edWisor Project-1

Cab Fare Prediction

Submitted By

Naveen Thomas

INDEX

1.Problem Statement

You are a cab rental start-up company. You have successfully run the pilot project and now want to launch your cab service across the country. You have collected the historical data from your pilot project and now have a requirement to apply analytics for fare prediction. You need to design a system that predicts the fare amount for a cab ride in the city.

Dataset:-

1) train_cab.csv (16067 observations, 7 variables)
2) test.csv (9914 observations, 6 variables)

Number of attributes:

1) pickup_datetime - timestamp value indicating when the cab ride started.
2) pickup_longitude - float for longitude coordinate of where the cab ride started.
3) pickup_latitude - float for latitude coordinate of where the cab ride started.
4) dropoff_longitude - float for longitude coordinate of where the cab ride ended.
5) dropoff_latitude - float for latitude coordinate of where the cab ride ended.
6) passenger_count - an integer indicating the number of passengers in the cab ride.

2.Assumption

- Assuming that test.csv dataset which is given for dependent variable prediction is perfect. We will not perform data cleaning on it i.e. no observations will be removed from test.csv dataset. But we can add/remove columns (feature engineering) to match with the train_cab.csv dataset.
- Assuming that there is no round trip, no waiting charge, no cancellation fee(if using an app).

3.Test Cases

We will take 4 different cases in data pre-processing.

1) case1: df_1 --> drop the observations which are non-sensible and remove all outliers based on boxplot.
2) case2: df_2 --> drop the observations which are non-sensible, remove all outliers decided by user based on observations.
3) case3: df_3 --> make the observations which are non-sensible and make all outliers (based on boxplot) to NaN and impute them.
4) case4: df_4 --> make the observations which are non-sensible and make all outliers (decided by user based on observations) to NaN and impute them.

4.Data Cleaning

Data type of train_cab.csv:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16067 entries, 0 to 16066
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   pickup_datetime    16067 non-null  object 
 1   pickup_longitude   16067 non-null  float64
 2   pickup_latitude    16067 non-null  float64
 3   dropoff_longitude  16067 non-null  float64
 4   dropoff_latitude   16067 non-null  float64
 5   passenger_count    16012 non-null  float64
 6   fare_amount        16043 non-null  object 
dtypes: float64(5), object(2)
memory usage: 878.8+ KB
```

Data type of test.csv:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9914 entries, 0 to 9913
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   pickup_datetime    9914 non-null   object 
 1   pickup_longitude   9914 non-null   float64
 2   pickup_latitude    9914 non-null   float64
 3   dropoff_longitude  9914 non-null   float64
 4   dropoff_latitude   9914 non-null   float64
 5   passenger_count    9914 non-null   int64  
dtypes: float64(4), int64(1), object(1)
memory usage: 464.8+ KB
```

4.1 fare_amount (target variable)

Since fare_amount is the target variable , whichever fare_amount observation that are non-sensible will be removed. We won't be changing those observations to NaN and impute them.

1) **fare_amount having 430- value**

First I tried to covert fare_amount from object to float datatype. It was showing me this error:

```
Error: could not convert string to float: '430-'
```

From this error we will come to know that in one of the observations the fare_amount value is '430-'. Changed the 430- value to 430. After that converted successfully the fare_amount datatype to float.

2) **fare_amount <=0**

```
count    16043.000000
mean        15.040871
std        430.459997
min         -3.000000
25%          6.000000
50%          8.500000
75%         12.500000
max      54343.000000
Name: fare_amount, dtype: float64
```

	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	fare_amount
2039	2010-03-09 23:37:10 UTC	-73.789450	40.643498	-73.788665	40.641952	1.0	-2.9
2486	2015-03-22 05:14:27 UTC	-74.000031	40.720631	-73.999809	40.720539	1.0	-2.5
10002	2010-02-15 14:26:01 UTC	-73.987115	40.738808	-74.005911	40.713960	1.0	0.0
13032	2013-08-30 08:57:10 UTC	-73.995062	40.740755	-73.995885	40.741357	4.0	-3.0

Fare_amount value which is <=0 is removed.
No. of observations removed= 4
No. of observations remaining= 16063

4.2 pickup_datetime

Changed the datatype of pickup_datetime from object to datetime datatype.

4.3 pickup_longitude and dropoff_longitude

Checked whether their value ranges only between -180 degree to 180 degree.

4.4 pickup_latitude and dropoff_latitude

Checked whether their value ranges only between -90 degree to 90 degree.

```
count    16063.000000
mean        39.914527
std          6.827426
min        -74.006893
25%         40.734935
50%         40.752615
75%         40.767382
max        401.083332
Name: pickup_latitude, dtype: float64
```

This is the summary of pickup_latitude. Since the max value of this variable>90, we need to do data cleaning for this variable.

	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	fare_amount
5684	2011-07-30 11:15:00+00:00	-73.947235	401.083332	-73.951392	40.778927	1.0	3.3

Here our 2 out of 4 cases starts. Our main dataset is divided into 2. In one case (case1), we will remove this non-sensible value and in the other case (case 3), we will set this non-sensible value to NaN and impute it. But still our case 1 =case 2 and case 3=case 4 since we didn't reach the outlier part.

4.4.1 case 1 & case 2

No. of observations removed= 1
No. of observations remaining= 16062

4.4.2 case 3 & case 4

No. of observations put to NaN= 1
No. of observations remaining= 16063

4.5 passenger_count

```
count    9914.000000
mean        1.671273
std         1.278747
min         1.000000
25%         1.000000
50%         1.000000
75%         2.000000
max         6.000000
Name: passenger_count, dtype: float64
```

This is the summary of passenger_count of test.csv dataset.

4.5.1 case 1 & case 2

```
count    16007.000000
mean         2.625390
std         60.853618
min          0.000000
25%          1.000000
50%          1.000000
75%          2.000000
max       5345.000000
Name: passenger_count, dtype: float64
```

Here max value of passenger_count = 5345 so we need to do data cleaning.

1) **Checking for passenger_count>6 with sorting of passenger_count in ascending order**

	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	fare_amount
1043	2012-08-22 22:08:29+00:00	-73.973573	40.760184	-73.953564	40.767392	35.0	5.7
8628	2012-12-10 22:28:00+00:00	-73.955445	40.670232	-74.004795	40.731477	43.0	20.0
1242	2011-10-16 00:22:00+00:00	-73.981095	40.738160	-73.990587	40.740105	43.0	5.3
8403	2010-08-25 11:41:00+00:00	0.000000	0.000000	0.000000	0.000000	53.0	6.9
1007	2010-12-14 14:46:00+00:00	-73.969157	40.759000	-73.968763	40.764617	53.0	3.7
413	2013-09-12 11:32:00+00:00	-73.982060	40.772705	-73.956213	40.771777	55.0	NaN
8442	2009-03-28 22:00:00+00:00	-73.982413	40.751320	-73.971292	40.748502	58.0	5.7
8568	2011-12-03 03:21:00+00:00	-73.993718	40.762039	-73.977527	40.734024	87.0	12.5
233	2011-07-24 01:14:35+00:00	0.000000	0.000000	0.000000	0.000000	236.0	8.5
1107	2009-08-08 21:50:50+00:00	-73.988977	40.721068	-73.982368	40.732064	345.0	4.9
386	2009-08-21 19:35:05+00:00	-73.960853	40.761557	-73.976335	40.748361	354.0	8.1

After 6 we can see a huge increase in passenger_count. Hence we remove those observations having passenger_count >6
No. of observations removed= 20
No. of observations remaining= 16042

2) **Checking for any decimal values in passenger_count**

```
1.00    11255
2.00     2322
5.00     1045
3.00      676
4.00      328
6.00      302
0.00       57
1.30        1
0.12        1
Name: passenger_count, dtype: int64
```

Here we removed those observations having passenger_count a decimal value
No. of observations removed= 59
No. of observations remaining= 15983

4.5.2 case 3 & case 4

```
count    16008.000000
mean         2.625289
std         60.851718
min          0.000000
25%          1.000000
50%          1.000000
75%          2.000000
max       5345.000000
Name: passenger_count, dtype: float64
```

1) **Checking for passenger_count>6 with sorting of passenger_count in ascending order**

	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	fare_amount
1043	2012-08-22 22:08:29+00:00	-73.973573	40.760184	-73.953564	40.767392	35.0	5.7
8629	2012-12-10 22:28:00+00:00	-73.955445	40.670232	-74.004795	40.731477	43.0	20.0
1242	2011-10-16 00:22:00+00:00	-73.981095	40.738160	-73.990587	40.740105	43.0	5.3
8404	2010-08-25 11:41:00+00:00	0.000000	0.000000	0.000000	0.000000	53.0	6.9
1007	2010-12-14 14:46:00+00:00	-73.969157	40.759000	-73.968763	40.764617	53.0	3.7
413	2013-09-12 11:32:00+00:00	-73.982060	40.772705	-73.956213	40.771777	55.0	NaN
8443	2009-03-28 22:00:00+00:00	-73.982413	40.751320	-73.971292	40.748502	58.0	5.7
8569	2011-12-03 03:21:00+00:00	-73.993718	40.762039	-73.977527	40.734024	87.0	12.5
233	2011-07-24 01:14:35+00:00	0.000000	0.000000	0.000000	0.000000	236.0	8.5

Here also we can see a huge increase in passenger_count after 6. We will set these values to NaN. We will impute them at the missing value section.

