MDSC - 102
Inferential Statistics (P)
End Sem Practicals - 2023

Dataset :- taxi_trip_fare_prediction

Source :- taxi_trip_fare_prediction

Regd No. :- 23905

Subject Code :- MDSC - 102

Date :- 17/10/23

Description of Dataset and it's Features:-

The data set basically talks about the traveling distance of a taxi and fare being collected for each trip from the users/people. The data set chosen is a training dataset which has 35000 rows and 20 columns, i.e, 35000 different values for the 20 features.

Data Description :-

Trip_distance :-

- ➔ The elapsed trip distance in miles reported by the taximeter.

Rate_code :-

- ➔ The final rate code is in effect at the end of the trip.
- ➔ 1= Standard rate,2=JFK,3=Newark, 4=Nassau or Westchester, 5=Negotiated fare,6=Group ride

Storeandfwd_flag :-

- ➔ This flag indicates whether the trip record was held in vehicle memory before sending it to the vendor and determines if the trip was stored in the server and forwarded to the vendor.
- ➔ Y = store and forward trip
- ➔ N = not a store and forward trip.

Payment_type :-

- ➔ A numeric code signifying how the passenger paid for the trip.
- ➔ 1 = Credit card, 2 = Cash, 3 = No charge, 4 = Dispute, 5= Unknown, 6 = Voided trip

Fare_amount :-

- ➔ The time-and-distance fare calculated by the meter

Extra :-

- ➔ Miscellaneous extras and surcharges.

Mta_tax :-

- ➔ $0.50 MTA tax that is automatically triggered based on the metered rate in use.

Trip_amount :-

- ➔ Tip amount credited to the driver for credit card transactions.

Tolls_amount :-

➔ Total amount of all tolls paid in the trip.

Imp_surcharge :-

➔ $0.30 extra charges added automatically to all rides

Total_amount :-

➔ The total amount charged to passengers. Does not include cash tips.

Pickup_location_id :-

➔ TLC Taxi Zone in which the taximeter was engaged.

Dropoff_location_id :-

➔ TLC Taxi Zone in which the taximeter was disengaged.

Year :-

➔ The year in which the taxi trip was taken.

Month :-

➔ The month on which the taxi trip was taken.

Day :-

➔ The day on which the taxi trip was taken.

Day_of_week :-

➔ The day of the week on which the taxi trip was taken.

Hour_of_day :-

➔ Used to determine the hour of the day in 24 hours format.

Trip_duration :-

➔ The total duration of the trip in seconds.

Calculated_total_amount :-

➔ The total amount the customer has to pay for the taxi.

Loading the Dataset :-

```
# Loading the dataset into a dataframe.
train_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/MDSC-102/final lab/train.csv')
train_df
```

	trip_distance	rate_code	store_and_fwd_flag	payment_type	fare_amount	extra	mta_tax	tip_amount	tolls_amount	imp_surcharge	total_amount	pickup_locat
0	9.01	NaN	N	1.0	26.0	0.0	0.5	8.14	5.76	0.3	40.70	
1	0.20	1.0	NaN	1.0	NaN	0.0	0.5	0.75	0.00	0.3	4.55	
2	9.65	1.0	N	1.0	NaN	NaN	0.5	9.61	5.76	0.3	57.67	
3	NaN	NaN	N	1.0	30.0	0.5	0.5	9.25	5.76	0.3	46.31	
4	5.80	1.0	N	1.0	21.5	NaN	0.5	4.56	0.00	NaN	27.36	
...	...	...	...	...	...	...	...	...	...	...	...	
34995	NaN	1.0	N	1.0	59.5	0.5	NaN	10.00	5.76	0.3	NaN	
34996	NaN	NaN	N	1.0	30.0	0.0	0.5	6.58	5.76	NaN	43.14	
34997	6.78	1.0	NaN	1.0	23.0	0.0	0.5	5.95	0.00	0.3	NaN	
34998	0.26	1.0	NaN	2.0	3.0	0.0	0.5	0.00	NaN	NaN	NaN	
34999	18.40	NaN	NaN	1.0	53.0	1.0	0.5	10.96	0.00	0.3	65.76	

35000 rows × 20 columns

Data Preprocessing:-

➔ Checking for null values.

```
[ ] null_count = train_df.isnull().sum()
    null_count
```

```
trip_distance             6375
rate_code                 6372
store_and_fwd_flag        6363
payment_type              6252
fare_amount               6301
extra                     6258
mta_tax                   6296
tip_amount                6367
tolls_amount              6398
imp_surcharge             6334
total_amount              6295
pickup_location_id        6461
dropoff_location_id       6311
year                      6478
month                     6375
day                       6381
day_of_week               6212
hour_of_day               6436
trip_duration             6367
calculated_total_amount   6400
dtype: int64
```

➔ Creating a list of null valued columns.

```
null_columns = train_df.columns[train_df.isnull().any()]
null_columns
```

```
Index(['trip_distance', 'rate_code', 'store_and_fwd_flag', 'payment_type',
       'fare_amount', 'extra', 'mta_tax', 'tip_amount', 'tolls_amount',
       'imp_surcharge', 'total_amount', 'pickup_location_id',
       'dropoff_location_id', 'year', 'month', 'day', 'day_of_week',
       'hour_of_day', 'trip_duration', 'calculated_total_amount'],
      dtype='object')
```

➔ Replacing null values with mode.

```
[10] for column in null_columns:
       if train_df[column].dtype == 'float64':
         train_df[column].fillna(train_df[column].mode()[0], inplace = True)
       elif train_df[column].dtype == 'object':
         train_df[column].fillna(train_df[column].mode()[0], inplace = True)
```

```
[11] null_count = train_df.isnull().sum()
     print(null_count)
     train_df.info()
```

```
trip_distance             0
rate_code                 0
store_and_fwd_flag        0
payment_type              0
fare_amount               0
extra                     0
mta_tax                   0
tip_amount                0
tolls_amount              0
imp_surcharge             0
total_amount              0
pickup_location_id        0
dropoff_location_id       0
year                      0
month                     0
day                       0
day_of_week               0
hour_of_day               0
trip_duration             0
calculated_total_amount   0
dtype: int64
```

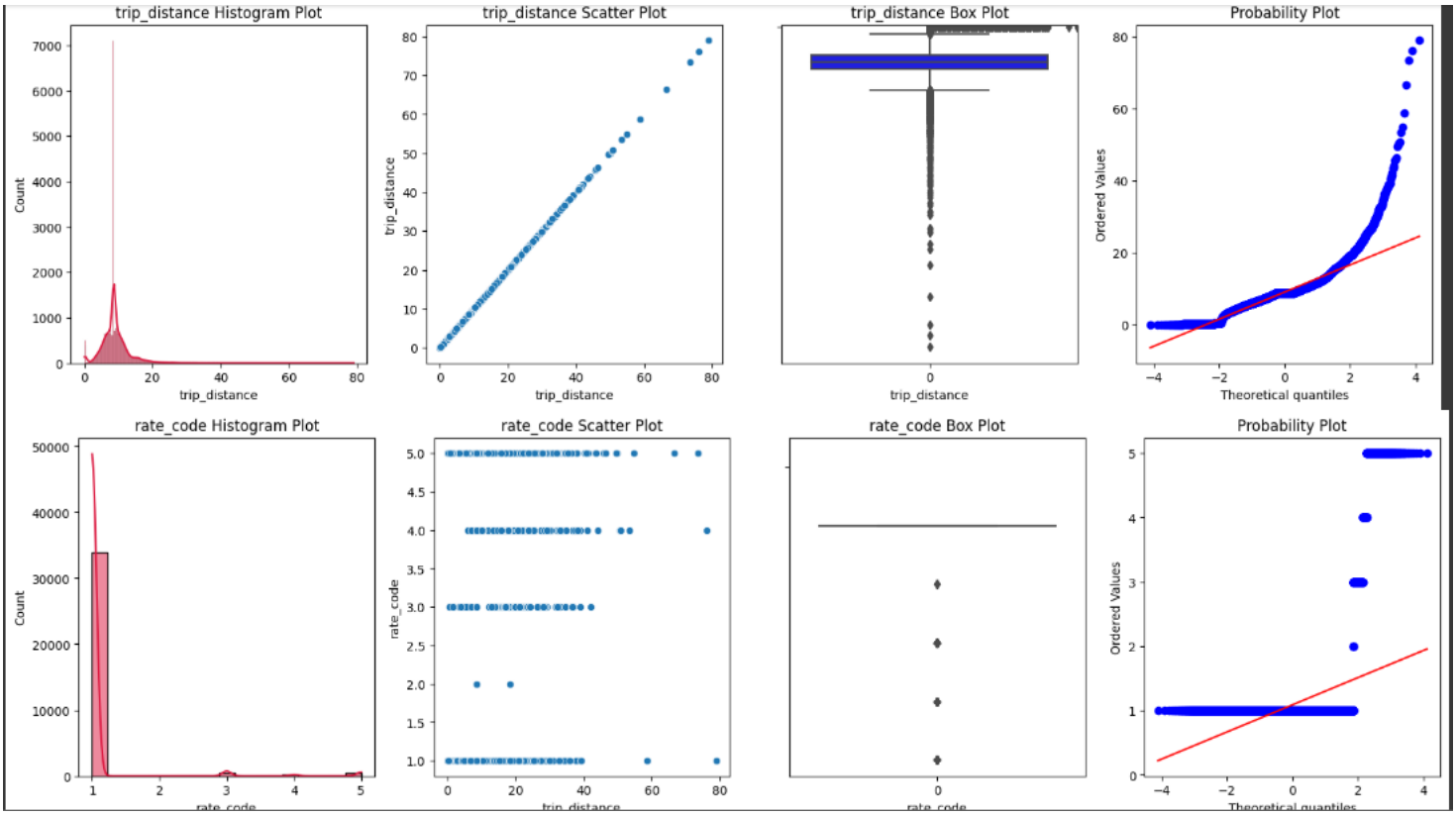
➔ Replacing the object value i.e Y/N with 1/0, hence converting the data type to int64.

```
[12] for element in train_df['store_and_fwd_flag']:
       if element == 'Y' or element =='y':
         train_df['store_and_fwd_flag'].replace('Y',1, inplace = True)

       else:
         train_df['store_and_fwd_flag'].replace('N',0, inplace = True)
```

