Data_Analysis and Visualization

May 1, 2022

1 Data analysis and visualization with python utilizing Seaborn and Matplotlib libraries.

TaxiCab is your neighbourhood friendly Taxi service making your life easier to commute through the busy streets of New York City. Since it began operations in 2019, the taxi service has really picked up and has become the number one service to use in NYC.

The main goal of this project is to analyze and visualize the provided dataset and draw the possible insight.

This project was completed using the jupyter notebook for Python 3 web-based interactive computing platform.Several libraries are used, including pandas and numpy. Visualization is also done via seaborn and matplotlib.

Many libraries, such as Dtale, can directly give a large number of visual details on a dataset. However, in this challenge, I did not use the built-in tools to offer a complete analysis.

**Please note that this project can be done with visualisation tool such as Power BI, Tableau.

[1]:
```
# importing the required libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import time
```

Import dataset

[2]:
```
dataset = pd.read_csv('yellow_tripdata_2021-01_raw.csv')
```

```
/opt/anaconda3/lib/python3.8/sitepackages/IPython/core/interactiveshel
l.py:3165: DtypeWarning: Columns (6) have mixed types.Specify dtype
option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

Check our data … Quickly

2 Analyze, clean, and correct any anomalies in the data

Pandas describe() is used to display some basic statistical information of a data frame or a sequence of numeric values, such as percentile, mean, and standard deviation.

Null values were present, and they must be eliminated.

"Trip distance" contains undesirable keywords such as km, which can be eliminated or dealt with in a variety of ways.

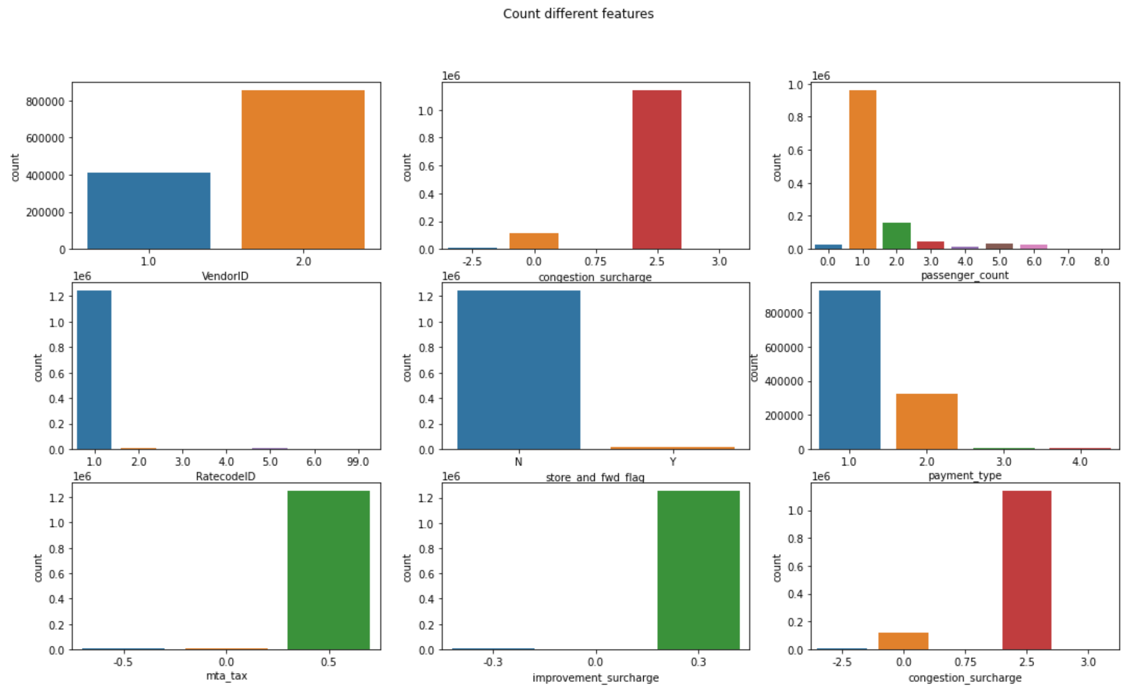
"All of the null values in passenger count can be eliminated."

We can look at the unique values in several features. This will assist us in filtering a large number of desired values.

[3]:
```
