0

apercentile_25 = df["total_amount"].quantile(0.25)
apercentile_75 = df["total_amount"].quantile(0.75)
iqr= apercentile_75 - apercentile_25
aupper_limit = apercentile_75 + 1.5 * iqr

df[df["total_amount"] > aupper_limit] = aupper_limit
df[df["total_amount"] < 0] = 0
```

6.3 Task 2c. Identify correlations

Next, code a correlation matrix to help determine most correlated variables.

[101]:
```
df.corr(method='pearson')
```

[101]:

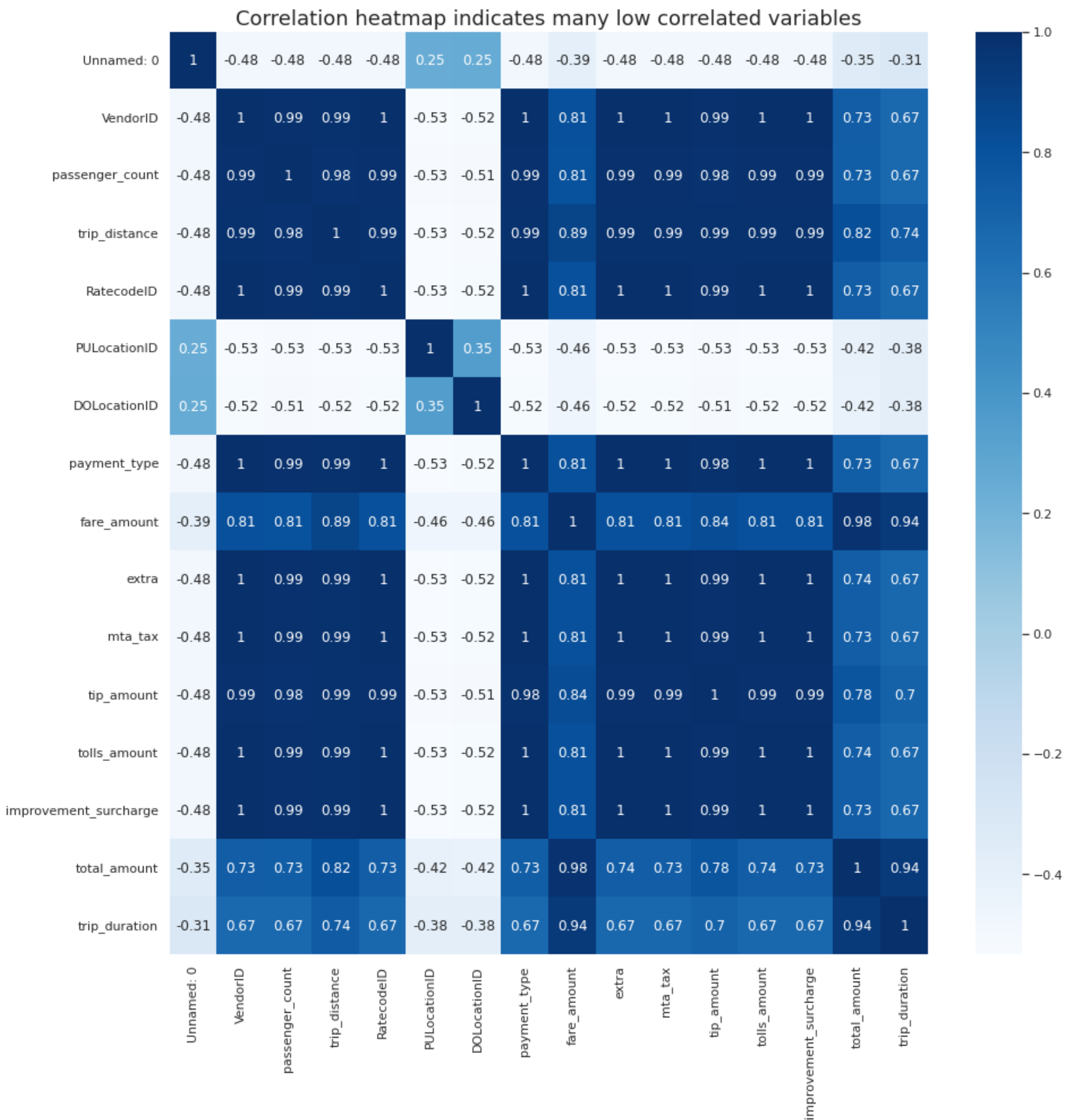
```
                       Unnamed: 0  VendorID  passenger_count  trip_distance  \
Unnamed: 0               1.000000 -0.484018        -0.480060      -0.479913
VendorID                -0.484018  1.000000         0.991630       0.986383
passenger_count         -0.480060  0.991630         1.000000       0.979200
trip_distance           -0.479913  0.986383         0.979200       1.000000
RatecodeID              -0.484752  0.998592         0.991031       0.987447
PULocationID             0.249919 -0.531011        -0.526598      -0.533220
DOLocationID             0.254780 -0.517895        -0.512815      -0.524301
payment_type            -0.483891  0.997301         0.989897       0.986066
fare_amount             -0.388750  0.812167         0.807055       0.885940
extra                   -0.484724  0.997964         0.990369       0.987087
mta_tax                 -0.484852  0.998656         0.991107       0.987586
tip_amount              -0.480481  0.988855         0.981257       0.985027
tolls_amount            -0.484900  0.998252         0.990722       0.987633
improvement_surcharge   -0.484874  0.998655         0.991107       0.987580
total_amount            -0.350981  0.733121         0.728428       0.818807
trip_duration           -0.309621  0.670141         0.666417       0.744132