No. of observations set to NaN= 20
No. of observations remaining= 16063

2) **Checking for any decimal values in passenger_count**

```
1.00    11256
2.00     2322
5.00     1045
3.00      676
4.00      328
6.00      302
0.00       57
1.30        1
0.12        1
Name: passenger_count, dtype: int64
```

Those decimal passenger_count values are set to NaN. Imputation will be done at the missing value section.

No. of observations set to NaN=59
No. of observations remaining= 16063

4.6 Cleaning of pickup_longitude, pickup_latitude, dropoff_longitude, dropoff_latitude variables combined

Since latitude=0 and longitude=0 is located in ocean we can remove them in case 1 and case 2.

Since both latitude and longitude which are zero will be converted to NaN, imputation will be inaccurate. So, we will remove them in case 3 and case 4 also.

4.6.1 case 1 & case 2

No. of observations removed= 320
No. of observations remaining= 15663

4.6.2 case 3 & case 4

No. of observations removed= 324
No. of observations remaining= 15739

5.Missing value

	Count	Percentage
passenger_count	55	0.351146
fare_amount	21	0.134074
pickup_datetime	1	0.006384
pickup_longitude	0	0.000000
pickup_latitude	0	0.000000
dropoff_longitude	0	0.000000
dropoff_latitude	0	0.000000

Missing value for case 1 & case 2

	Count	Percentage
passenger_count	130	0.825974
fare_amount	22	0.139780
pickup_datetime	1	0.006354
pickup_latitude	1	0.006354
pickup_longitude	0	0.000000
dropoff_longitude	0	0.000000
dropoff_latitude	0	0.000000

Missing value for case 3 & case 4

5.1 fare_amount

Since fare_amount is the target variable we are only dropping the missing values.

5.1.1 case 1 & case 2

No. of observations removed= 21
No. of observations remaining= 15642

5.1.2 case 3 & case 4

No. of observations removed= 22
No. of observations remaining= 15717

5.2 passenger_count

5.2.1 case 1 & case 2

No. of observations removed= 55
No. of observations remaining= 15587

5.2.2 case 3 & case 4

We set the 34^{th} observation of passenger_count (which is known to us) to NaN. Then performed mean, median, mode and knn imputation.

Actual value for 34th obs = 6
Mean imputation = 2
Median imputation = 1
Mode imputation = 1
KNN imputation = 4
Based on the above observations KNN performs the best. So we will choose KNN to impute for passenger_count.

5.3 pickup_datetime

Since we can derive more new features out of pickup_datetime, it's better to drop the missing value of pickup_datetime which is one in number.

5.3.1 case 1 & case 2

No. of observations removed= 1
No. of observations remaining= 15586

5.3.2 case 3 & case 4

No. of observations removed= 1
No. of observations remaining= 15716

5.4 pickup_latitude

5.4.1 case 3 & case 4

We set the 100^{th} observation of pickup_latitude (which is known to us) as NaN. Then performed mean, median and knn imputation.

Actual value for 100th obs = 40.74732
Mean imputation = 40.6898
Median imputation = 40.7532
KNN imputation = 40.7476
Based on the above observations KNN performs the best. So we will choose KNN to impute for pickup_latitude.

5.5 Imputation of missing values

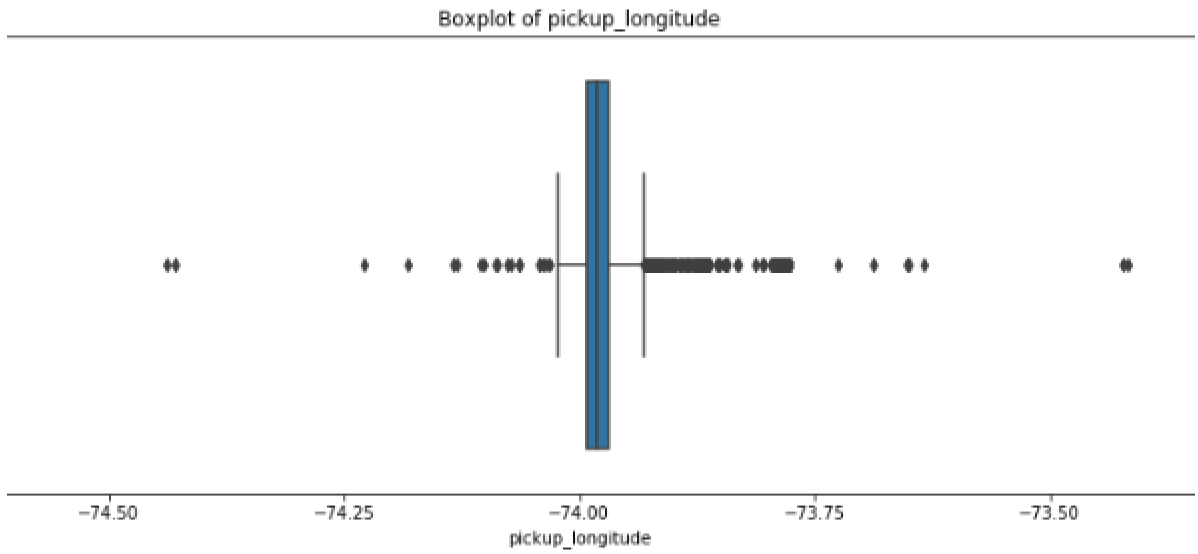
All the missing values are imputed by KNN as it performed best.

6.Outlier Analysis

Till now our case 1 = case 2 and case 3 = case 4. But in outlier analysis case 1 won't be equal to case 2 and case 3 won't be equal to case 4.

6.1 pickup_longitude

6.1.1 case 1



iqr=q75-q25

min=q25-(iqr*1.5)
max=q75+(iqr*1.5)
min= -74.02885038125   max= -73.93157377125
pickup_longitude whose values > max and <min is treated as outlier according to the box plot method.

No. of observations removed= 804
No. of observations remaining= 14782

6.1.2 case 2

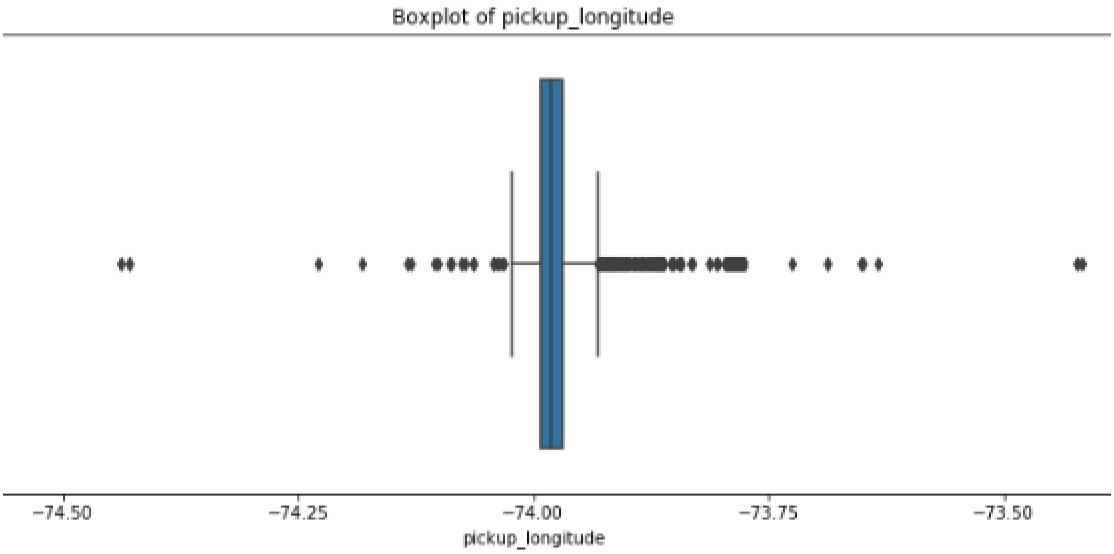
```
count    15586.000000
mean       -73.911174
std          2.665436
min        -74.438233
25%        -73.992372
50%        -73.982042
75%        -73.968052
max         40.766125
Name: pickup_longitude, dtype: float64
```

	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	fare_amount
6468	2013-05-22 15:33:00+00:00	40.766125	-73.983285	40.757417	-73.977962	2.0	13.0
9228	2013-07-13 12:31:00+00:00	40.764420	-73.992947	40.802437	-73.950730	1.0	15.5
7267	2013-05-22 10:54:00+00:00	40.760495	-73.973047	40.740367	-73.994392	1.0	13.0
2026	2013-05-24 14:54:00+00:00	40.751582	-73.986968	40.758867	-73.978353	2.0	5.0
4592	2013-05-22 06:28:00+00:00	40.748262	-73.991840	40.740372	-73.979010	1.0	6.5
9757	2013-05-22 20:15:00+00:00	40.736875	-74.006210	40.736887	-74.006377	6.0	52.0
8156	2013-05-24 00:32:00+00:00	40.729127	-74.006893	40.763367	-73.961550	1.0	15.0
3661	2013-06-20 04:28:00+00:00	40.719830	-73.988467	40.723305	-73.939430	1.0	11.0
1097	2012-10-11 00:21:00+00:00	-0.004093	0.033500	0.016852	0.017980	2.0	25.0

This is the observation where pickup_longitude>-73.137 and pickup_longitude sorted in descending order. We can see there is a huge increase in pickup_longitude after -73.137. So we set pickup_longitude values greater than -73.137 as an outlier. Now removing those observations.

No. of observations removed= 9
No. of observations remaining= 15577

6.1.3 case 3



Based on box plot method:
min= -74.02885237500001 max= -73.931615375

The observations having pickup_longitude whose values >max and values < min are considered as outliers and they are set to NaN.