dataset.describe()
#Let's see if our dataset contains any null values.
dataset.isnull().sum()
#Because the datatype "Trip distance" is object, it should be changed
to float64.
#Unwanted phrases like km can be eliminated from #"Trip distance."
dataset["trip_distance"] = pd.to_numeric(dataset["trip_distance"],␣
↪errors='coerce') dataset =
dataset[dataset['passenger_count'].notna()]
dataset =
dataset[dataset['trip_distance'].notna()]
```

#Individually examining patterns of a variable with a few distinct values

[4]:
```
fig, axes = plt.subplots(3, 3, figsize=(18, 10))
fig.suptitle('Count different features')
ax=sns.countplot(x="VendorID",data=dataset,ax=axes[0, 0])
ax=sns.countplot(x="congestion_surcharge",data=dataset,ax=axes[0, 1])
ax=sns.countplot(x="passenger_count",data=dataset,ax=axes[0, 2])
ax=sns.countplot(x="RatecodeID",data=dataset,ax=axes[1, 0])
ax=sns.countplot(x="store_and_fwd_flag",data=dataset,ax=axes[1, 1])
ax=sns.countplot(x="payment_type",data=dataset,ax=axes[1, 2])
ax=sns.countplot(x="mta_tax",data=dataset,ax=axes[2, 0])
ax=sns.countplot(x="improvement_surcharge",data=dataset,ax=axes[2,
1]) ax=sns.countplot(x="congestion_surcharge",data=dataset,ax=axes[2,
2])
```



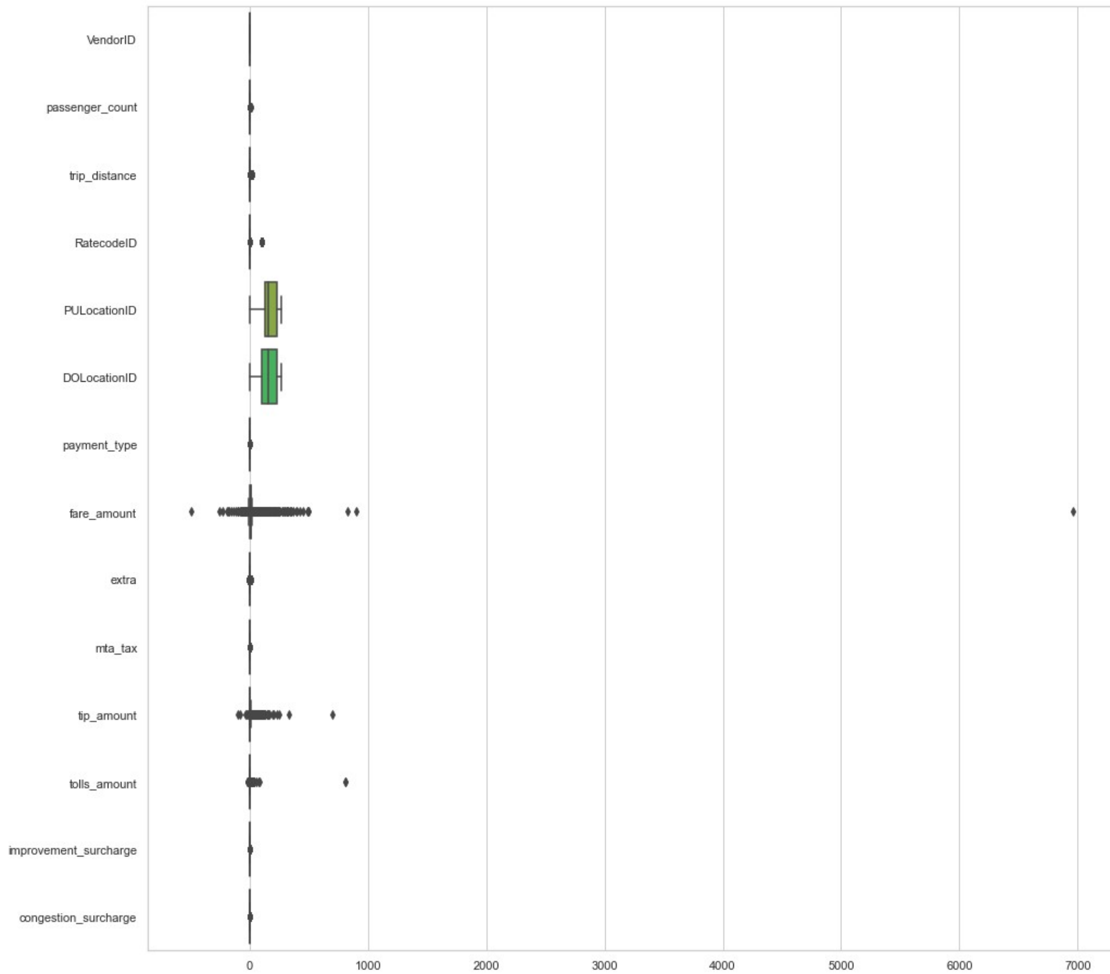
#Outlier removal and analysis (Box plot).

The box plot shows that there are negative values, which must be dealt with before moving further. For example, fare amount must be handled if it is negative.

There a plenty of additional negative numbers in other columns that must be dealt with before proceeding with the study.

[5]:
```
sns.set_theme(style="whitegrid")
plt.figure(figsize=(16,16))
sns.boxplot(data=dataset,orient="h")
```

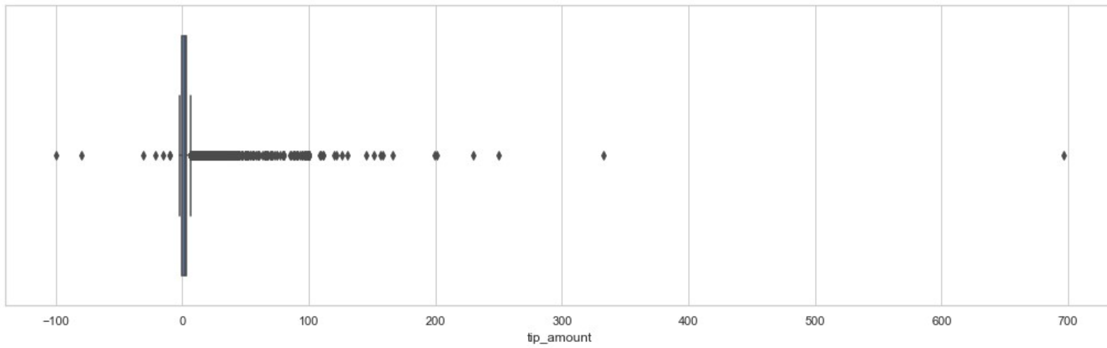
[5]: <AxesSubplot:>

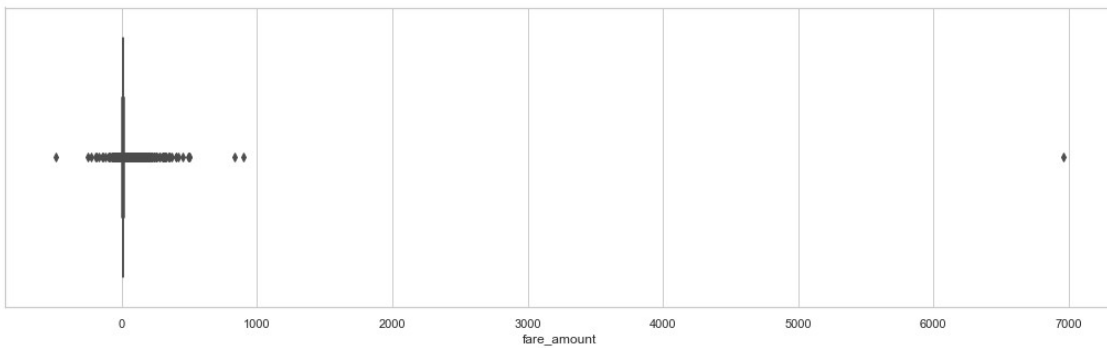


#Some more examples

[6]:
```
                sns.set_theme(style="whitegrid")
plt.figure(figsize=(18,5))          ax          =
sns.boxplot(x=dataset["tip_amount"],orient="h
")                    #ax                      =
sns.swarmplot(x=dataset["tip_amount"])
plt.figure(figsize=(18,5))
```