                       RatecodeID  PULocationID  DOLocationID  payment_type  \
Unnamed: 0              -0.484752      0.249919      0.254780     -0.483891
VendorID                 0.998592     -0.531011     -0.517895      0.997301
passenger_count          0.991031     -0.526598     -0.512815      0.989897
trip_distance            0.987447     -0.533220     -0.524301      0.986066
RatecodeID               1.000000     -0.530776     -0.517753      0.998632
PULocationID            -0.530776      1.000000      0.349153     -0.530324
DOLocationID            -0.517753      0.349153      1.000000     -0.517543
payment_type             0.998632     -0.530324     -0.517543      1.000000
fare_amount              0.813416     -0.459258     -0.459545      0.811169
extra                    0.999219     -0.530773     -0.518097      0.997943
mta_tax                  0.999908     -0.530843     -0.517718      0.998685
tip_amount               0.990166     -0.526370     -0.514729      0.983980
tolls_amount             0.999550     -0.530851     -0.518053      0.998322
improvement_surcharge    0.999921     -0.530875     -0.517755      0.998688
total_amount             0.734320     -0.419093     -0.423095      0.726262
trip_duration            0.670954     -0.379278     -0.380769      0.668878

                       fare_amount     extra   mta_tax  tip_amount  \
Unnamed: 0               -0.388750 -0.484724 -0.484852   -0.480481
VendorID                  0.812167  0.997964  0.998656    0.988855
passenger_count           0.807055  0.990369  0.991107    0.981257
trip_distance             0.885940  0.987087  0.987586    0.985027
RatecodeID                0.813416  0.999219  0.999908    0.990166
PULocationID             -0.459258 -0.530773 -0.530843   -0.526370
DOLocationID             -0.459545 -0.518097 -0.517718   -0.514729
payment_type              0.811169  0.997943  0.998685    0.983980
fare_amount               1.000000  0.812888  0.813251    0.835297
extra                     0.812888  1.000000  0.999308    0.989821
```

```
mta_tax                  0.813251  0.999308  1.000000    0.990257
tip_amount               0.835297  0.989821  0.990257    1.000000
tolls_amount             0.814062  0.998920  0.999605    0.989955
improvement_surcharge    0.813258  0.999309  0.999999    0.990254
total_amount             0.980331  0.735808  0.734164    0.780913
trip_duration            0.940505  0.670915  0.671400    0.702192

                       tolls_amount  improvement_surcharge  total_amount  \
Unnamed: 0                -0.484900              -0.484874     -0.350981
VendorID                   0.998252               0.998655      0.733121
passenger_count            0.990722               0.991107      0.728428
trip_distance              0.987633               0.987580      0.818807
RatecodeID                 0.999550               0.999921      0.734320
PULocationID              -0.530851              -0.530875     -0.419093
DOLocationID              -0.518053              -0.517755     -0.423095
payment_type               0.998322               0.998688      0.726262
fare_amount                0.814062               0.813258      0.980331
extra                      0.998920               0.999309      0.735808
mta_tax                    0.999605               0.999999      0.734164
tip_amount                 0.989955               0.990254      0.780913
tolls_amount               1.000000               0.999609      0.736071
improvement_surcharge      0.999609               1.000000      0.734161
total_amount               0.736071               0.734161      1.000000
trip_duration              0.671988               0.671348      0.941669

                       trip_duration
Unnamed: 0                 -0.309621
VendorID                    0.670141
passenger_count             0.666417
trip_distance               0.744132
RatecodeID                  0.670954
PULocationID               -0.379278
DOLocationID               -0.380769
payment_type                0.668878
fare_amount                 0.940505
extra                       0.670915
mta_tax                     0.671400
tip_amount                  0.702192
tolls_amount                0.671988
improvement_surcharge       0.671348
total_amount                0.941669
trip_duration               1.000000
```

Visualize a correlation heatmap of the data.

[102]:
```
# Create correlation heatmap
plt.figure(figsize=(15,15))
```