No. of observations set to NaN= 808
No. of observations remaining= 15716

6.1.4 case 4

```
count    15716.000000
mean       -73.911723
std          2.654397
min        -74.438233
25%        -73.992389
50%        -73.982050
75%        -73.968079
max         40.766125
Name: pickup_longitude, dtype: float64
```

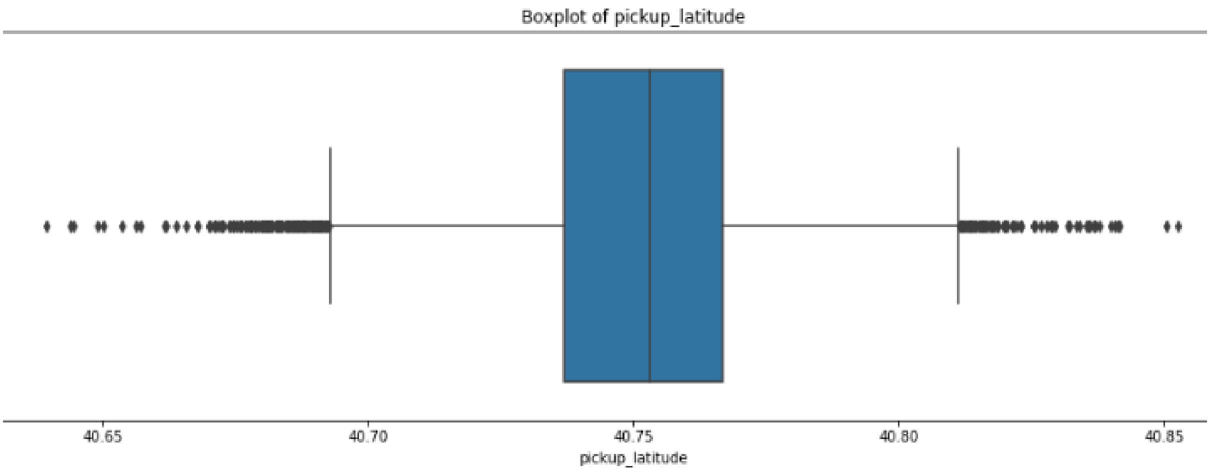
	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	fare_amount
6550	2013-05-22 15:33:00+00:00	40.766125	-73.983285	40.757417	-73.977962	2.0	13.0
9339	2013-07-13 12:31:00+00:00	40.764420	-73.992947	40.802437	-73.950730	1.0	15.5
7353	2013-05-22 10:54:00+00:00	40.760495	-73.973047	40.740367	-73.994392	1.0	13.0
2084	2013-05-24 14:54:00+00:00	40.751582	-73.986968	40.758867	-73.978353	2.0	5.0
4663	2013-05-22 06:28:00+00:00	40.748262	-73.991840	40.740372	-73.979010	1.0	6.5
9869	2013-05-22 20:15:00+00:00	40.736875	-74.006210	40.736887	-74.006377	6.0	52.0
8255	2013-05-24 00:32:00+00:00	40.729127	-74.006893	40.763367	-73.961550	1.0	15.0
3728	2013-06-20 04:28:00+00:00	40.719830	-73.988467	40.723305	-73.939430	1.0	11.0
1141	2012-10-11 00:21:00+00:00	-0.004093	0.033500	0.016852	0.017980	2.0	25.0

By seeing the observation , pickup_longitude >-73.137 is set as outlier.
Now setting those outliers as NaN.

No. of observations set to NaN= 9
No. of observations remaining= 15716

6.2 pickup_latitude

6.2.1 case 1



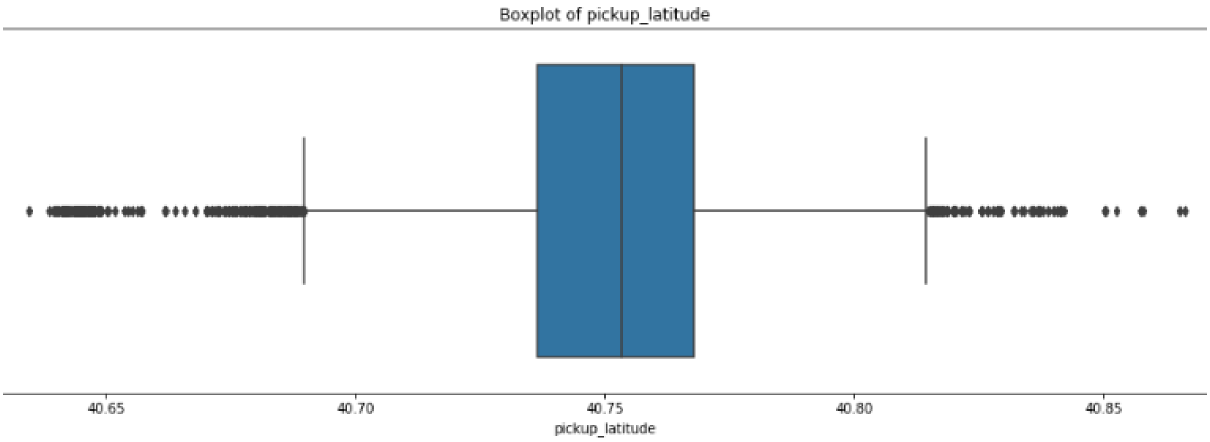
Based on boxplot:
min= 40.69270012499999 max= 40.811133125
No. of observations removed= 260
No. of observations remaining= 14522

6.2.2 case 2

```
count    15577.000000
mean        40.750921
std          0.038021
min         39.603178
25%         40.736588
50%         40.753350
75%         40.767807
max         41.366138
Name: pickup_latitude, dtype: float64
```

By seeing the observation, there is no outlier in pickup_latitude.

6.2.3 case 3



Based on boxplot:
min= 40.689670500000005 max= 40.81467649999999

No. of observations set to NaN= 514
No. of observations remaining= 15716

6.2.4 case 4

```
count    15716.000000
mean        40.689903
std          2.608731
min        -74.006893
25%         40.736548
50%         40.753294
75%         40.767799
max         41.366138
Name: pickup_latitude, dtype: float64
```

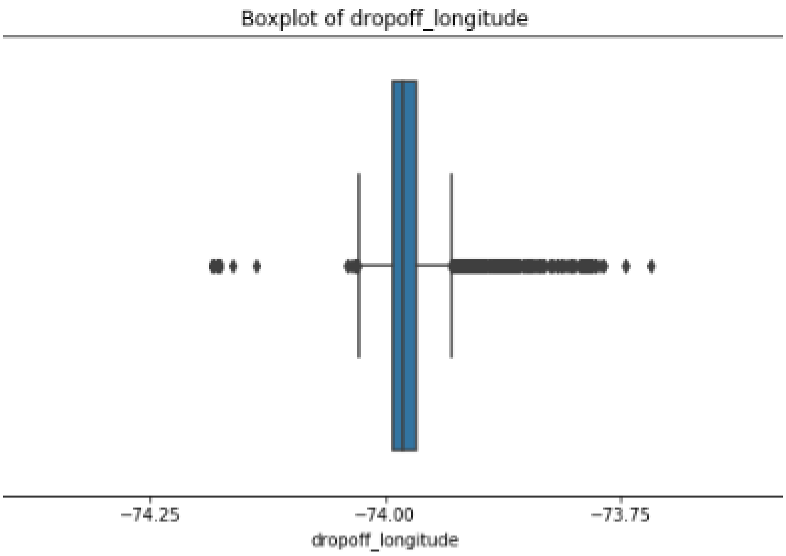
	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	fare_amount
8255	2013-05-24 00:32:00+00:00	NaN	-74.006893	40.763367	-73.961550	1.0	15.0
9869	2013-05-22 20:15:00+00:00	NaN	-74.006210	40.736887	-74.006377	6.0	52.0
9339	2013-07-13 12:31:00+00:00	NaN	-73.992947	40.802437	-73.950730	1.0	15.5
4663	2013-05-22 06:28:00+00:00	NaN	-73.991840	40.740372	-73.979010	1.0	6.5
3728	2013-06-20 04:28:00+00:00	NaN	-73.988467	40.723305	-73.939430	1.0	11.0
2084	2013-05-24 14:54:00+00:00	NaN	-73.986968	40.758867	-73.978353	2.0	5.0
6550	2013-05-22 15:33:00+00:00	NaN	-73.983285	40.757417	-73.977962	2.0	13.0
7353	2013-05-22 10:54:00+00:00	NaN	-73.973047	40.740367	-73.994392	1.0	13.0
1141	2012-10-11 00:21:00+00:00	NaN	0.033500	0.016852	0.017980	2.0	25.0

By seeing the observation, pickup_latitude <39.6 is set as outlier.