Data Visualization :-

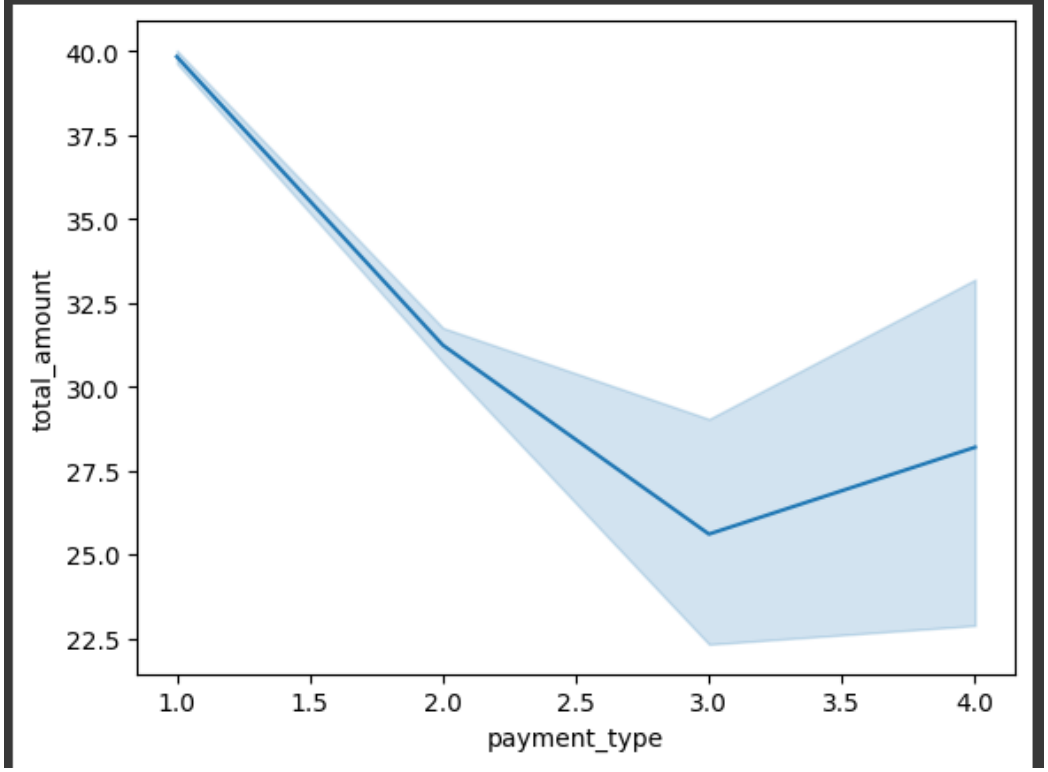
➔ Plotting as many plots as possible for insights into the dataset.

```
for element in train_df:
  i, j = 1,1
  plot.figure(figsize = (20,5))
  plot.subplot(i,4,j)
  sns.histplot(data = train_df, x = train_df[element], kde = True, color = 'crimson').set(title = ""+element+" Histogram Plot")
  j += 1

  plot.subplot(i,4,j)
  sns.scatterplot(data = train_df, x = 'trip_distance', y = element).set(title = ""+element+" Scatter Plot")
  j += 1

  plot.subplot(i,4,j)
  sns.boxplot(train_df, x = train_df[element], color = 'yellow').set(title = ""+element+" Box Plot")
  sns.boxplot(train_df[element], color = 'b')
  j += 1

  plot.subplot(i,4,j)
  sp.stats.probplot(train_df[element], plot = plot)
  plot.show()
```

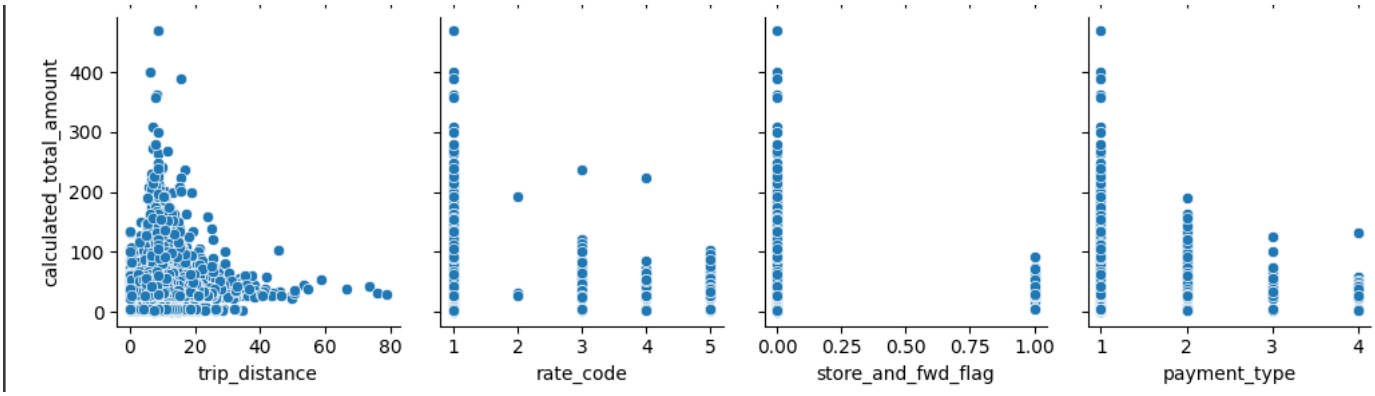


➔ Using sns for a line plot between payment type and total amount features.

```
sns.lineplot(x="payment_type", y="total_amount", data=train_df)
plot.show()
```