```python
sns.heatmap(df.corr(method="pearson"), annot=True, cmap="Blues")
plt.title("Correlation heatmap indicates many low correlated variables",
          fontsize=18)
plt.show()
```



7 PACE: Construct

After analysis and deriving variables with close relationships, it is time to begin constructing the model. Consider the questions in the PACE Strategy Doc to reflect on the Constructing stage of this task. * Why did you select the X variables you did?

Dropped columns based on correlations between variables and multicollinearity, and fine-tuned by running and rerunning models to examine change in R^2, MAE, and RMSE.

7.1 Task 3a. Select outcome variable and features

Set your Y and X variables. Y represents the outcome variable, and X represents the features.

[103]:
```python
Y = df[["trip_duration"]]
X = df.drop(columns="trip_duration")
X.head()
```

[103]:
```
   Unnamed: 0  VendorID    tpep_pickup_datetime   tpep_dropoff_datetime  \
0  24870114.0       2.0   03/25/2017 8:55:43 AM   03/25/2017 9:09:47 AM
1  35634249.0       1.0   04/11/2017 2:53:28 PM   04/11/2017 3:19:58 PM
2 106203690.0       1.0   12/15/2017 7:26:56 AM   12/15/2017 7:34:08 AM
3  38942136.0       2.0   05/07/2017 1:17:59 PM   05/07/2017 1:48:14 PM
4  30841670.0       2.0  04/15/2017 11:32:20 PM  04/15/2017 11:49:03 PM

  passenger_count  trip_distance  RatecodeID store_and_fwd_flag  \
0             6.0           3.34         1.0                  N
1             1.0           1.80         1.0                  N
2             1.0           1.00         1.0                  N
3             1.0           3.70         1.0                  N
4             1.0           4.37         1.0                  N

  PULocationID  DOLocationID  payment_type  fare_amount  extra  mta_tax  \
0        100.0         231.0           1.0         13.0    0.0      0.5
1        186.0          43.0           1.0         16.0    0.0      0.5
2        262.0         236.0           1.0          6.5    0.0      0.5
3        188.0          97.0           1.0         20.5    0.0      0.5
4          4.0         112.0           2.0         16.5    0.5      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

   drop_off_converted    pick_up_converted
0 2017-03-25 09:09:47 2017-03-25 08:55:43
1 2017-04-11 15:19:58 2017-04-11 14:53:28
2 2017-12-15 07:34:08 2017-12-15 07:26:56
3 2017-05-07 13:48:14 2017-05-07 13:17:59
4 2017-04-15 23:49:03 2017-04-15 23:32:20
```

7.2 Task 3b. Pre-process data

To help with processing time, consider dropping irrelevant and redundant columns.

[104]:
```
columns_to_drop = ['tpep_pickup_datetime', 'tpep_dropoff_datetime',
                   'store_and_fwd_flag', 'passenger_count', 'VendorID',
                   'fare_amount', 'PULocationID', 'DOLocationID',␣
 ↪'total_amount',
                   'drop_off_converted', 'pick_up_converted']
X = X.drop(columns_to_drop, axis=1)
X = X.loc[:, ~X.columns.str.contains("Unnamed")]
X.head()
```

[104]:
```
   trip_distance  RatecodeID  payment_type  extra  mta_tax  tip_amount  \
0           3.34         1.0           1.0    0.0      0.5        2.76
1           1.80         1.0           1.0    0.0      0.5        4.00
2           1.00         1.0           1.0    0.0      0.5        1.45
3           3.70         1.0           1.0    0.0      0.5        6.39
4           4.37         1.0           2.0    0.5      0.5        0.00

   tolls_amount  improvement_surcharge
0           0.0                    0.3
1           0.0                    0.3
2           0.0                    0.3
3           0.0                    0.3
4           0.0                    0.3
```

Use `StandardScaler()` and `fit_transform()` to standardize the X variables.

[105]:
```
scaler = StandardScaler()
x_scaled = scaler.fit_transform(X)
print("X scaled:", X_scaled)
```