No. of observations set to NaN= 9
No. of observations remaining= 15716

6.3 dropoff_longitude

6.3.1 case 1



Based on boxplot:
min= -74.02879098749996 max= -73.92959050750004

No. of observations removed= 663
No. of observations remaining= 13859

6.3.2 case 2

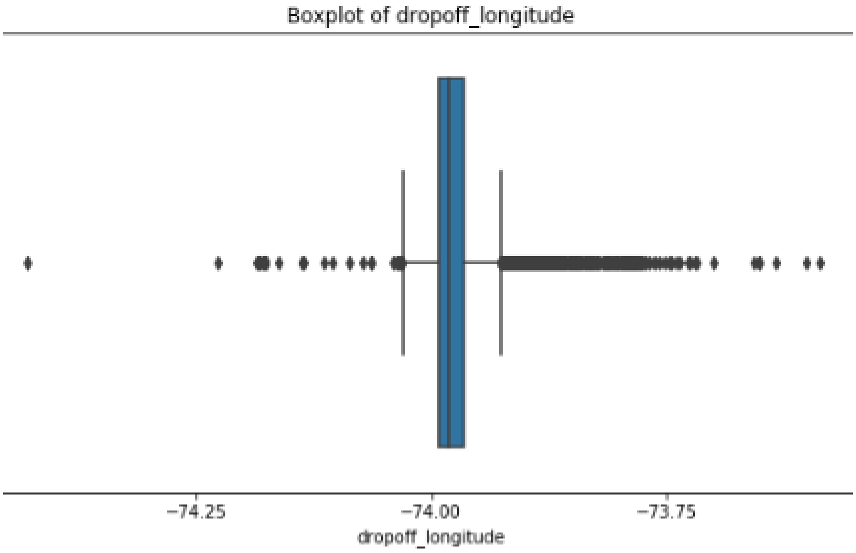
```
count    15577.000000
mean       -73.960104
std          0.991741
min        -74.429332
25%        -73.991369
50%        -73.980555
75%        -73.965385
max          0.000000
Name: dropoff_longitude, dtype: float64
```

	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	fare_amount
2154	2011-08-29 08:24:00+00:00	-73.936667	40.757815	0.00000	40.757815	1.0	8.9
15269	2012-05-12 17:58:00+00:00	-73.967183	40.772403	0.00000	40.740677	1.0	10.9
5640	2012-03-04 01:35:00+00:00	-73.995030	40.744945	-7.98664	40.729937	1.0	8.5

By seeing the observation, dropoff_longitude >-73.137 is set as outlier.

No. of observations removed= 3
No. of observations remaining= 15574

6.3.3 case 3



Based on boxplot:
Min= -74.03036654749998 max= -73.92637108750002

No. of observations set to NaN= 923
No. of observations remaining= 15716

6.3.4 case 4

```
count     15716.000000
mean        -73.897143
std           2.831607
min         -74.429332
25%         -73.991368
50%         -73.980563
75%         -73.965369
max          40.802437
Name: dropoff_longitude, dtype: float64
```

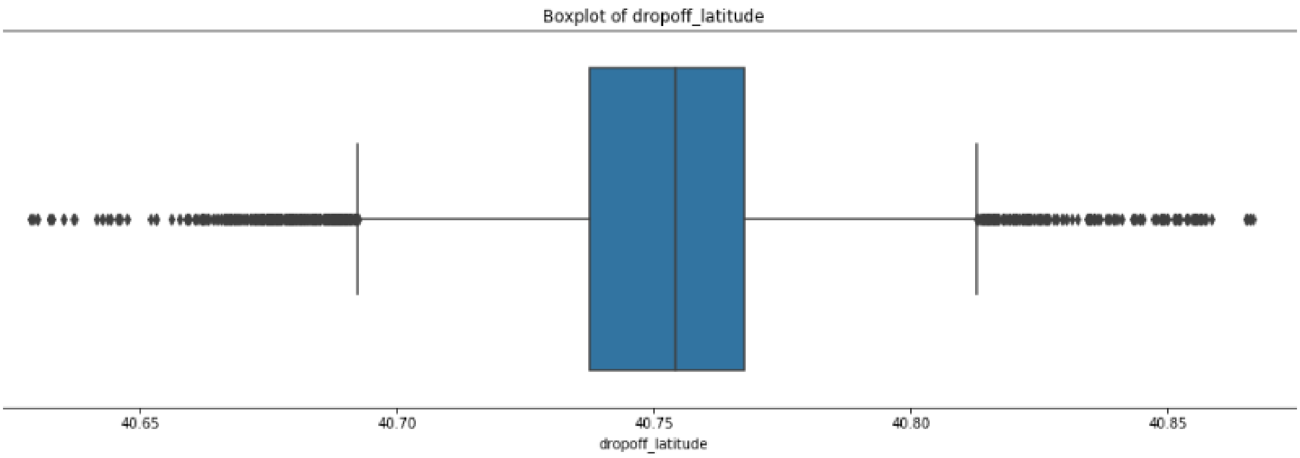
	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	fare_amount
9339	2013-07-13 12:31:00+00:00	NaN	NaN	40.802437	-73.950730	1.0	15.5
8255	2013-05-24 00:32:00+00:00	NaN	NaN	40.763367	-73.961550	1.0	15.0
2084	2013-05-24 14:54:00+00:00	NaN	NaN	40.758867	-73.978353	2.0	5.0
6550	2013-05-22 15:33:00+00:00	NaN	NaN	40.757417	-73.977962	2.0	13.0
4663	2013-05-22 06:28:00+00:00	NaN	NaN	40.740372	-73.979010	1.0	6.5
7353	2013-05-22 10:54:00+00:00	NaN	NaN	40.740367	-73.994392	1.0	13.0
9869	2013-05-22 20:15:00+00:00	NaN	NaN	40.736887	-74.006377	6.0	52.0
3728	2013-06-20 04:28:00+00:00	NaN	NaN	40.723305	-73.939430	1.0	11.0
1141	2012-10-11 00:21:00+00:00	NaN	NaN	0.016852	0.017980	2.0	25.0
2216	2011-08-29 08:24:00+00:00	-73.936667	40.757815	0.000000	40.757815	1.0	8.9
15407	2012-05-12 17:58:00+00:00	-73.967183	40.772403	0.000000	40.740677	1.0	10.9
5723	2012-03-04 01:35:00+00:00	-73.995030	40.744945	-7.986640	40.729937	1.0	8.5

By seeing the observation , dropoff_longitude >-73.137 is set as outlier.

No. of observations set to NaN= 12
No. of observations remaining= 15716

6.4 dropoff_latitude

6.4.1 case 1



Based on boxplot:
min= 40.69240849500001 max= 40.812931175

No. of observations removed= 405
No. of observations remaining= 13454

6.4.2 case 2

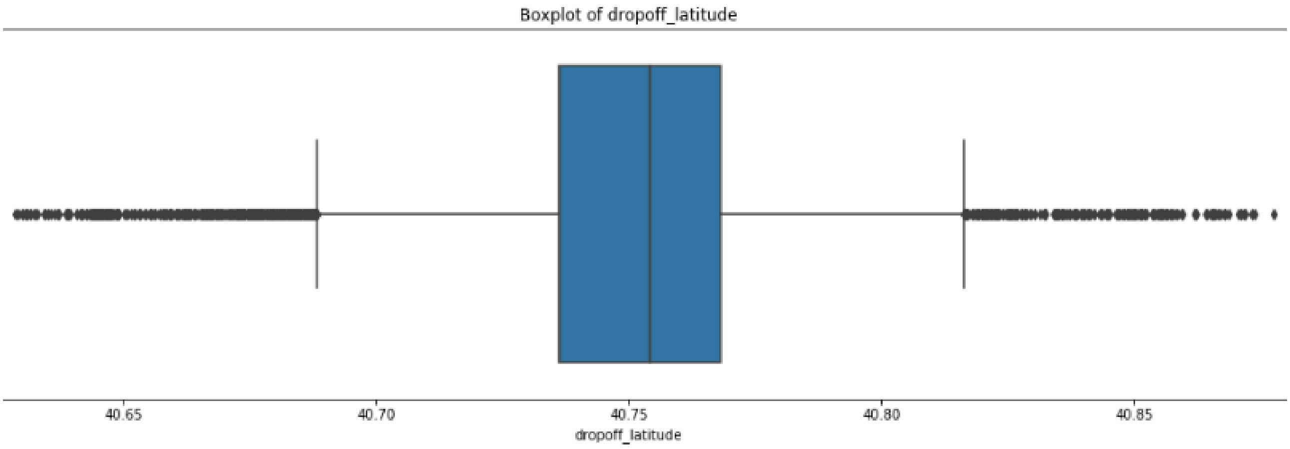
```
count    15574.000000
mean        40.748841
std          0.323157
min          0.728087
25%         40.736332
50%         40.754242
75%         40.768332
max         41.366138
Name: dropoff_latitude, dtype: float64
```

	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	fare_amount
6761	2011-06-18 04:03:00+00:00	-73.98898	40.721697	-74.001073	0.728087	3.0	4.5

By seeing the observation , dropoff_latitude <39.6 is set as outlier.