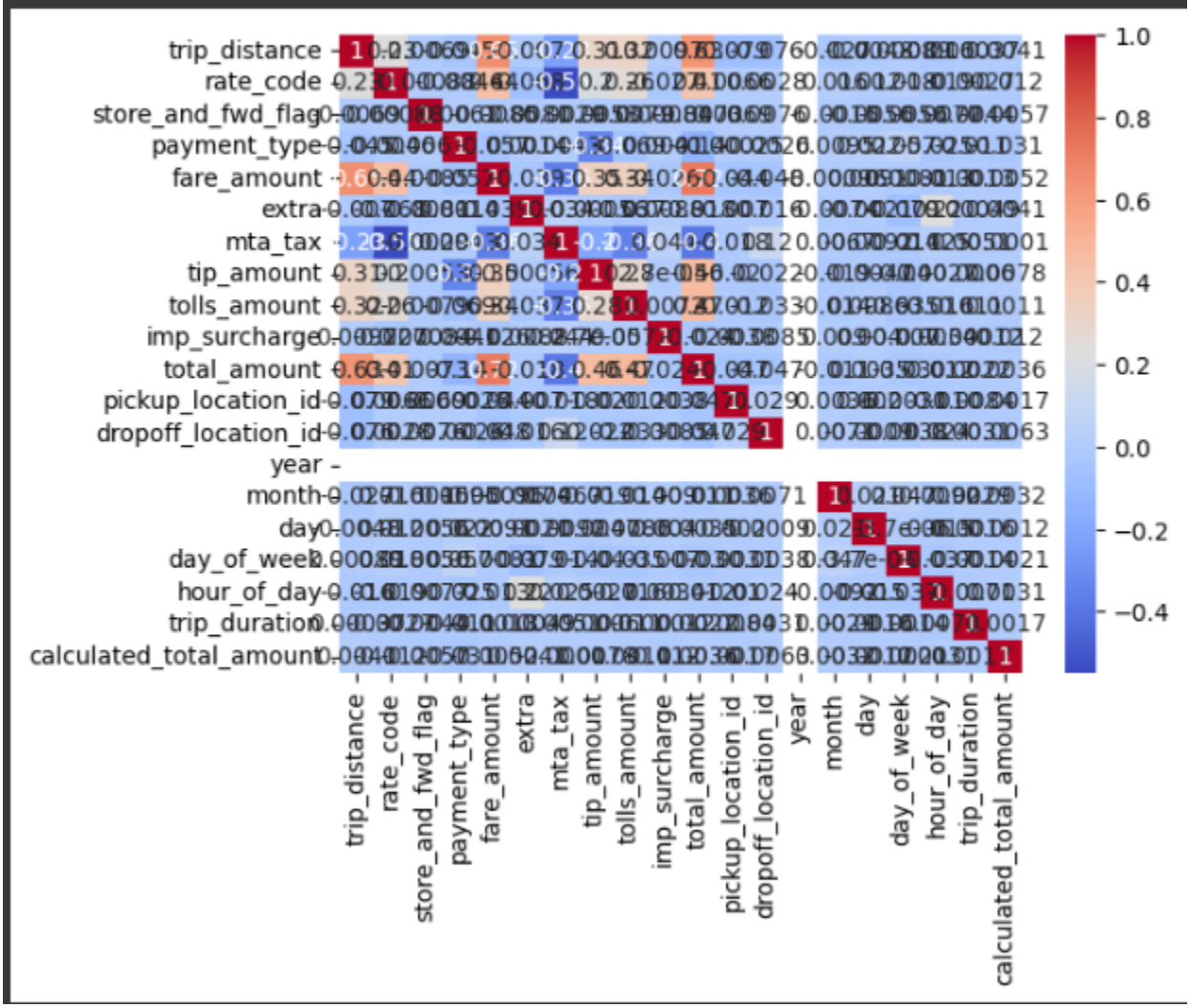


➔ Using a pair plot to show relationships between the variables in a dataset.



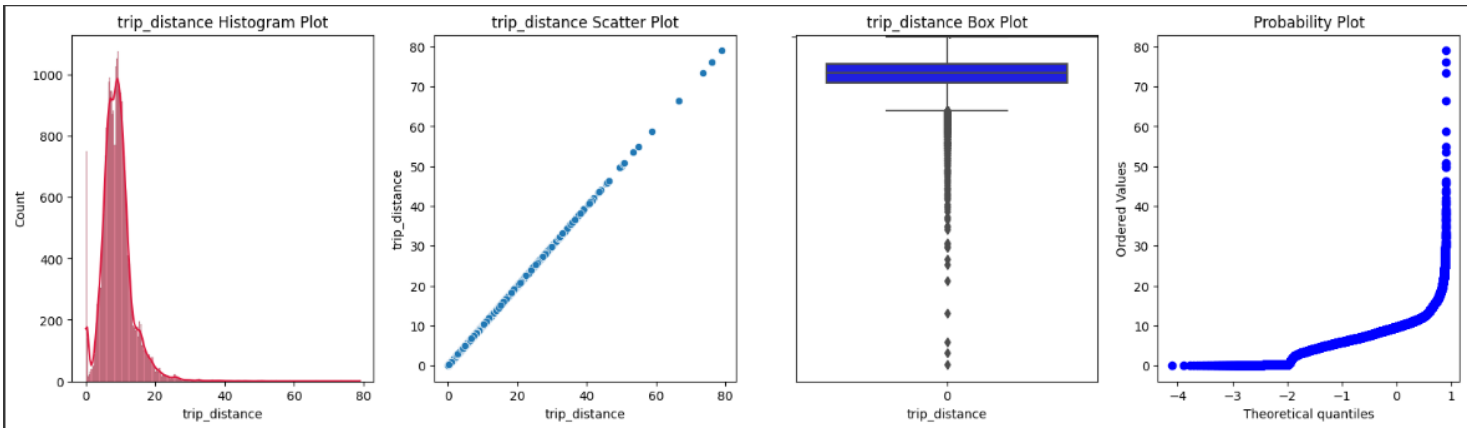
➔ Using Heatmap to see the correlation between features in the dataset.

```
# Heatmaps are used to show the correlation between variables in a dataset.
sns.heatmap(train_df.corr(), annot=True, cmap="coolwarm")
plot.show()
```



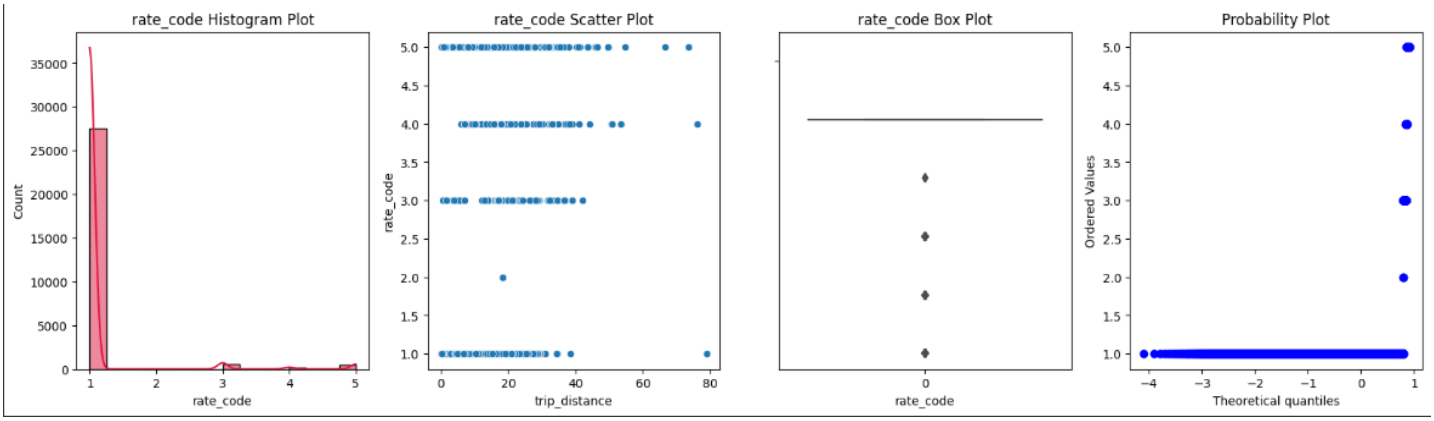
Inferences from Data Visualization :-

Trip_Distance :-



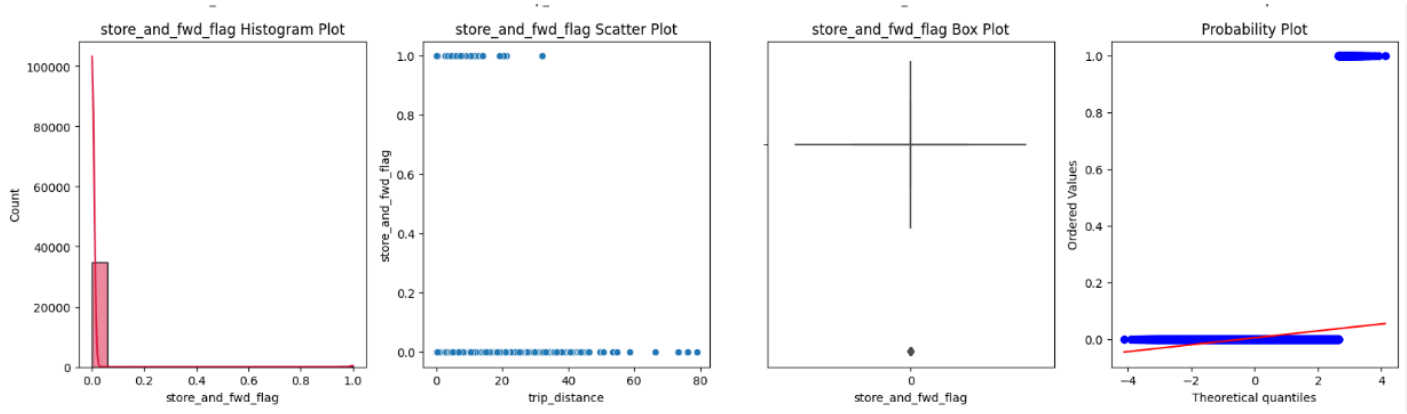
➔ We can infer from the plots of the trip_distance feature that around 1000 people traveled in taxis a distance of 10 miles, and very few people traveled more than 20 miles. The best range of distance people used for traveling in taxis is (2,20) miles, we can see that there are many outliers too indicating people not only traveled less than 25 miles, but very few people traveled more than it.