```
                                                   ay =
sns.boxplot(x=dataset["fare_amount"],orient="h")
```





When we look at box plot and statistic summary, we have few discoveries:

#The minimum fare amount is negative.

#The minimum passenger count is 0.

#For 7 and 8 persons, there are only a few journeys available. It is preferable to eliminate the item.

#Remove passengers with a count of 0 from the equation. This is illogical.

Remove all other numbers except the congestion surcharge, which is either 0 or 2.5.

#I'm going to eliminate some negative values from the dataset for "fare amount," "extra," "mta tax," "tip amount," "improvement surcharge," "tip amount," "tolls amount," and "congestion surcharge" because it's not desirable to have negative values in our scenario.

[7]:
```
dataset = dataset[dataset['passenger_count'] > 0]
dataset = dataset[dataset['passenger_count'] <=6]
dataset = dataset[(dataset['congestion_surcharge']
>= 0) &␣
↪(dataset['congestion_surcharge']<=2.5)] dataset =
dataset[(dataset['fare_amount'] > 0) & (dataset['extra'] >= 0)
&␣
```

```
↪(dataset['mta_tax'] >= 0)
       & (dataset['tip_amount'] >= 0) & (dataset['tolls_amount'] >= 0)
                   & (dataset['improvement_surcharge'] >= 0) &␣
↪(dataset['congestion_surcharge'] >= 0)]
dataset.describe()
```

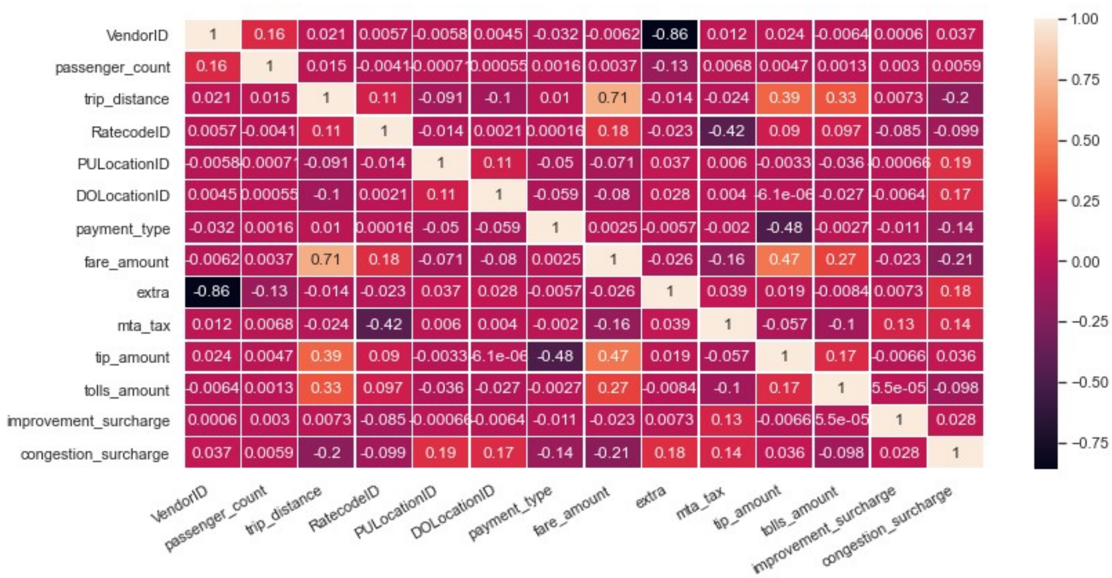
[7]:

```
          VendorID passenger_count trip_distance  RatecodeID \
count 1.230581e+06   1.230581e+06 1.230581e+06
                                  1.230581e+06
mean  1.689445e+00   1.441943e+00 2.486442e+00
                                  1.027947e+00
std   4.627210e-01   1.051176e+00 2.853972e+00
                                  4.539549e-01
min   1.000000e+00   1.000000e+00 0.000000e+00
                                  1.000000e+00
25%   1.000000e+00   1.000000e+00 9.900000e-01
                                  1.000000e+00
50%   2.000000e+00   1.000000e+00 1.600000e+00
                                  1.000000e+00
75%   2.000000e+00   1.000000e+00 2.750000e+00
                                  1.000000e+00
max   2.000000e+00   6.000000e+00 2.000000e+01
                                  9.900000e+01
      PULocationID DOLocationID payment_type     fare_amount
extra \ count 1.230581e+06 1.230581e+06 1.230581e+06 1.230581e+06
1.230581e+06
mean  1.665172e+02    1.635112e+02    1.267978e+00    1.080799e+01
      9.581482e-01
std   6.660927e+01    7.129066e+01    4.631332e-01    1.081163e+01
      1.207390e+00
min   1.000000e+00    1.000000e+00    1.000000e+00    1.000000e-02
      0.000000e+00
25%   1.320000e+02    1.070000e+02    1.000000e+00    6.000000e+00
      0.000000e+00
50%   1.620000e+02    1.620000e+02    1.000000e+00    8.000000e+00
      5.000000e-01
75%   2.360000e+02    2.360000e+02    2.000000e+00    1.200000e+01
      2.500000e+00
max   2.650000e+02    2.650000e+02    4.000000e+00    6.960500e+03
      7.000000e+00
          mta_tax tip_amount tolls_amount improvement_surcharge \
count 1.230581e+06 1.230581e+06 1.230581e+06     1.230581e+06
mean  4.984564e-01 1.924502e+00 1.528465e-01     2.999554e-01
std   2.773820e-02 2.317530e+00 1.449216e+00     3.658134e-03
min   0.000000e+00 0.000000e+00 0.000000e+00     0.000000e+00
25%   5.000000e-01 0.000000e+00 0.000000e+00     3.000000e-01
50%   5.000000e-01 1.860000e+00 0.000000e+00     3.000000e-01
```

```
75%    5.000000e-01 2.700000e+00 0.000000e+00     3.000000e-01
max    5.000000e-01 6.964800e+02 8.117500e+02     3.000000e-01
       congestion_surcharge

count          1.230581e+06

mean           2.270593e+00

std            7.217254e-01

min            0.000000e+00

25%            2.500000e+00

50%            2.500000e+00

75%            2.500000e+00

max            2.500000e+00
```

We can see the relationship between different features using this correlation heatmap.

[8]:
```python
corrmat = dataset.corr()
plt.figure(figsize=(13, 6))
sns.heatmap(corrmat, vmax=1, annot=True, linewidths=.5)
plt.xticks(rotation=30, horizontalalignment="right")
plt.show()
```



3 Feature creation which is required for the analysis

#Let’s make some additional features out of the current variables to get more information from the data.

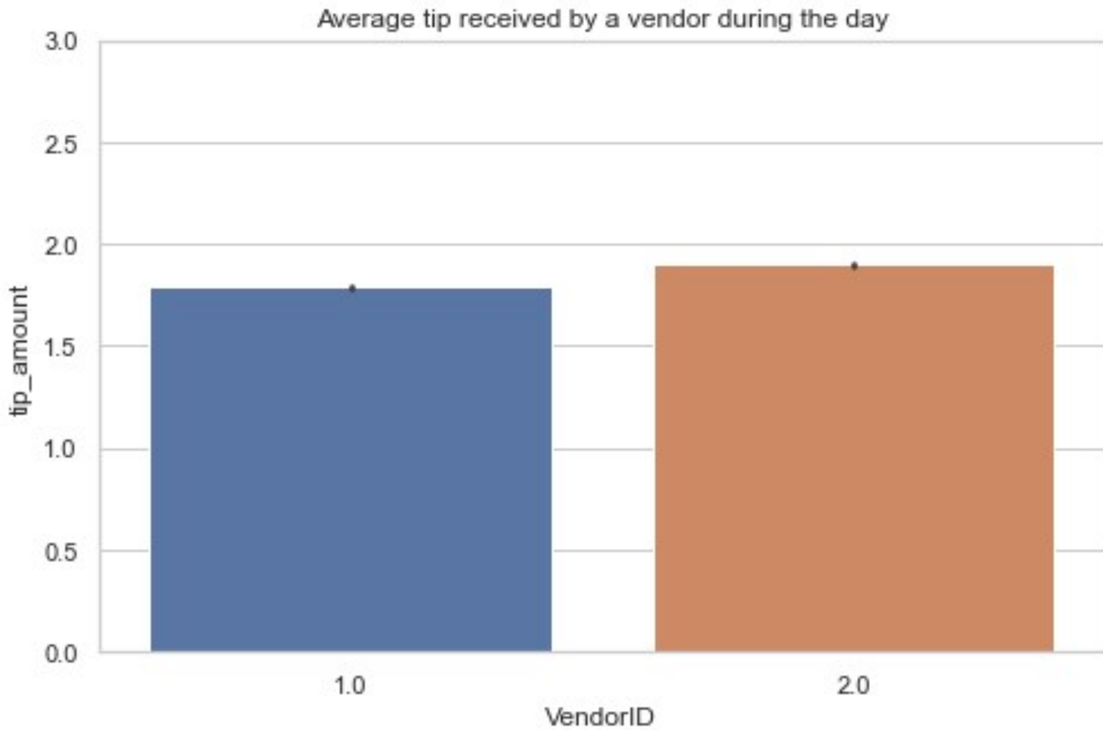
[9]:
```
#The 2 columns pickup_datetime and dropoff_datetime are now converted to␣
↪datetime format
#which makes analysis of date and time data much more easier
dataset['tpep_pickup_datetime']=pd.to_datetime(dataset['tpep_pickup_datetime']) dataset['tpep_dropoff_datetime']=pd.
↪to_datetime(dataset['tpep_dropoff_datetime'])

dataset['pickup_day']=dataset['tpep_pickup_datetime'].dt.day_name()
dataset['dropoff_day']=dataset['tpep_dropoff_datetime'].dt.day_name()