No. of observations removed= 1
No. of observations remaining= 15573

6.4.3 case 3



Based on boxplot:
min= 40.68826587499998 max= 40.81633287500002

No. of observations set to NaN= 757
No. of observations remaining= 15716

6.4.4 case 4

```
count    15716.000000
mean        40.687861
std          2.627792
min        -74.006377
25%         40.736291
50%         40.754220
75%         40.768308
max         41.366138
Name: dropoff_latitude, dtype: float64
```

	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	fare_amount
9869	2013-05-22 20:15:00+00:00	NaN	NaN	NaN	-74.006377	6.0	52.0
7353	2013-05-22 10:54:00+00:00	NaN	NaN	NaN	-73.994392	1.0	13.0
4663	2013-05-22 06:28:00+00:00	NaN	NaN	NaN	-73.979010	1.0	6.5
2084	2013-05-24 14:54:00+00:00	NaN	NaN	NaN	-73.978353	2.0	5.0
6550	2013-05-22 15:33:00+00:00	NaN	NaN	NaN	-73.977962	2.0	13.0
8255	2013-05-24 00:32:00+00:00	NaN	NaN	NaN	-73.961550	1.0	15.0
9339	2013-07-13 12:31:00+00:00	NaN	NaN	NaN	-73.950730	1.0	15.5
3728	2013-06-20 04:28:00+00:00	NaN	NaN	NaN	-73.939430	1.0	11.0
1141	2012-10-11 00:21:00+00:00	NaN	NaN	NaN	0.017980	2.0	25.0
6852	2011-06-18 04:03:00+00:00	-73.98898	40.721697	-74.001073	0.728087	3.0	4.5

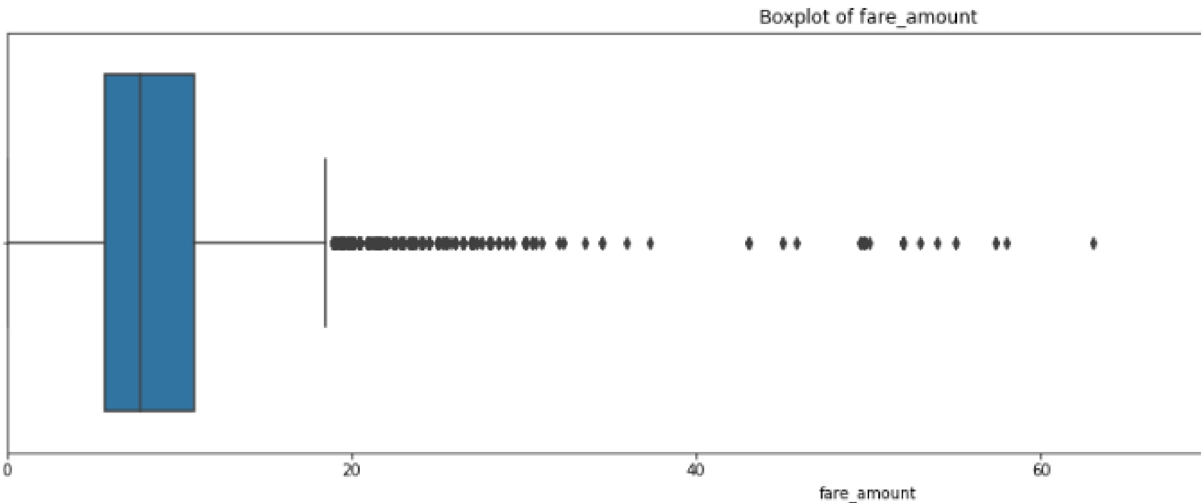
By seeing the observation , dropoff_latitude <39.6 is set as outlier.

No. of observations set to NaN= 10
No. of observations remaining= 15716

6.5 fare_amount

Since fare_amount is the target variable, we will be directly dropping the outliers instead of setting to NaN.

6.5.1 case 1



Based on boxplot:
min= -2.1000000000000005 max= 18.700000000000003

No. of observations removed= 519
No. of observations remaining= 12935

6.5.2 case 2

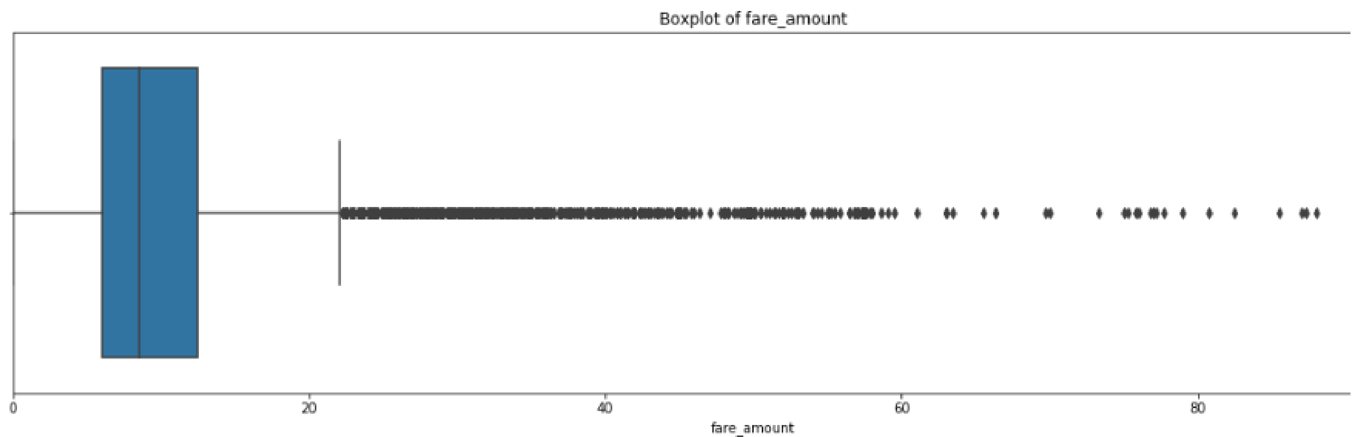
```
count    15573.000000
mean        15.159538
std        436.903326
min          0.010000
25%          6.000000
50%          8.500000
75%         12.500000
max      54343.000000
Name: fare_amount, dtype: float64
```

	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	fare_amount
935	2015-02-27 17:03:50+00:00	-74.003319	40.727455	-73.964470	40.764378	1.0	54343.0
991	2012-01-15 20:42:04+00:00	-73.976309	40.751634	-74.014854	40.709044	1.0	4343.0
555	2011-03-03 07:40:59+00:00	-74.007816	40.733536	-73.986556	40.740040	1.0	453.0
901	2011-10-24 22:54:00+00:00	-73.990602	40.761100	-73.960025	40.779580	2.0	434.0
1041	2009-09-22 19:01:01+00:00	-73.979610	40.771326	-73.975764	40.781965	1.0	430.0

By seeing the observation , fare_amount >180 is set as outlier.

No. of observations removed= 5
No. of observations remaining= 15568

6.5.3 case 3



Based on boxplot:
min= -3.75 max= 22.25

No. of observations removed= 1362
No. of observations remaining= 14354

6.5.4 case 4

```
count     15716.000000
mean         15.113220
std         434.911842
min           0.010000
25%           6.000000
50%           8.500000
75%          12.500000
max       54343.000000
Name: fare_amount, dtype: float64
```

	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	fare_amount
975	2015-02-27 17:03:50+00:00	-74.003319	40.727455	-73.964470	40.764378	1.0	54343.0
1032	2012-01-15 20:42:04+00:00	-73.976309	40.751634	-74.014854	40.709044	1.0	4343.0
583	2011-03-03 07:40:59+00:00	-74.007816	40.733536	-73.986556	40.740040	1.0	453.0
940	2011-10-24 22:54:00+00:00	-73.990602	40.761100	-73.960025	40.779580	2.0	434.0
1083	2009-09-22 19:01:01+00:00	-73.979610	40.771326	-73.975764	40.781965	1.0	430.0

By seeing the observation , fare_amount >180 is set as outlier.

No. of observations removed= 5
No. of observations remaining= 15711

6.6 Imputation of missing values of outliers

Checking whether pickup_longitude and pickup_latitude = NaN or dropoff_longitude and dropoff_latitude = NaN. If found drop them.
Then the remaining missing values of outliers are imputed by KNN.

6.6.1 case 3

No. of observations removed= 110
No. of observations remaining= 14244

6.6.2 case 4

No. of observations removed= 9
No. of observations remaining= 15702

7.Feature Extraction

7.1 Using pickup_longitude, pickup_latitude, dropoff_longitude and dropoff_latitude

From longitude and latitude coordinates given for pickup and dropoff, we can calculate a new feature called distance (vincenty distance) which is more meaningful to predict the target variable.

```
count    9914.000000
mean        3.436325
std         3.975529
min         0.000000
25%         1.299268
50%         2.218088
75%         4.045368
max       100.063275
Name: distance, dtype: float64
```

Summary of distance in test dataset

```
count    12935.000000
mean         2.246305
std          1.478465
min          0.000000
25%          1.165144
50%          1.876459
75%          2.970621
max          9.910799
Name: distance, dtype: float64
```

Summary of distance in df_1

```
count    15568.000000
mean         3.409912
std          4.590243
min          0.000000
25%          1.257965
50%          2.170056
75%          3.898851
max        129.767395
Name: distance, dtype: float64
```

Summary of distance in df_2

```
count    14244.000000
mean         2.384545
std          1.637553
min          0.000000
25%          1.193217
50%          1.945168
75%          3.164825
max         11.409653
Name: distance, dtype: float64
```

Summary of distance in df_3

```
count    15702.000000
mean         3.405270
std          4.578931
min          0.000000
25%          1.257765
50%          2.169765
75%          3.896224
max        129.767395
Name: distance, dtype: float64
```

Summary of distance in df_4

On observing the summary of distance in test dataset we can see min distance =0 , assuming that test dataset is perfect we are not removing those observations in train dataset where distance =0. And also max distance in test dataset is 100.06 which is an outlier according to boxplot(in case 1 and case 3), that's why we have made additional two cases (case 2 and case 4).

Assuming that there is no round trip, no waiting charge, no cancellation fee (if using an app), implies fare_amount should be zero for distance equals to zero.

Now every fare_amount values whose distance = 0 is set to 0 in all test cases.

7.2 Using pickup_datetime

From pickup_datetime we can extract new features like pickup_year, pickup_month, pickup_day_of_week, pickup_hour. We did encoding for pickup_year i.e. {2009:0,2010:1,2011:2,2012:3,2013:4,2014:5,2015:6}. pickup_month, pickup_day_of_week, pickup_hour is already encoded.