Rate_code :-



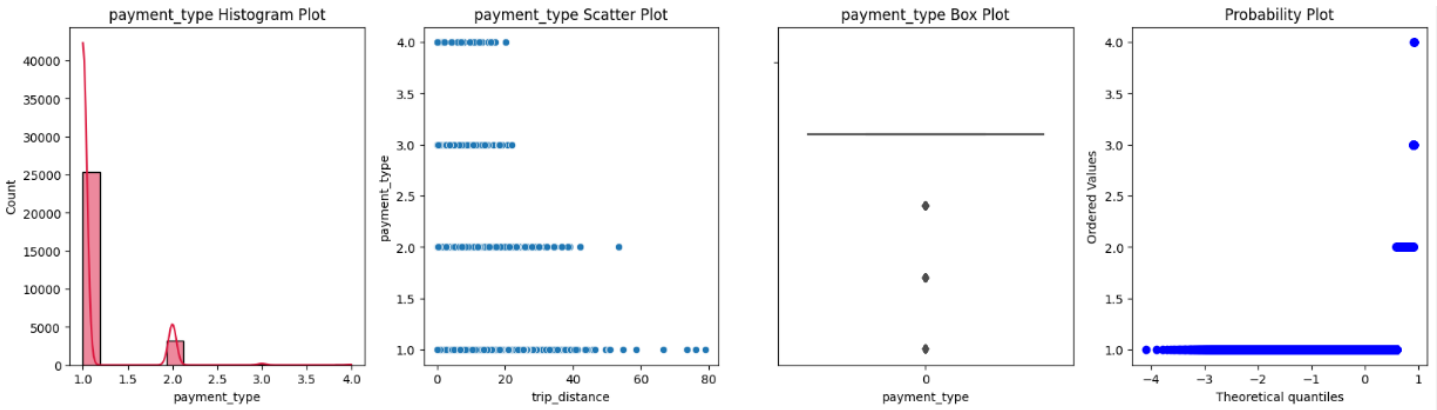
➔ We can infer from the plots that the people who have traveled less than 40 miles, around 25000 people, rated it as 1, and people who have traveled a distance of 46 miles, around 1000 rated the trip as 3, and people who have traveled a distance of above 50 miles but less than 60 miles, around 500 rated it as 4, and very few people who traveled near to 80 miles, around 1000 people, rated it as 5

Store_and_fwd_flag :-



➔ This indicates that the trip memory is present in the server or not before being sent to the vendor. We can see that the travel distance less than 30 miles is some time stored in the memory and is being forwarded.

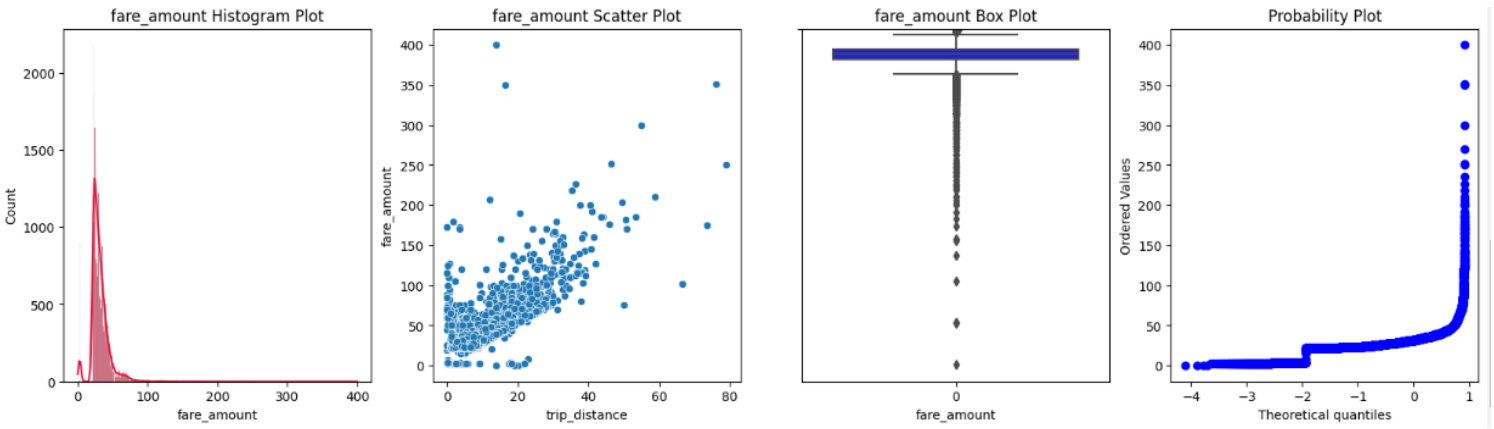
Payement_type :-



➔ It is a numeric code indicating the type of payment, 1 - Credit Card, 2 - Cash, 3 - No Charge, 4 - Dispute, 5 - Unknown, 6 - Voided Trip.

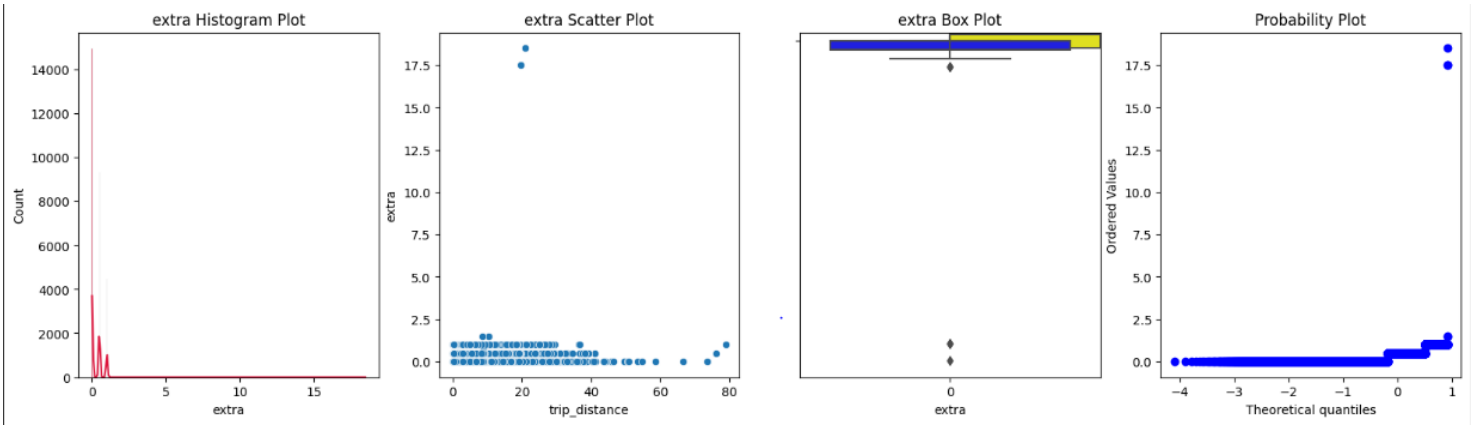
➔ We can infer from the plots that around 25000 people paid the taxi charge through credit card, and nearly 4000 people paid through cash, and very few were charged nothing for traveling a distance of 20 miles, and few had a dispute regarding payment type for people who traveled nearly 20 miles. And there were no trips with unknown payment mode and void trip.

Fare_amount :-



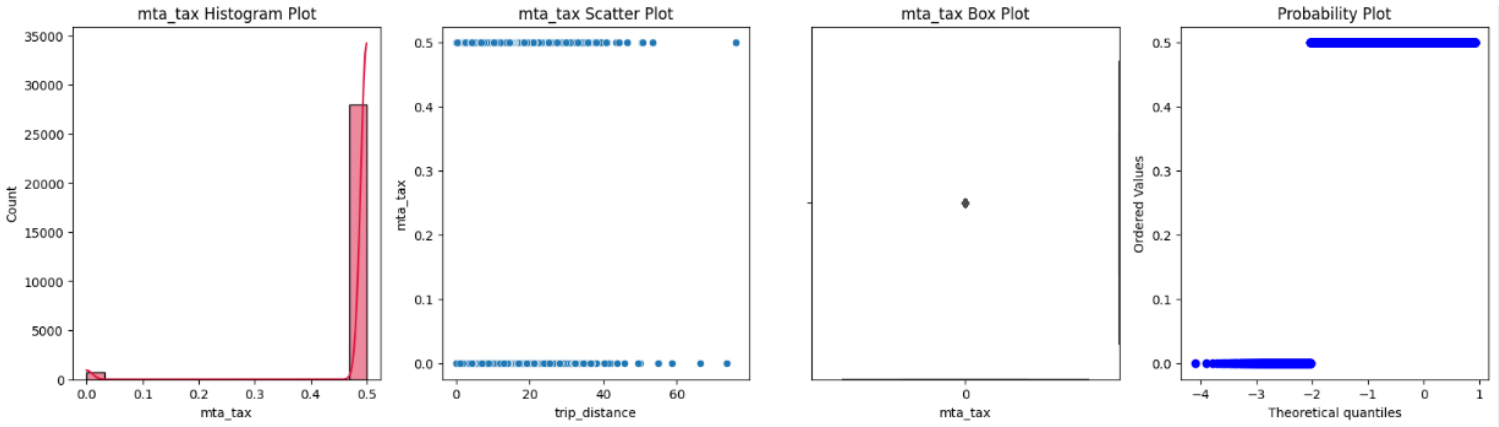
➔ We can infer from the plots that around 1300 people who traveled paid an amount of 40 dollars, and people who paid an amount of 60 dollars are around 500, and we can see clearly from scatter plot that most of the trip distances which varied from 20-40 miles costed in the range 60-100 dollars, and people who traveled in the range of 0-20 miles costed less than 80 dollars, and few travel distance of 40-60 miles costed 150-250 dollars.