dataset['pickup_day_no']=dataset['tpep_pickup_datetime'].dt.weekday
dataset['dropoff_day_no']=dataset['tpep_dropoff_datetime'].dt.weekday

dataset['pickup_hour']=dataset['tpep_pickup_datetime'].dt.hour
dataset['dropoff_hour']=dataset['tpep_dropoff_datetime'].dt.hour
dataset['pickup_month']=dataset['tpep_pickup_datetime'].dt.month
dataset['dropoff_month']=dataset['tpep_dropoff_datetime'].dt.month
```

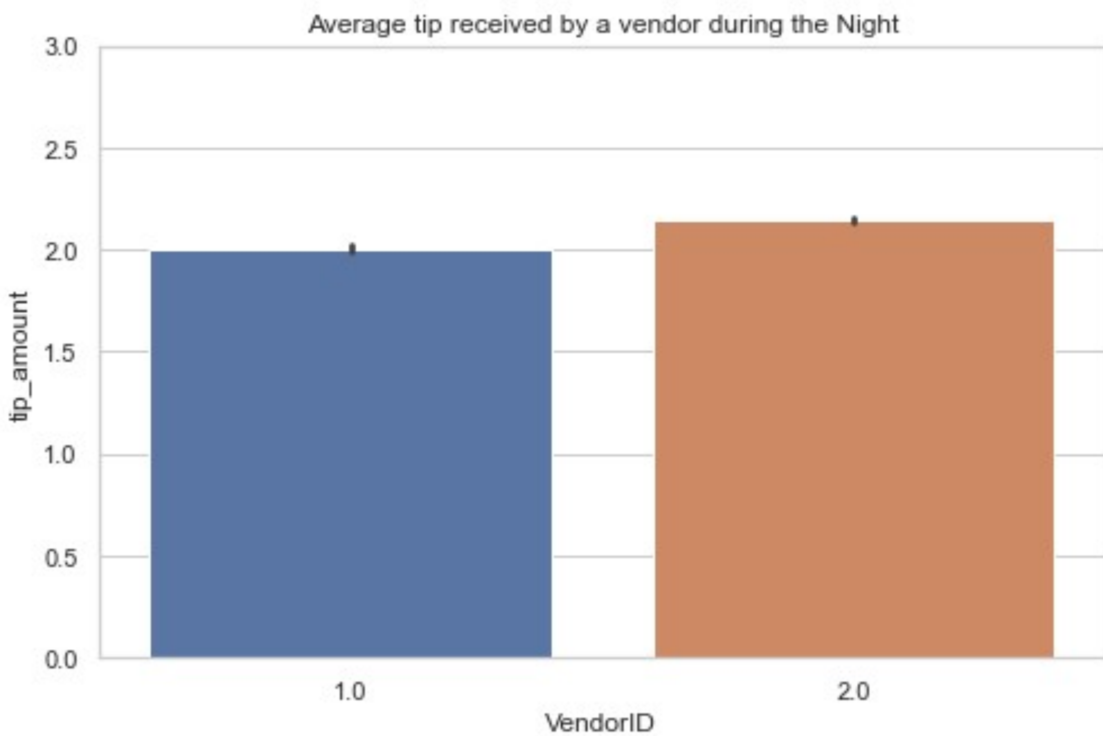
4 Average tip received by a vendor during the day (6 AM to 6 PM)

[10]:
```
#Function to identify day and night
def time_of_day(x):
    if x in range(6,18):
        return 'Day'
    else:
        return 'Night'
```

[11]:
```
dataset['pickup_timeofday']=dataset['pickup_hour'].apply(time_of_day)
dataset['dropoff_timeofday']=dataset['dropoff_hour'].apply(time_of_day)

dataset1 = dataset.loc[dataset.pickup_timeofday == 'Day']
dataset2 = dataset.loc[dataset.pickup_timeofday == 'Night']
```

[12]:
```
dataset1.groupby(['VendorID'], axis=0)['tip_amount'].mean()
dataset2.groupby(['VendorID'], axis=0)['tip_amount'].mean()

plt.figure(figsize=(8,5))
plt.title('Average tip received by a vendor during the day ')
sns.barplot(x="VendorID",  ="tip_amount",data=dataset1)
plt.ylim(0, 3)
plt.show()