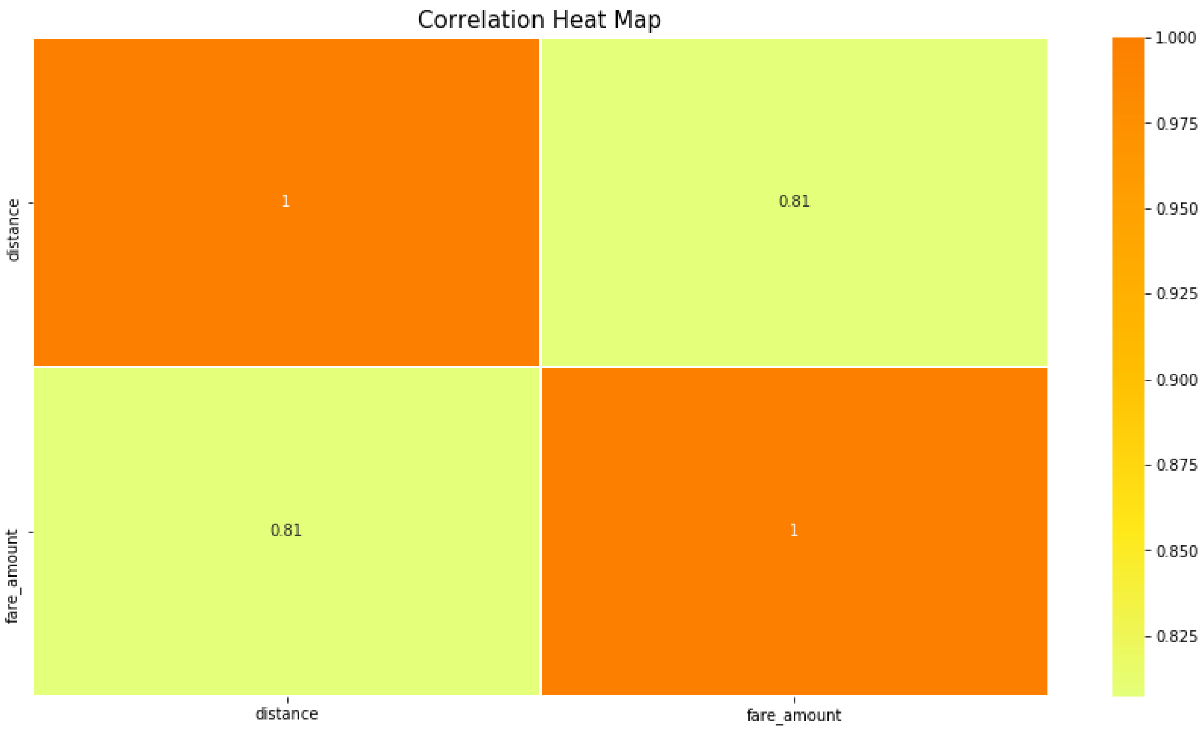
pickup_month = {January:1, ……, December:12}
pickup_day_of_week = {Monday:0, ……, Sunday:6}
pickup_hour = {12am:0, ……., 11 pm:23}

8.Feature Selection

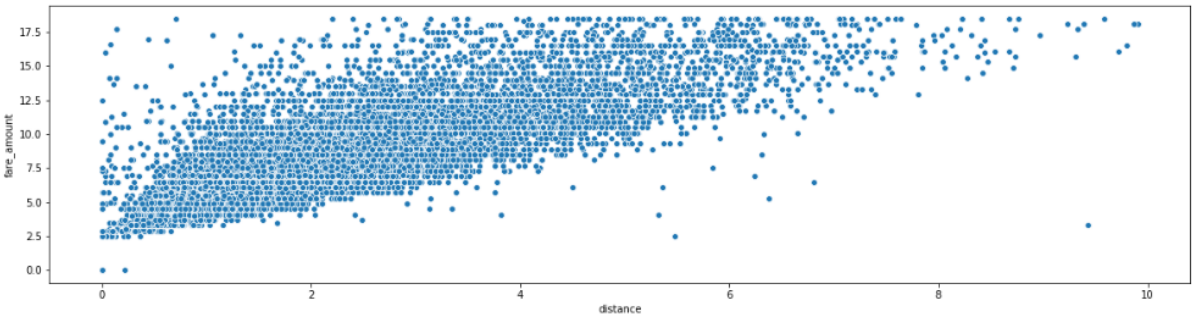
Categorical variables --> ['pickup_year', 'pickup_month', 'pickup_day_of_week', 'pickup_hour', 'passenger_count']

Continuous variables --> ['distance', 'fare_amount']

8.1 case 1



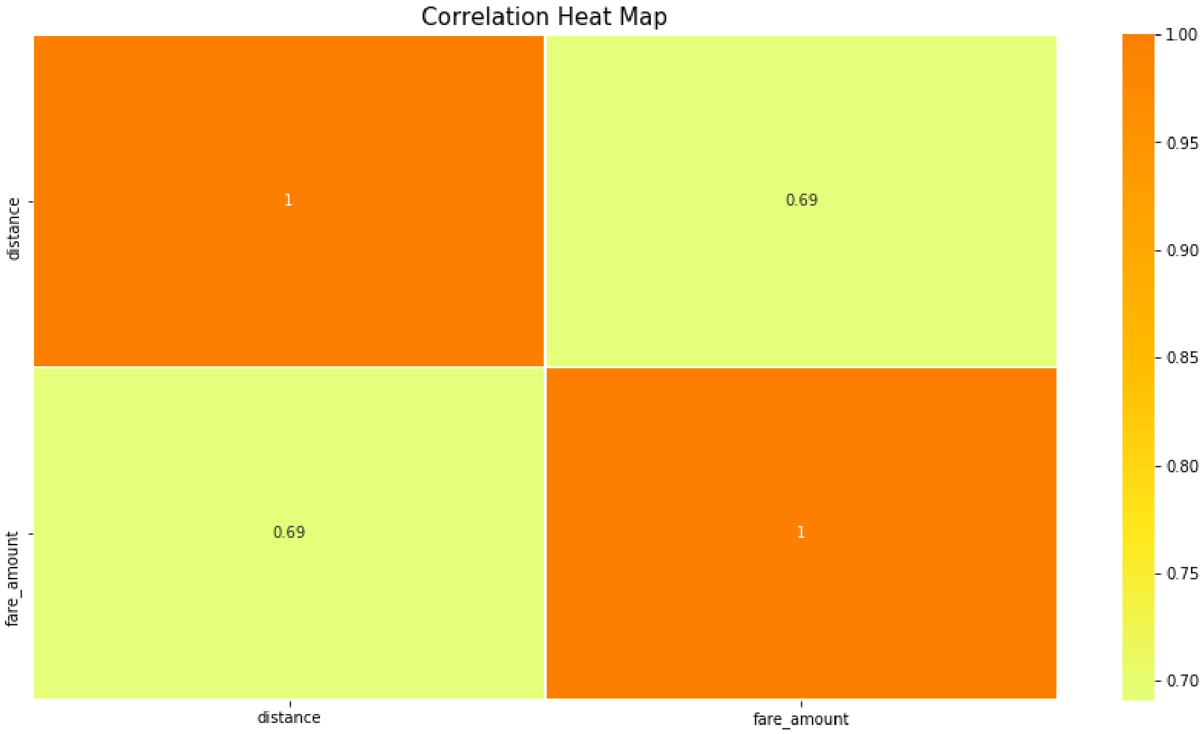
distance is highly correlated with fare_amount.



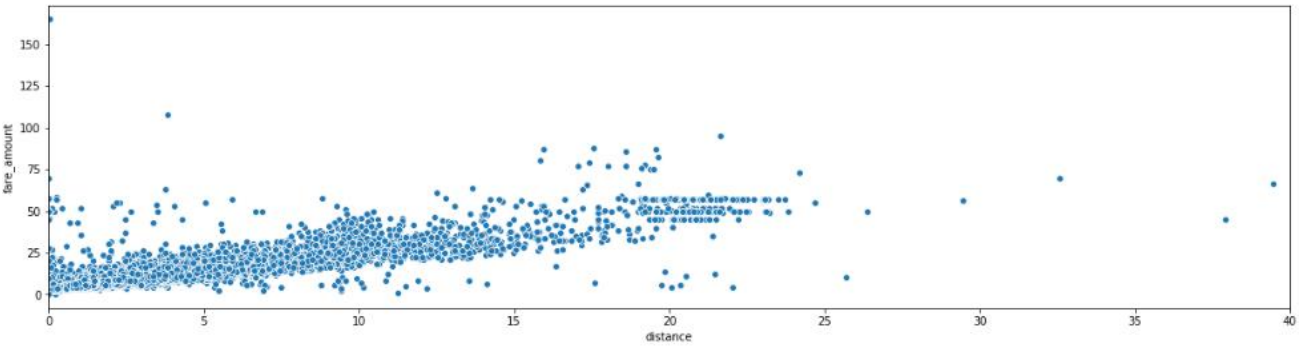
Scatter plot

There is some linear relationship between distance and fare_amount.