Extra :-



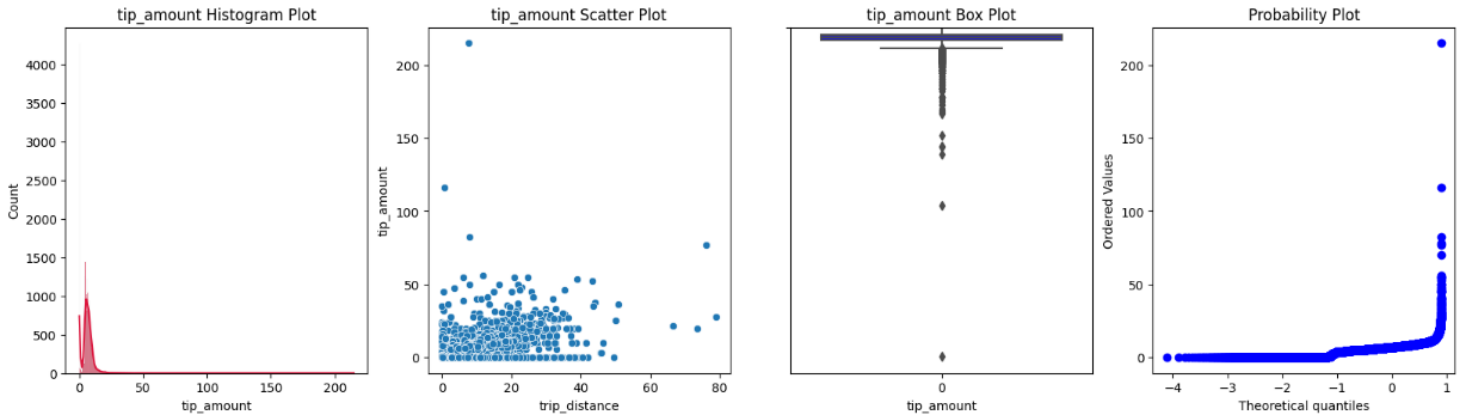
➔ This indicates the extra miscellaneous charges for the trip or payment. Only people who traveled a distance till 40 miles, paid an extra charge of at most 2 dollars.

Mta_tax :-



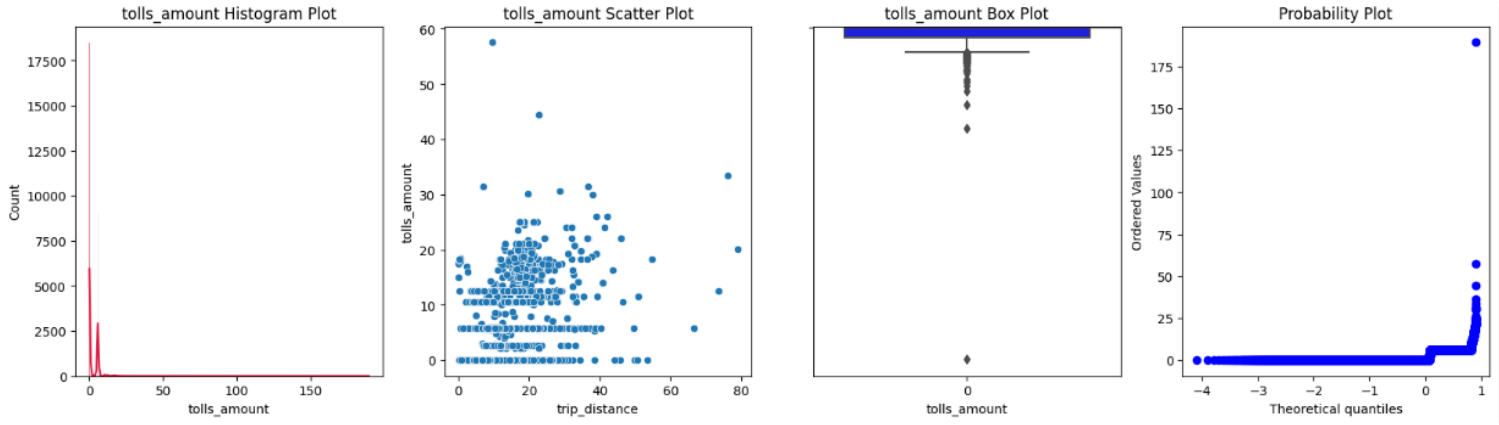
➔ This feature is 0.50 dollars that is automatically triggered based on the metered rate in use. From the plots we can infer that people who traveled a distance of 50 miles, i.e 25000 people paid an extra Mta_tax of 0.5 dollars.

Trip_amount :-



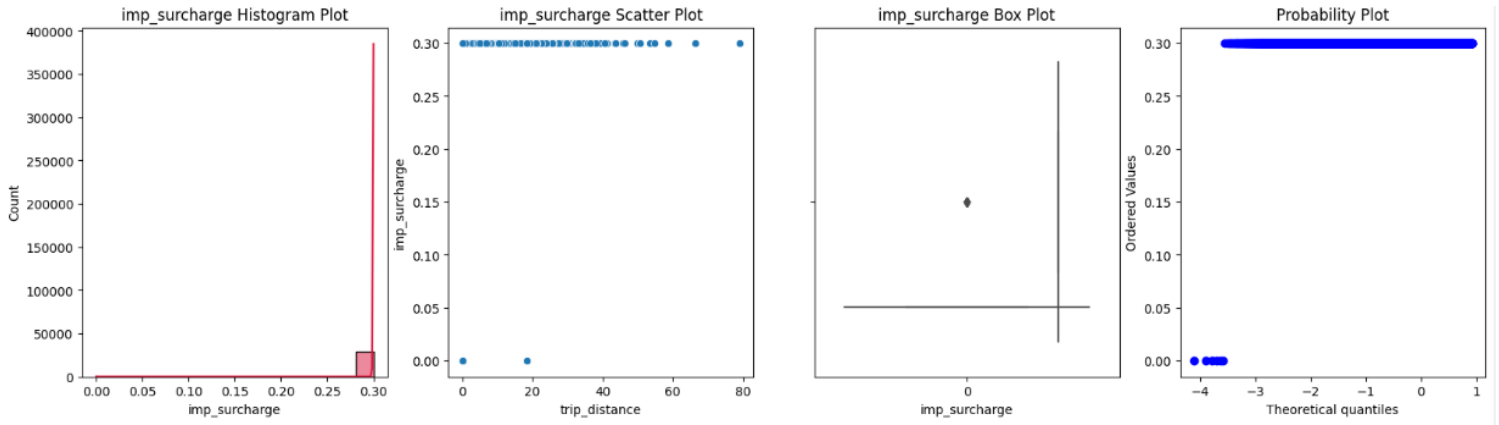
➔ This is the amount credited to the driver for all the credit card transactions being made. From the plots we can infer that around 1000 people who traveled a distance of 40 miles paid the Trip_amount of at most 50 dollars to the driver for the credit card transactions being made.

Tolls_amount :-



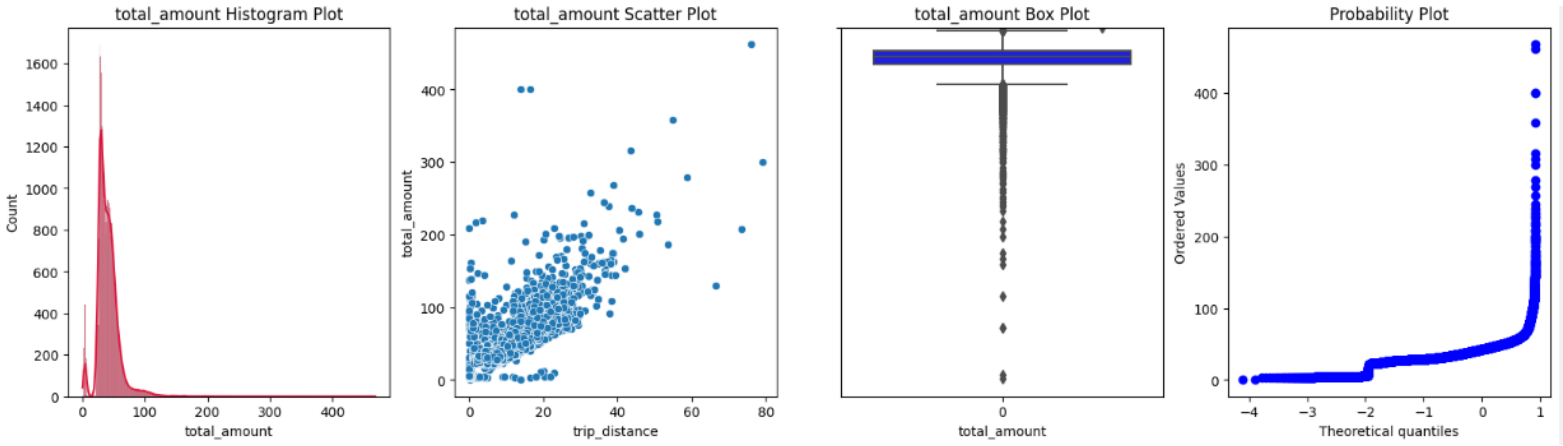
➔ This indicates the total amount of all tolls paid during a ride, from the plots we can infer that, 6000 people who traveled a distance of 60 miles paid a total tolls_amount around 30 dollars.