plt.figure(figsize=(8,5))
plt.title('Average tip received by a vendor during the Night ')
sns.barplot(x="VendorID",  ="tip_amount",data=dataset2)
plt.ylim(0, 3)
plt.show()
```





The average tip amount is calculated in the diagram. Vendors receive higher tips at night than they do during the day, as can be seen from the visualisation.

5 Which time of the day is the busiest?

The busiest period is computed based on the whole duration of the event.

[13]:
```
#Trips per Day
figure,(ax1,ax2)=plt.subplots(ncols=2,figsize=(20,10))
figure.suptitle('Busiest time calculated on overall duration ')
ax1.set_title('Pickup Hours')
ax=sns.countplot(x="pickup_hour",data=dataset,ax=ax1)
ax2.set_title('Dropoff Hours')
ax=sns.countplot(x="dropoff_hour",data=dataset,ax=ax2)
```

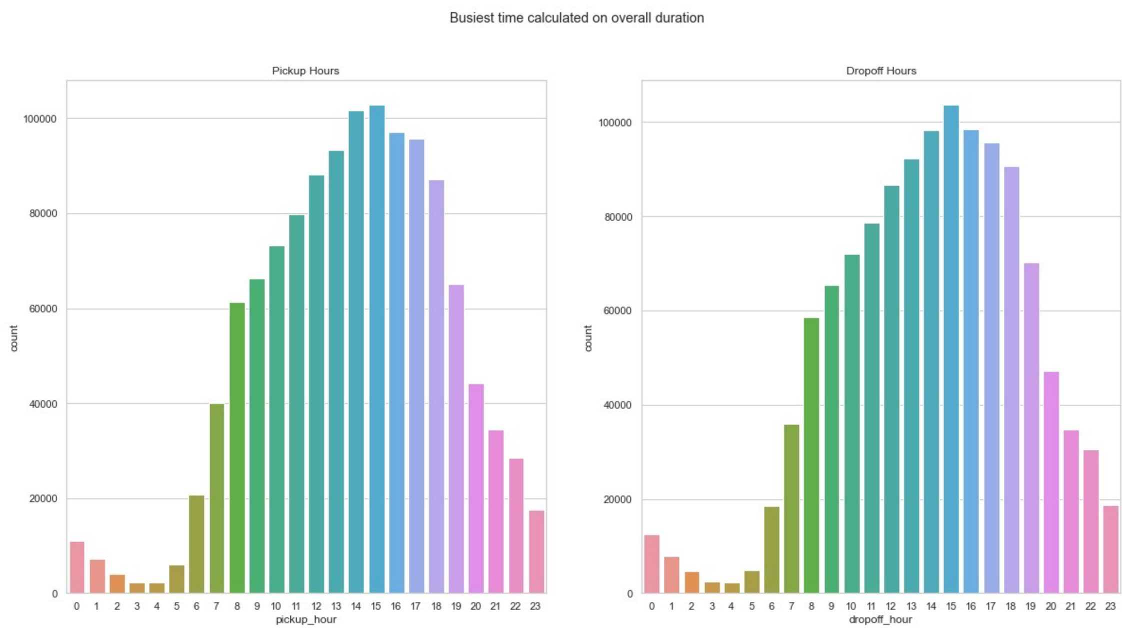


Figure: The busiest pickup and dropoff times are 3 to 4 p.m.

The TaxiCab becomes busier starting at 6 a.m. and peaks at 3 p.m.

It continues to operate till midnight.

Busiest time calculated on WEEKDAYS AND WEEKENDS

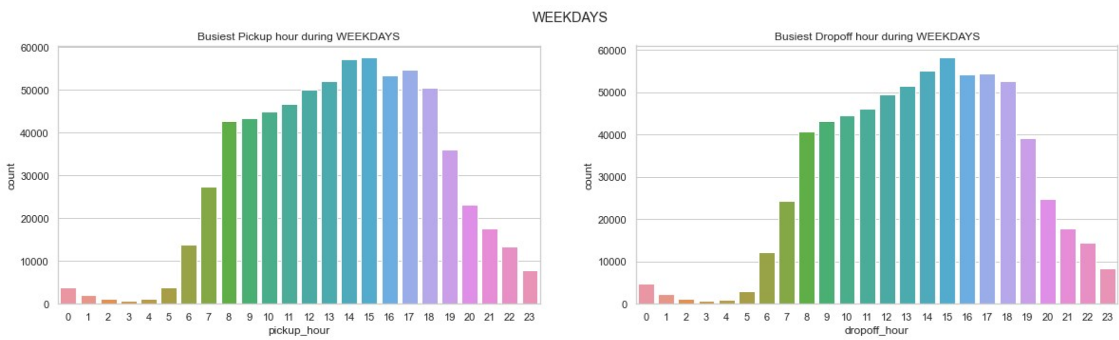
[14]:
```
#Function to identify weekday and weekends
def time_of_week(x):
    if x in range(0,4):
        return 'Weekday'
    else:
```

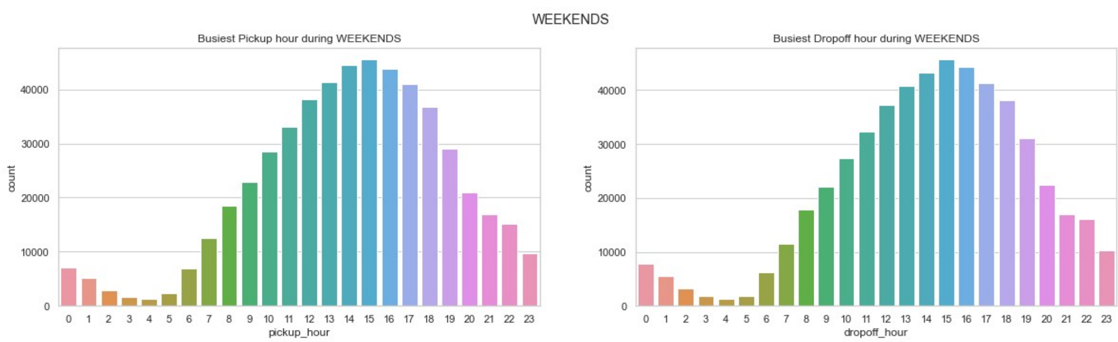
```
        return 'Weekend'
```

[15]:
```
dataset['time_of_week']=dataset['pickup_day_no'].apply(time_of_week)
```

[16]:
```
dataset3 = dataset.loc[dataset.time_of_week == 'Weekday']
dataset4 = dataset.loc[dataset.time_of_week == 'Weekend']
```

[17]:
```
figure,(ax1,ax2)=plt.subplots(ncols=2,figsize=(20,5))
figure.suptitle('WEEKDAYS')
ax1.set_title('Busiest Pickup hour during WEEKDAYS ')
ax = sns.countplot(x="pickup_hour",data=dataset3, ax=ax1)
ax2.set_title('Busiest Dropoff hour during WEEKDAYS ')
ax = sns.countplot(x="dropoff_hour",data=dataset3, ax=ax2)