8.2 case 2



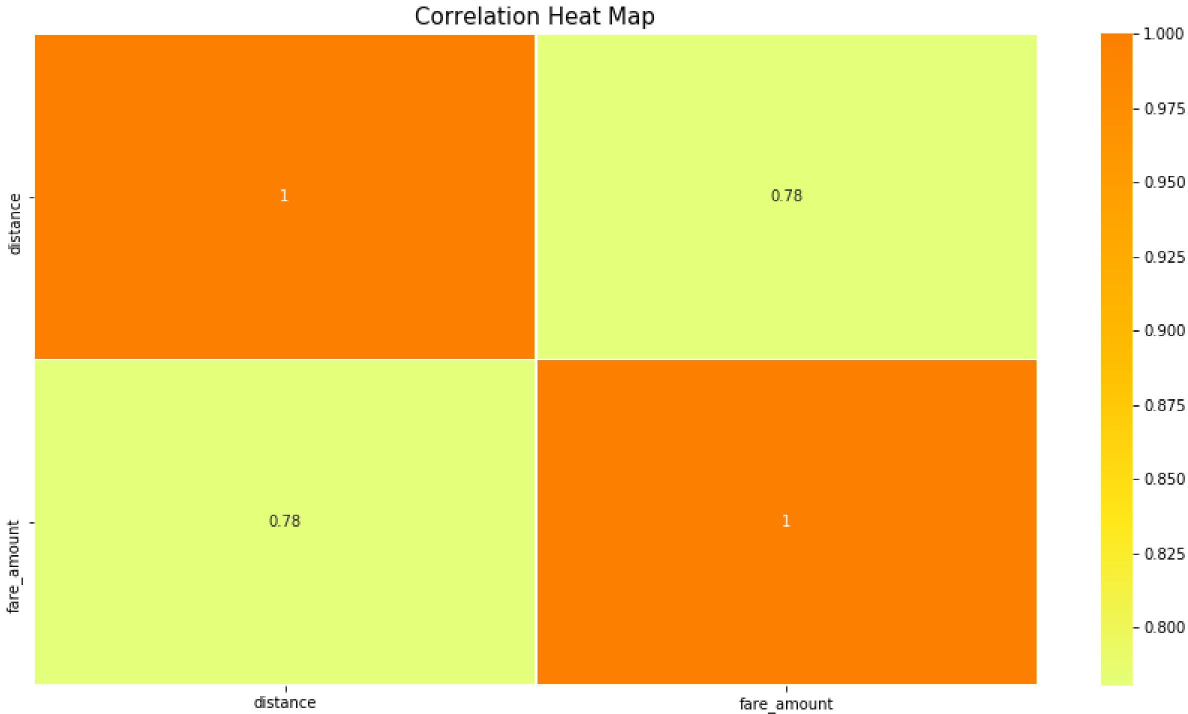
distance is moderately correlated to fare_amount.



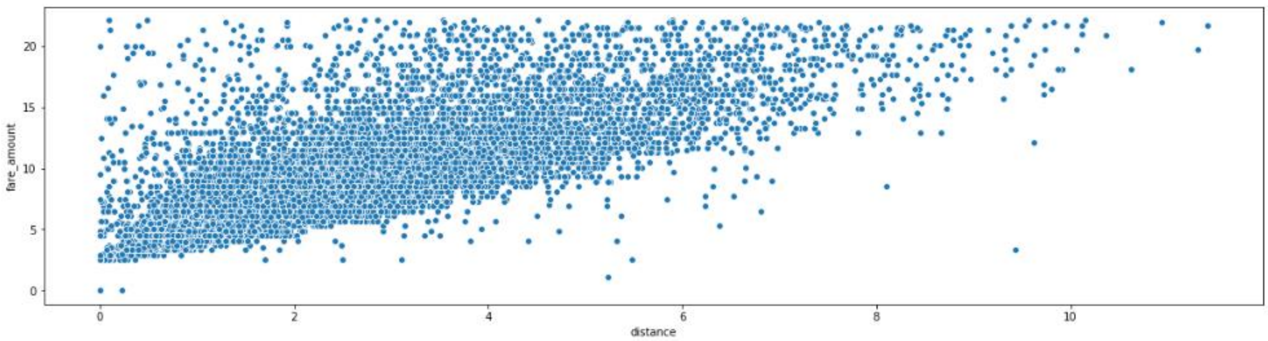
Scatter plot

There is some linear relationship between distance and fare_amount.

8.3 case 3



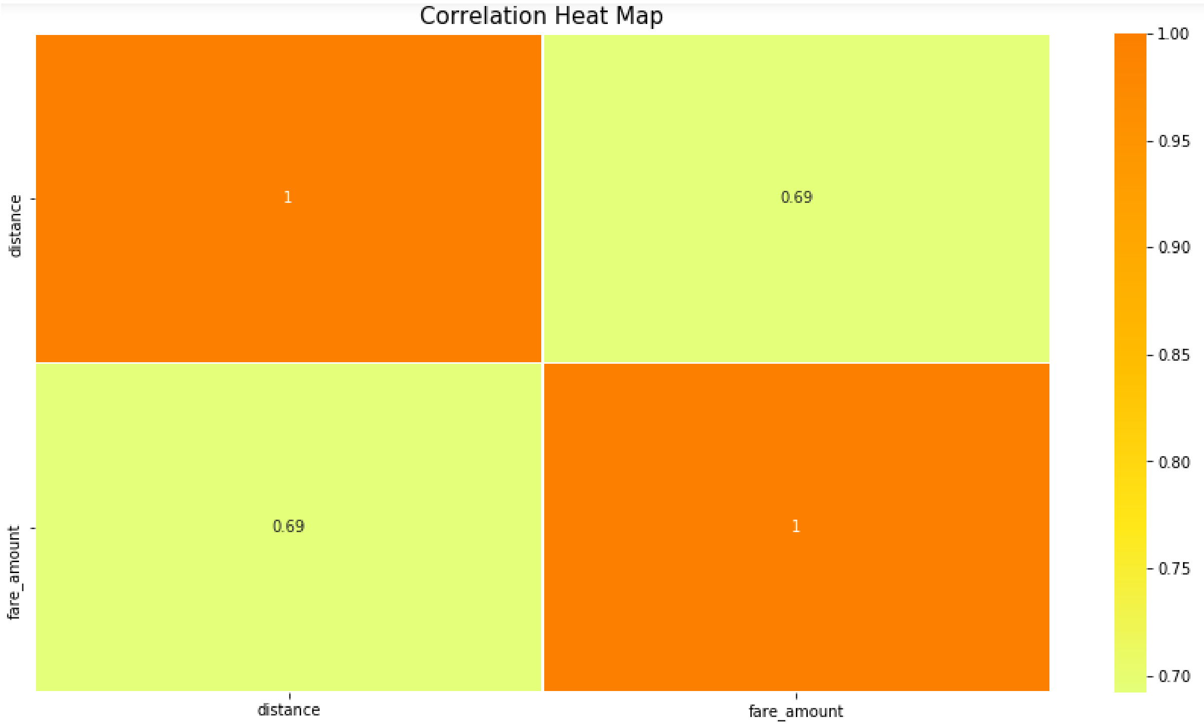
distance is highly correlated with fare_amount.



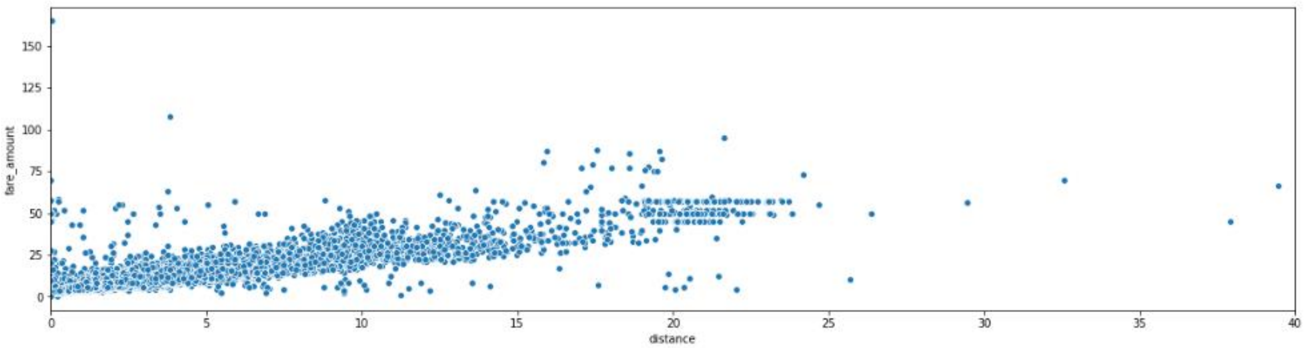
Scatter plot

There is some linear relationship between distance and fare_amount.

8.4 case 4



distance is moderately correlated to fare_amount.



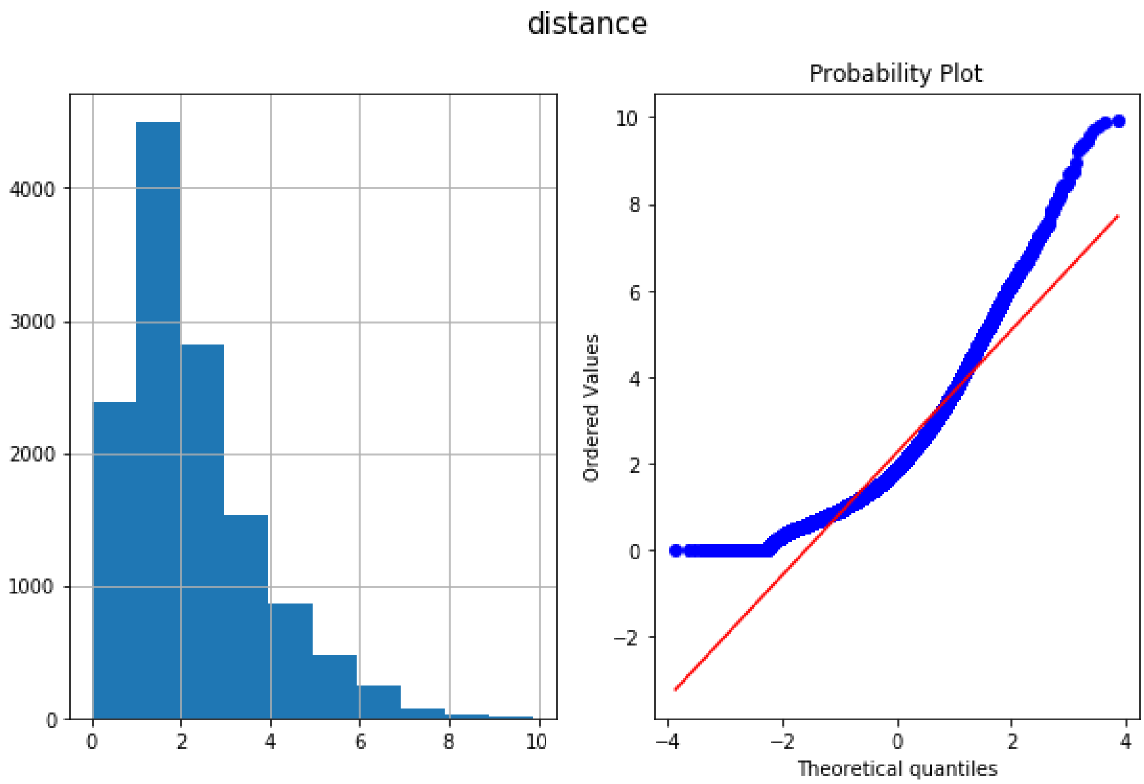
Scatter plot

There is some linear relationship between distance and fare_amount.