Imp_surcharge :-



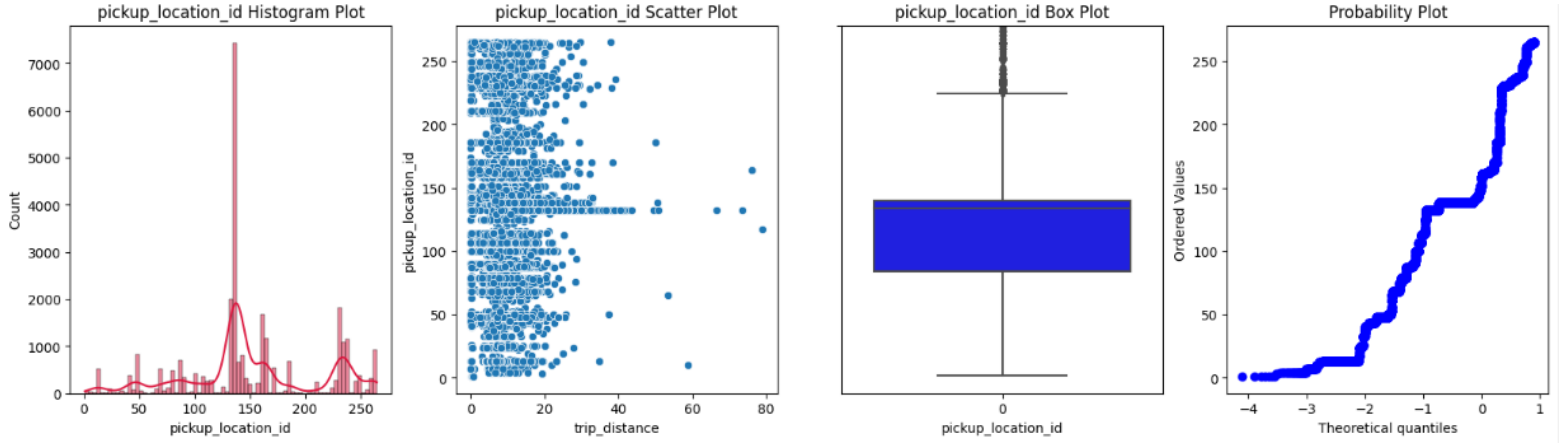
➔ This feature tells all the people we imposed Imp_suracharge of 0.3 dollars.

Total_amount :-



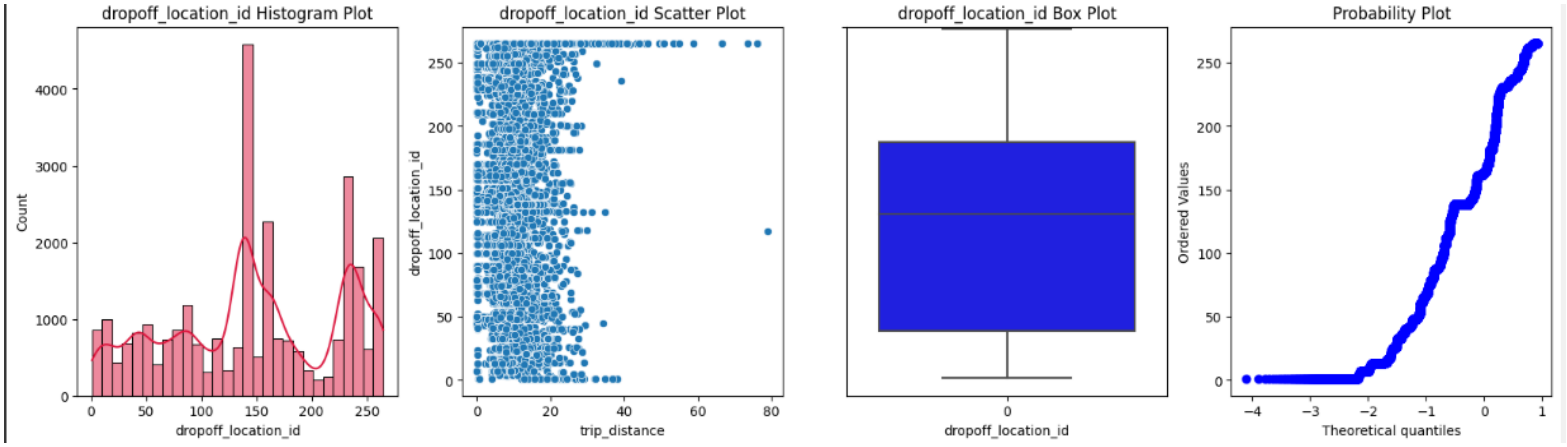
➔ The tells the total amount being charged to the travelers excluding cash tips, from the plots we can infer that people who traveled less than 20 miles, amount totalled to be 100 dollars, and people who traveled less than 40 miles, amount totalled to be 200 dollars at max, and for few outliers the total amounts came around 200-400 dollars.

Pickup_location_id :-



➔ From the plots we can infer that around people who traveled distance of 20 miles i.e less than 1000 people had the pickup_location_id to less than 50 ranging from 0, and the majority registered location id was 130, and few other major pickup_location_id were 140,150, 230, 260.

Dropoff_location_id :-

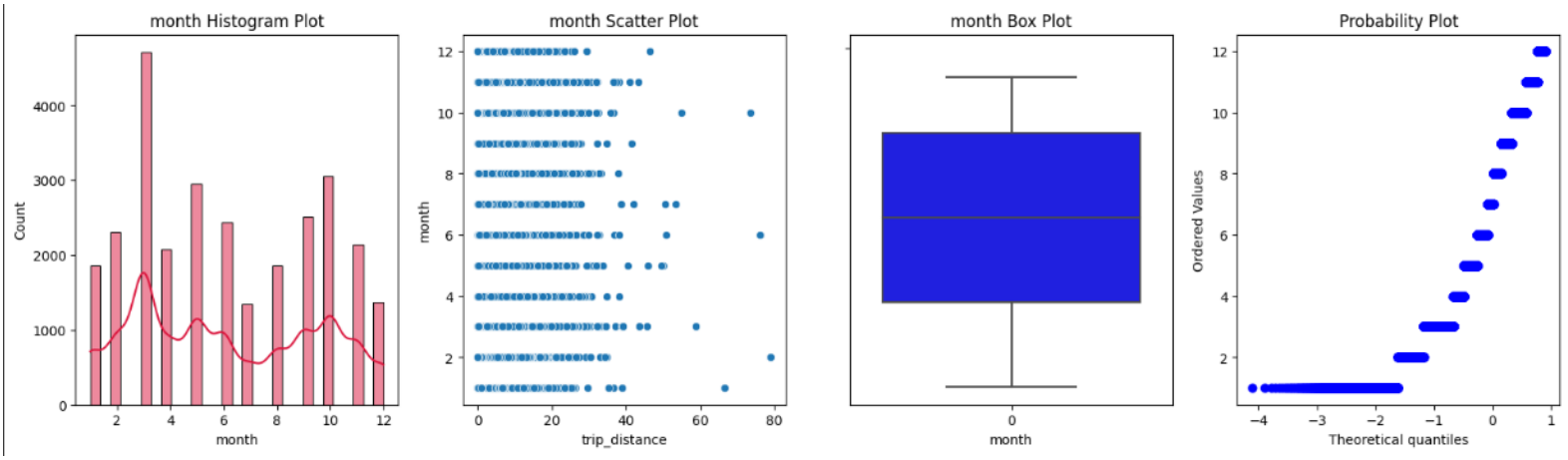


➔ Here in the plots we can see that dropoff_location_id was evenly distributed over various destination location, but here also we can see that majority dropoff location id is 150, 160, and many people's drop off location was also 230 or 260 , but we can see that most of the dropoff location id was less than the average dropoff location id, and a few were above the average's dropoff location id.

Year :-

➔ This feature indicates in which year people used taxi as their mode of trips. From the plots we can see that most of the people traveled in the year 2017 and 2018.