figure,(ax3,ax4)=plt.subplots(ncols=2,figsize=(20,5))
figure.suptitle('WEEKENDS')
ax3.set_title('Busiest Pickup hour during WEEKENDS ')
ax = sns.countplot(x="pickup_hour",data=dataset4, ax=ax3)
ax4.set_title('Busiest Dropoff hour during WEEKENDS ')
ax = sns.countplot(x="dropoff_hour",data=dataset4, ax=ax4)
```





We can produce a more detailed visualisation of the busiest hour when measured individually on weekdays and weekends.

Weekend nights appear to be busier than weekday nights.

Weekday office travel time appears to be busier than weekend travel time, which is understandable.

6 Classify trips based on payment type (not the number in the excel sheet but the actual payment type. Use metadata table for reference)

[18]:
```
def payment_type(x):
    if x == 1:
        return 'Credit card'
    elif x == 2:
        return 'Cash'
    elif x == 3:
        return 'No Charge'
    elif x == 4:
        return 'Dispute'
    elif x == 5:
        return 'Unknown'
    elif x == 6:
        return 'Voided trip'
    else:
        return 'Payment type not found'
```

[19]:
```
dataset['Payment_style']=dataset['payment_type'].apply(payment_type)
```

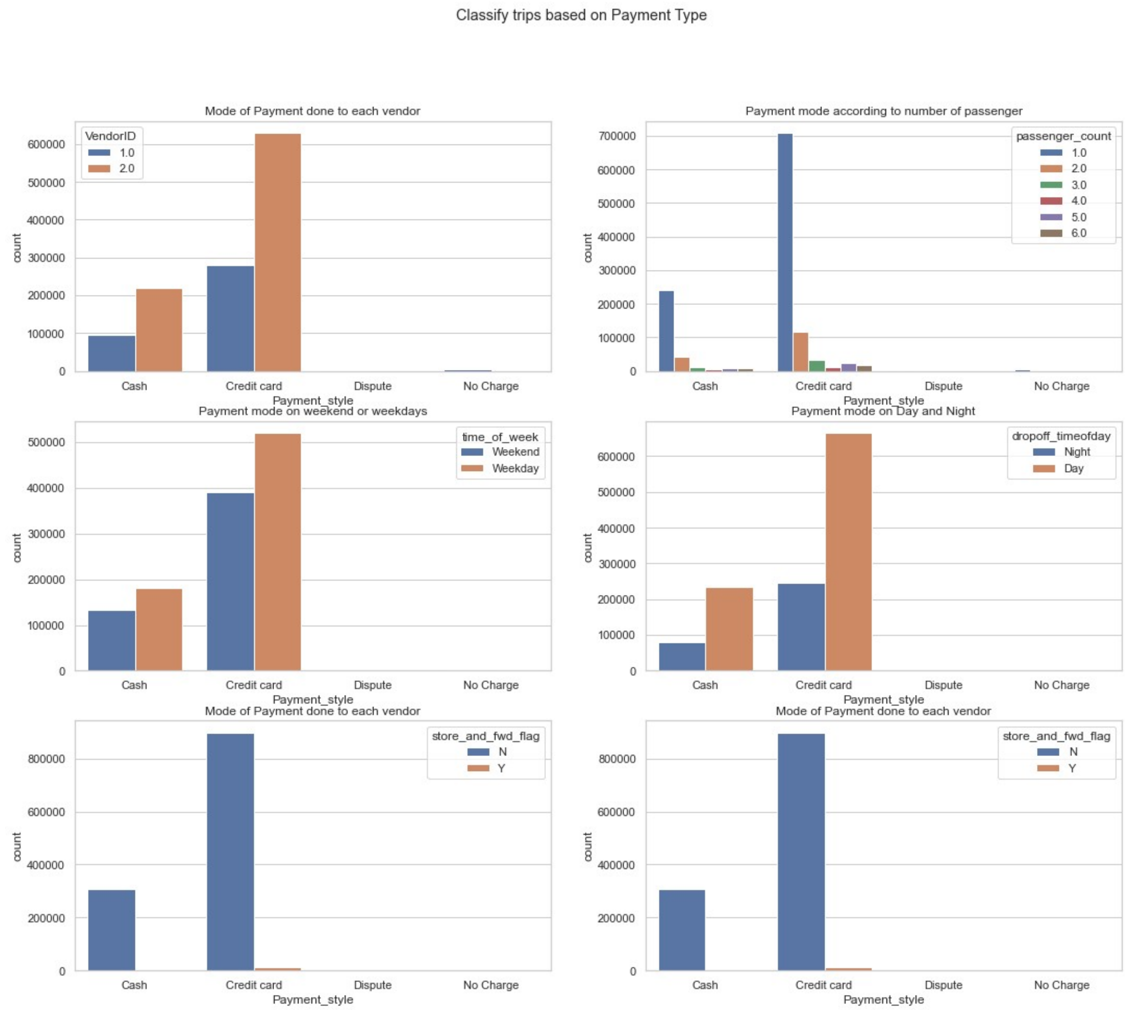
[20]:
```
dataset.head()
```

[20]:
```
VendorID tpep_pickup_datetime tpep_dropoff_datetime passenger_count \
0     1.0 2021-01-01 00:30:10 2021-01-01 00:36:12         1.0
1     1.0 2021-01-01 00:51:20 2021-01-01 00:52:19         1.0
2     1.0 2021-01-01 00:43:30 2021-01-01 01:11:06         1.0
4     2.0 2021-01-01 00:31:49 2021-01-01 00:48:21         1.0
5     1.0 2021-01-01 00:16:29 2021-01-01 00:24:30         1.0
   trip_distance RatecodeID store_and_fwd_flag PULocationID
   DOLocationID \
0          2.10        1.0                  N          142           43
1          0.20        1.0                  N          238          151
2         14.70        1.0                  N          132          165
4          4.94        1.0                  N           68           33
5          1.60        1.0                  N          224           68
   payment_type … pickup_day_no dropoff_day_no pickup_hour \
0           2.0 …              4              4           0
1           2.0 …              4              4           0
2           1.0 …              4              4           0
```