```
X scaled: [[-0.18116118 -0.33999272 -0.37659905 … -0.18555461 -0.34096692
  -0.33964103]
 [-0.35082338 -0.33999272 -0.37659905 … -0.05118079 -0.34096692
  -0.33964103]
 [-0.43895959 -0.33999272 -0.37659905 … -0.32751404 -0.34096692
  -0.33964103]
 …
 [-0.50285834 -0.33999272 -0.26740498 … -0.48464472 -0.34096692
  -0.33964103]
 [-0.28912804 -0.33999272 -0.37659905 … -0.30042255 -0.34096692
  -0.33964103]
 [-0.31777231 -0.33999272 -0.37659905 … -0.22998466 -0.34096692
  -0.33964103]]
```

7.3 Task 3c. Build model

Create training and testing sets.

[106]:
```
x_train, x_test, y_train, y_test = train_test_split(x_scaled, Y, test_size=0.2,␣
↪random_state=0)
```

Build and fit your model to the data.

[107]:
```
lr.fit(x_train, y_train)
```

[107]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

7.4 Task 3d. Evaluate model

Evaluate your model performance by calculating the residual sum of squares and the explained variance score (R^2). Calculate the Mean Absolute Error, Mean Squared Error, and the Root Mean Squared Error.

[108]:
```
r_sq = lr.score(x_train, y_train)
print("Coefficient of determination:", r_sq)
y_pred = lr.predict(x_train)
print("R^2:", r2_score(y_train, y_pred))
print("MAE:", mean_absolute_error(y_train,y_pred))
print("RMSE:",np.sqrt(mean_squared_error(y_train, y_pred)))
```

```
Coefficient of determination: 0.7391110948153115
R^2: 0.7391110948153116
MAE: 3.2119417836310444
RMSE: 4.5781331414091895
```

[109]:
```
r_sq_test = lr.score(x_test, y_test)
print("Coefficient of determination:", r_sq_test)
y_pred_test = lr.predict(x_test)
print("R^2:", r2_score(y_test, y_pred_test))
print("MAE:", mean_absolute_error(y_test,y_pred_test))
print("RMSE:",np.sqrt(mean_squared_error(y_test, y_pred_test)))
```