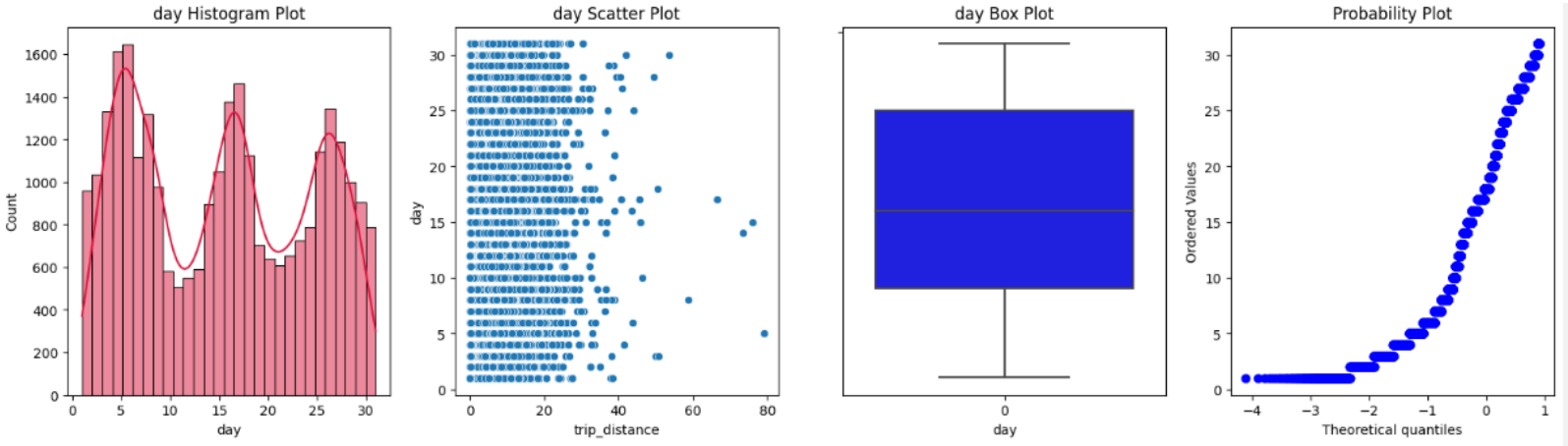
Month :-



➔ This feature tells in which month how many people traveled how much distance, so from the plots we can infer as follows :-

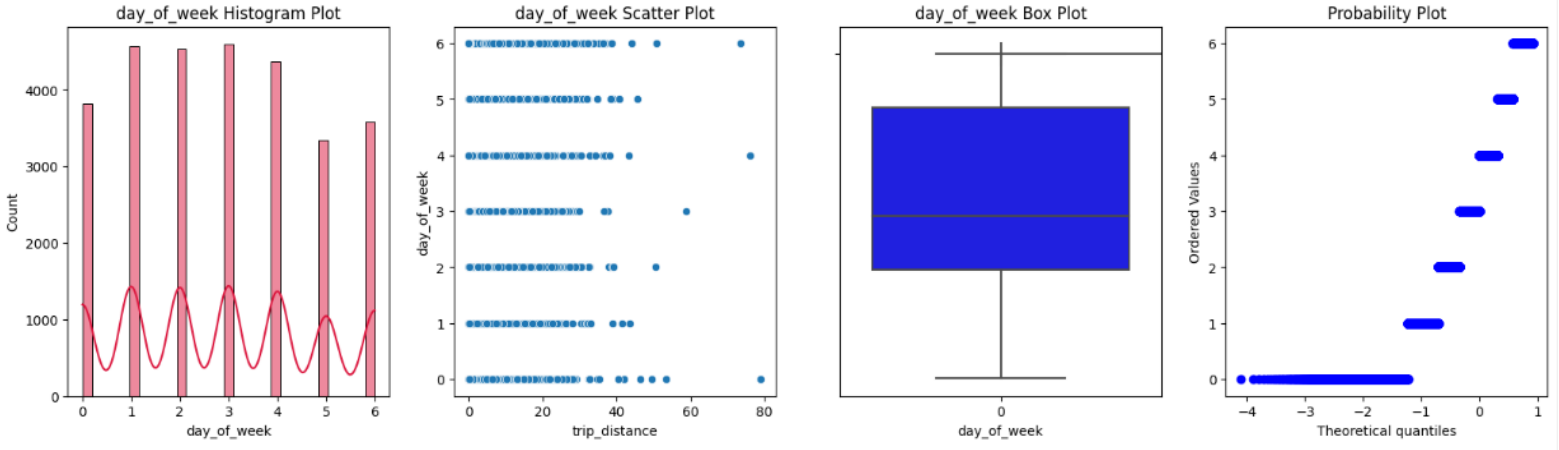
➔ January - 1900 people traveled in a taxi a distance of 40 miles at maximum.
➔ February - 2500 people traveled in a taxi a distance of 35 miles at maximum.
➔ March - 5000 People traveled in a taxi a distance of 48 miles at maximum.
➔ April - 2000 People traveled in a taxi a distance of 40 miles at maximum.
➔ May - 3000 People traveled in a taxi a distance of 50 miles at maximum.
➔ June - 2500 People traveled in a taxi a distance of 40 miles at maximum
➔ July - 1200 People traveled in a taxi a distance of 30 miles at maximum.
➔ August - 1800 People traveled in a taxi a distance of 35 miles at maximum.
➔ September - 2500 People traveled in a taxi a distance of 30 miles at maximum.
➔ October - 3000 People traveled in a taxi a distance of 40 miles at maximum.
➔ November - 2000 People traveled in a taxi a distance of 45 miles at maximum.
➔ December - 1300 People traveled in a taxi a distance of 25 miles at maximum.

Day :-



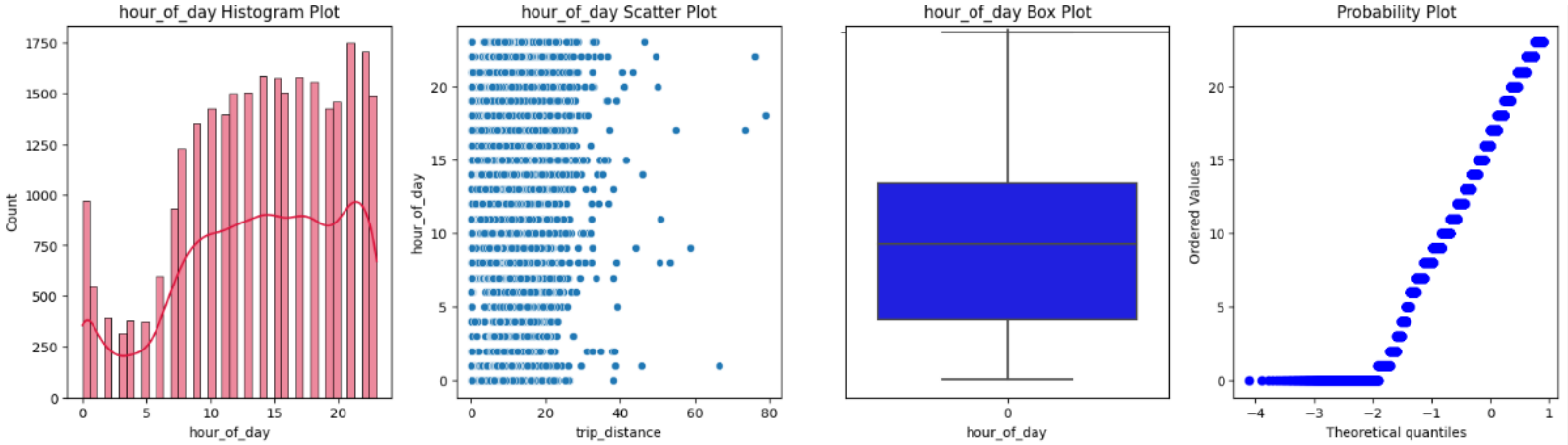
➔ This feature tells about the date of the month on which a taxi trip is preferred by people. We can see that taxi was preferred by people most on 4,5, and then comes on the dates 3, 7, 15, 16, 17, 24, 25, 26, 27. We can also see that taxis were least preferred on dates 9, 10, 11, 12, 19, 20, 21, 22.

Day_of_week :-



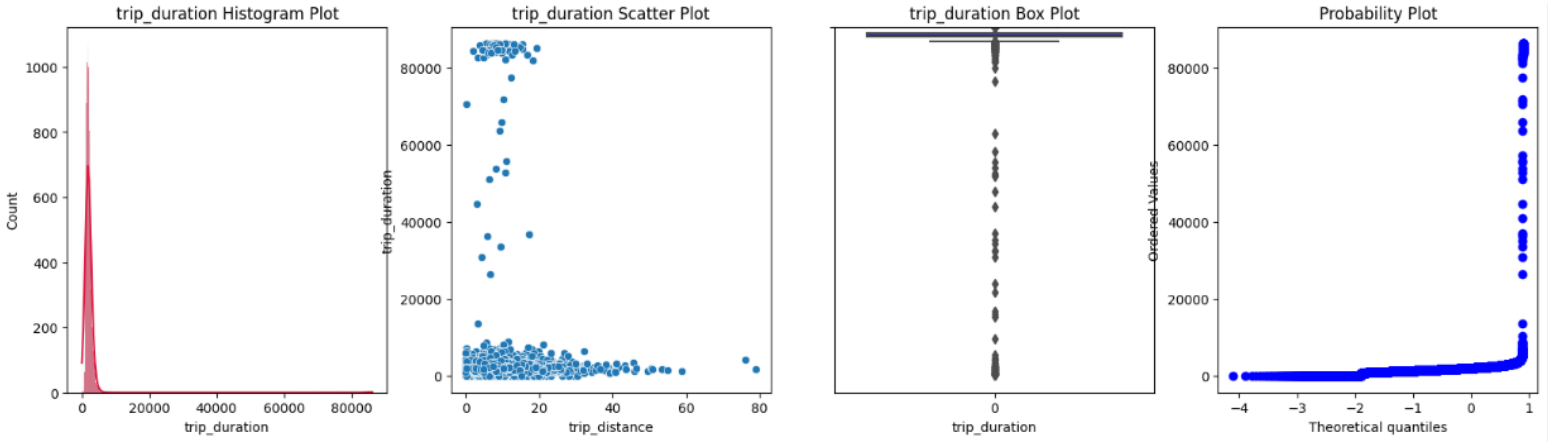
➔ From the plots related to these days of the week taxi being preferred by people, we can infer that taxis were equally preferred by people on monday, tuesday, wednesday, thursday, and there is decrease no. of taxi's preferred on friday, and equal during sunday and saturday.