```
4          1.0 …              4             4          0
5          1.0 …              4             4          0
   dropoff_hour pickup_month dropoff_month pickup_timeofday \
0            0            1             1           Night
1            0            1             1           Night
2            1            1             1           Night
4            0            1             1           Night
5            0            1             1           Night

  dropoff_timeofday time_of_week Payment_style
0           Night     Weekend    Cash
1           Night     Weekend    Cash
2           Night     Weekend    Credit card
4           Night     Weekend    Credit card
5           Night     Weekend    Credit card
[5 rows x 29 columns]
```

[21]:
```
fig, axes = plt.subplots(3, 2, figsize=(18, 15))
fig.suptitle('Classify trips based on Payment Type')

ax1=sns.countplot(x="Payment_style", hue = 'VendorID', data=dataset,
ax=axes[0,␣
↪0]) ax1.title.set_text('Mode of Payment done to
each vendor')

ax2=sns.countplot(x="Payment_style", hue = 'passenger_count',
data=dataset,␣
↪ax=axes[0, 1]) ax2.title.set_text('Payment mode
according to number of passenger')

ax3=sns.countplot(x="Payment_style", hue =␣
↪'time_of_week',data=dataset,ax=axes[1, 0])
ax3.title.set_text('Payment mode on weekend or weekdays')

ax4=sns.countplot(x="Payment_style", hue =␣
↪'dropoff_timeofday',data=dataset,ax=axes[1, 1])
ax4.title.set_text('Payment mode on Day and Night')

ax5=sns.countplot(x="Payment_style", hue =␣
↪'store_and_fwd_flag',data=dataset,ax=axes[2, 0])
ax5.title.set_text('Mode of Payment done to each vendor')

ax6=sns.countplot(x="Payment_style", hue =␣
↪'store_and_fwd_flag',data=dataset,ax=axes[2, 1])
```

```
ax6.title.set_text('Mode of Payment done to each vendor ')
```



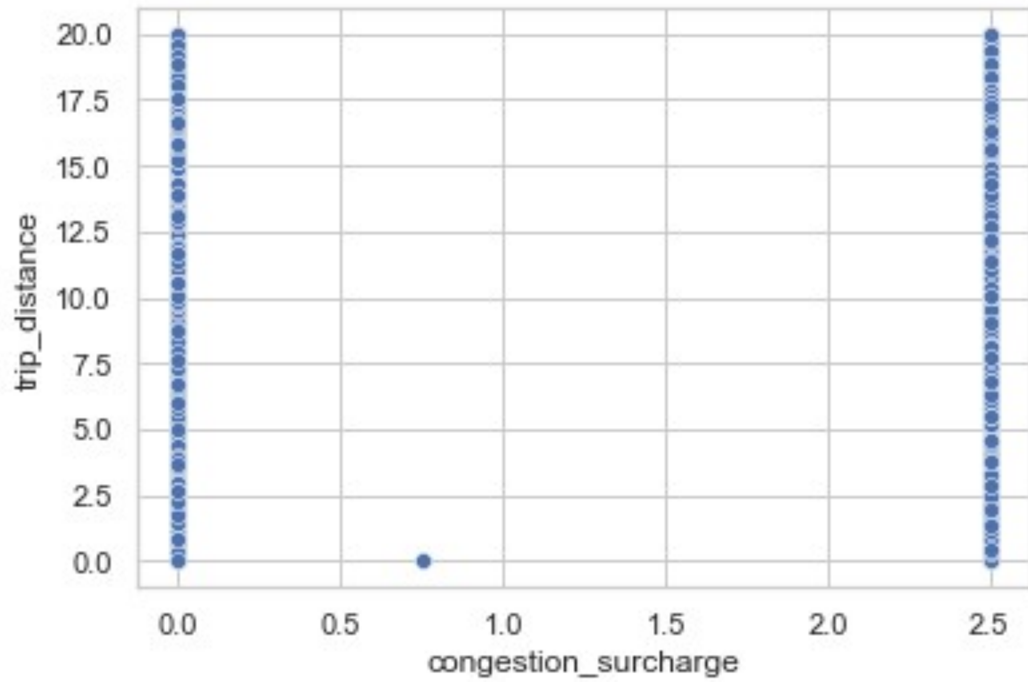
This visualization was created solely to show the mode of payment that was used in comparison to other elements.

The best way to visualize is to create a ratio and plot the findings in percentage. This gives you a lot more information.

7 Is there any relationship between congestion surcharge and trip distance?

[22]:
```
sns.scatterplot(data=dataset, x="congestion_surcharge",
y="trip_distance")
```

[22]:
```
<AxesSubplot:xlabel='congestion_surcharge', ylabel='trip_distance'>
```



The congestion surcharge is obviously random on each journey distance, as can be seen in the plot above. We may conclude that there is no link between congestion surcharge and travel distance based on our findings. Moreover, it can also be verified with the correlation heat map plotted obove.

According to my findings, there is a link between PULocationID, DOLocationID, and the congestion surcharge. Due to a lack of time, I was unable to present my thoughts on the subject.