```
Coefficient of determination: 0.7332003491146651
R^2: 0.7332003491146653
MAE: 3.2214448090342804
RMSE: 4.601147377111918
```

8 PACE: Execute

Consider these questions PACE Strategy Doc to reflect on the Execute stage of this task.

8.1 Task 4a. Results

If the linear regression assumptions are met, the model results can be appropriately interpreted.

Use the code cell below to get `actual`,`predicted`, and `residual` for the testing set, and store them as columns in a `results` dataframe.

[110]:
```
results = pd.DataFrame(data={"actual": y_test["trip_duration"],
                             "predicted": y_pred_test.ravel()})
results["residual"] = results["actual"] - results["predicted"]
results.head()
```

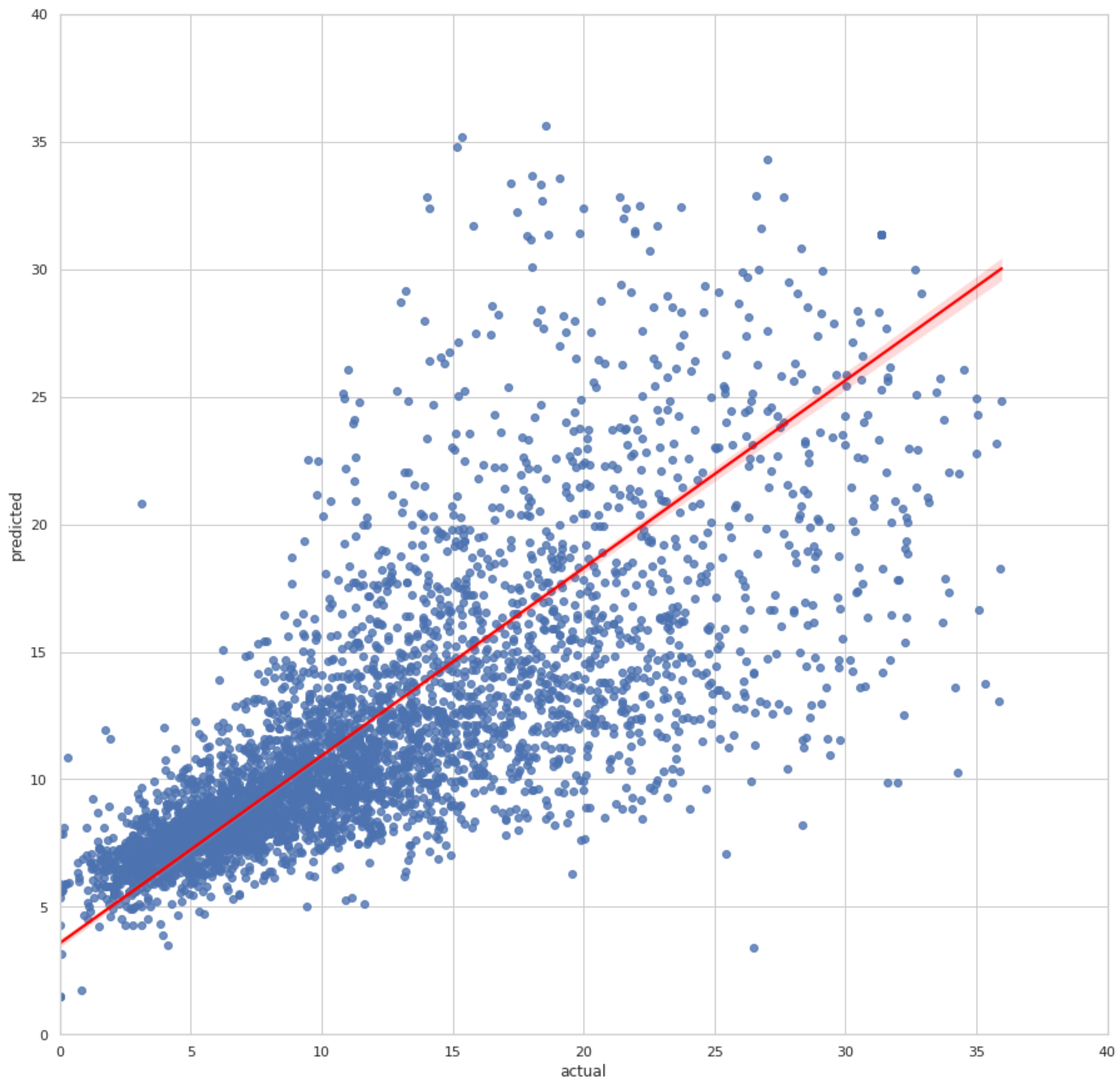
[110]:

	actual	predicted	residual
5818	18.016667	15.666455	2.350212
18134	31.375000	31.375124	-0.000124
4655	5.883333	7.334027	-1.450693
7378	15.950000	18.339652	-2.389652
13914	11.900000	11.543187	0.356813

8.2 Task 4b. Visualize model results

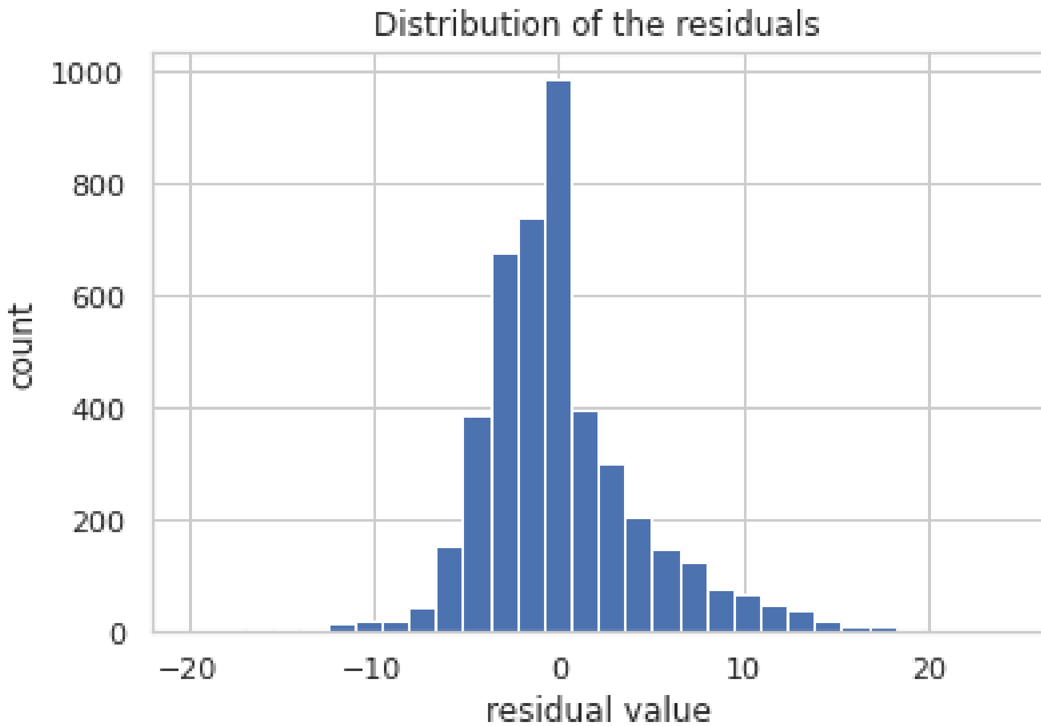