Hour_of_day :-



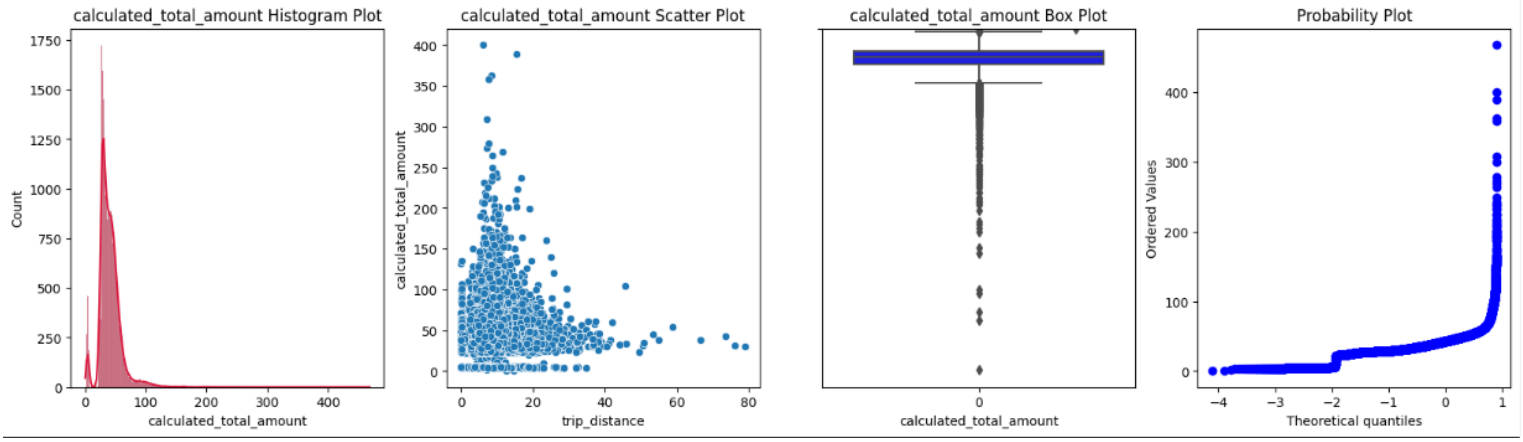
➔ Than in the early morning hours we can see that people used taxi services majorly after 8 Am, and increased till 8 Pm, and suddenly reached the peak at 9-10 Pm, If our guess is correct, we can say that taxi services are used by people only during the working hours mostly of a day. And people used it to travel at max 40 miles.

Trip_duration :-



➔ This feature gives the total time traveled by people in seconds. From the plots we can see that most of the people used the taxi service only for around 1000-5000 seconds, and rarely people used for 40000-80000 seconds, and few others used the taxi services for above 80000 seconds of their journey.

Calculated_total_amount :-



➔ From the plots we can see that the amount at max is calculated to be 250 on an average, and rare values show people paid 300,350, and 400. And we can also see from the scatterplot that most of the people whose travel costs less than 250 traveled less than 40 miles.

Normalization and Hypothesis Testing :-

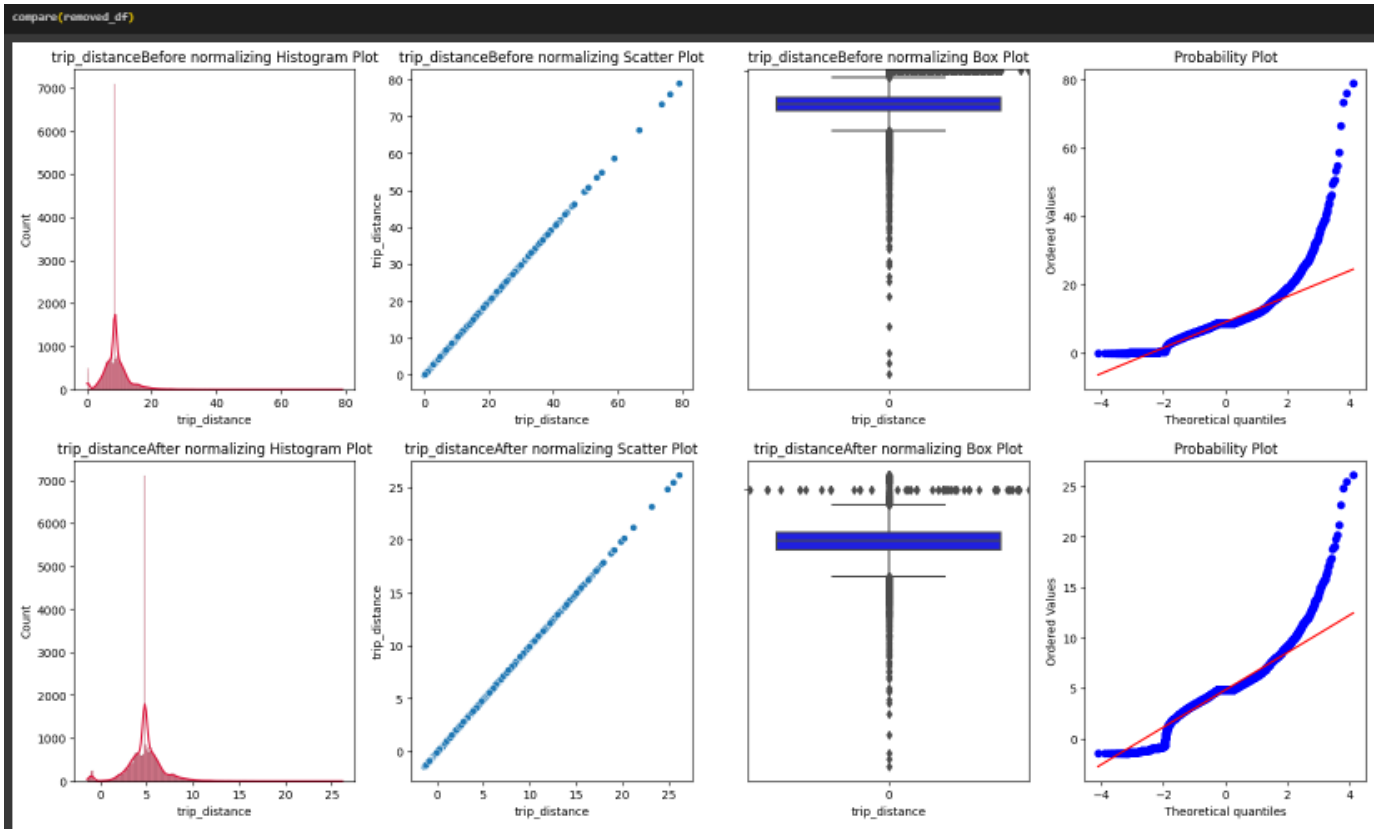
➔ Removing the unwanted columns whose values could not be normalized.

Normalizing :-

```
# Removing the columns whose values are wrong and having negative values
columns_to_remove = ['store_and_fwd_flag', 'extra', 'mta_tax','tip_amount','tolls_amount','imp_surcharge','year','day_of_week','hour_of_day']
removed_df = train_df.drop(columns_to_remove, axis=1)
```

➔ Comparing the Plots before and after Normalization.

```
compare(removed_df)
```



Hypothesis Testing :-

Test the Hypothesis for The feature Trip_distance.

H0: μ = 10 miles vs H1: μ != 10 miles

- **level of significance α = 5%**

```python
# function for sigma being not known.
def func_sigma_not_known(func_data, func_alpha, func_mu_not):
  if(len(func_data) > 25):
    # sigma_2 = np.std(func_data)**2
    z_cal = (np.mean(func_data) - func_mu_not)/(np.std(func_data)/np.sqrt(len(func_data)))
    p_val = 2 *(1 - sp.stats.norm.cdf(np.abs(z_cal)))
    if p_val < func_alpha:
      print("Reject H0")
    else:
      print("Donot reject H0")
  else:
    print("sairam")
    t_cal = (np.mean(func_data) - func_mu_not)/(np.std(func_data)/np.sqrt(len(func_data)))
    t_val = 2 * (1 - sp.stats.t.cdf(np.abs(t_cal),len(func_data)-1))
    print(t_cal,t_val)
    if t_val < func_alpha:
      print("Reject H0")
    else:
      print("Donot reject H0")
```

➔ The above function checks whether to Reject or Do not Reject our H0.

```
train_df['trip_distance'].describe()
```

```
count    35000.000000
mean         9.002194
std          4.051970
min          0.010000
25%          6.950000
50%          8.600000
75%         10.400000
max         79.010000
Name: trip_distance, dtype: float64
```

```
[127] func_sigma_not_known(train_df['trip_distance'],0.05,10)

      Reject H0
```

```
func_sigma_not_known(train_df['trip_distance'],0.05,9)

Donot reject H0
```