Create a scatterplot to visualize `predicted` over `actual`.

[111]:
```
sns.set(style='whitegrid')
f = plt.figure()
f.set_figwidth(15)
f.set_figheight(15)
sns.regplot(x="actual",
           y="predicted",
           data=results, line_kws={"color": "red"})
plt.ylim(0, 40)
plt.xlim(0,40)
plt.show()
```



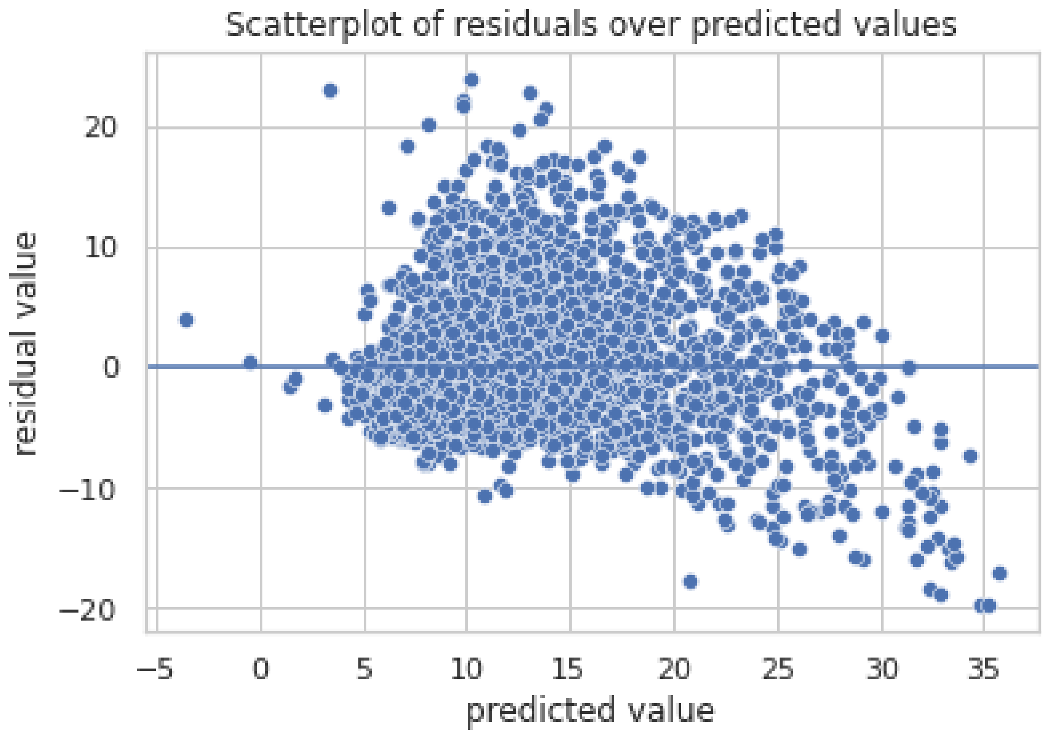
Visualize the distribution of the `residuals`.

[112]:
```
# Visualize the distribution of the `residuals`
### YOUR CODE HERE ###

plt.hist(results["residual"], bins=30)
plt.title("Distribution of the residuals")
plt.xlabel("residual value")
plt.ylabel("count")
plt.show()
```



Create a scatterplot of `residuals` over `predicted`.

[113]:
```
sns.scatterplot(x="predicted", y="residual", data=results)
plt.axhline(0)
plt.title("Scatterplot of residuals over predicted values")
plt.xlabel("predicted value")
plt.ylabel("residual value")
plt.show()
```



8.3 Task 4c. Conclusion

1. What results can be presented from this notebook?

1. It would be important to present the model's assumptions and precision of the model through the MAE and the RMSE