9.Feature Transformation

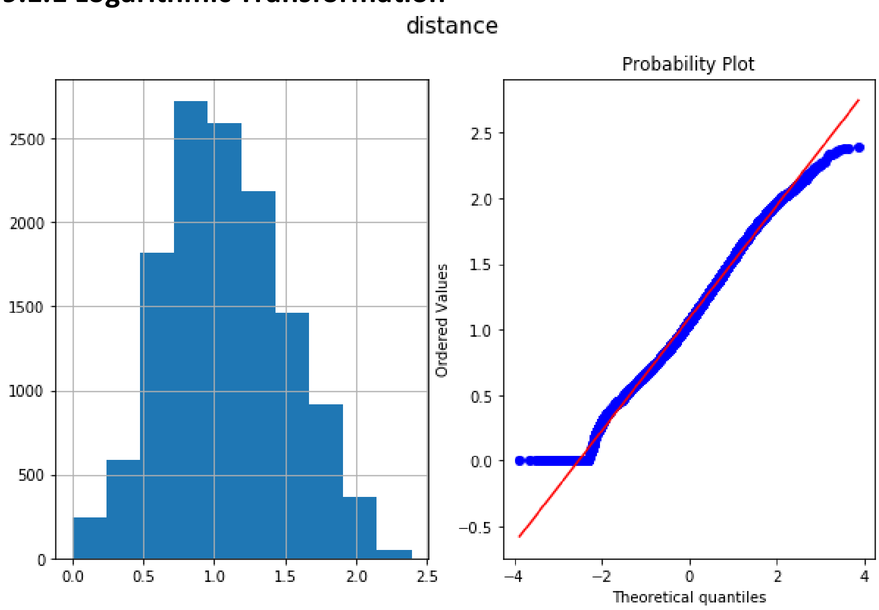
distance is our independent continuous variable. We have to check whether this variable follows normal/gaussian distribution. If not we have to transform it to normal/gaussian distribution. We will use Q-Q plot to check whether it is normally distributed or not.

9.1 case 1

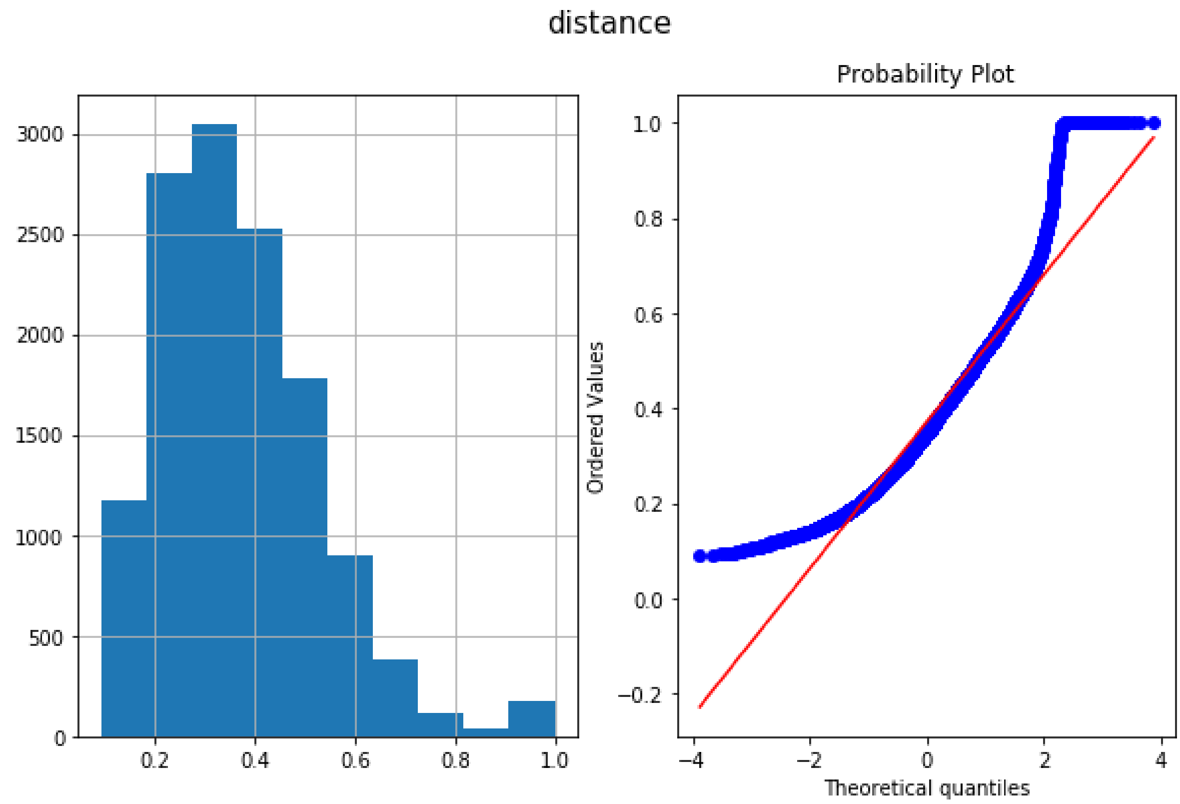


Based on Q-Q plot distance is not normally distributed.

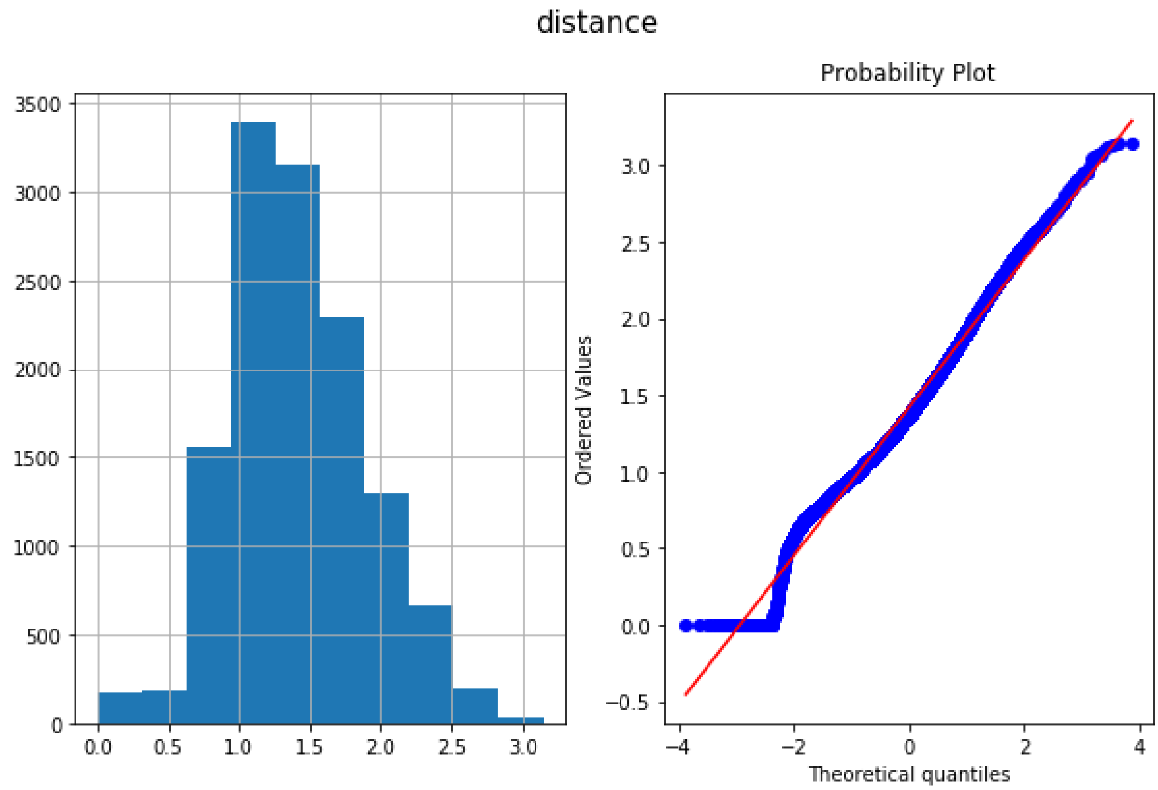
9.1.1 Logarithmic Transformation



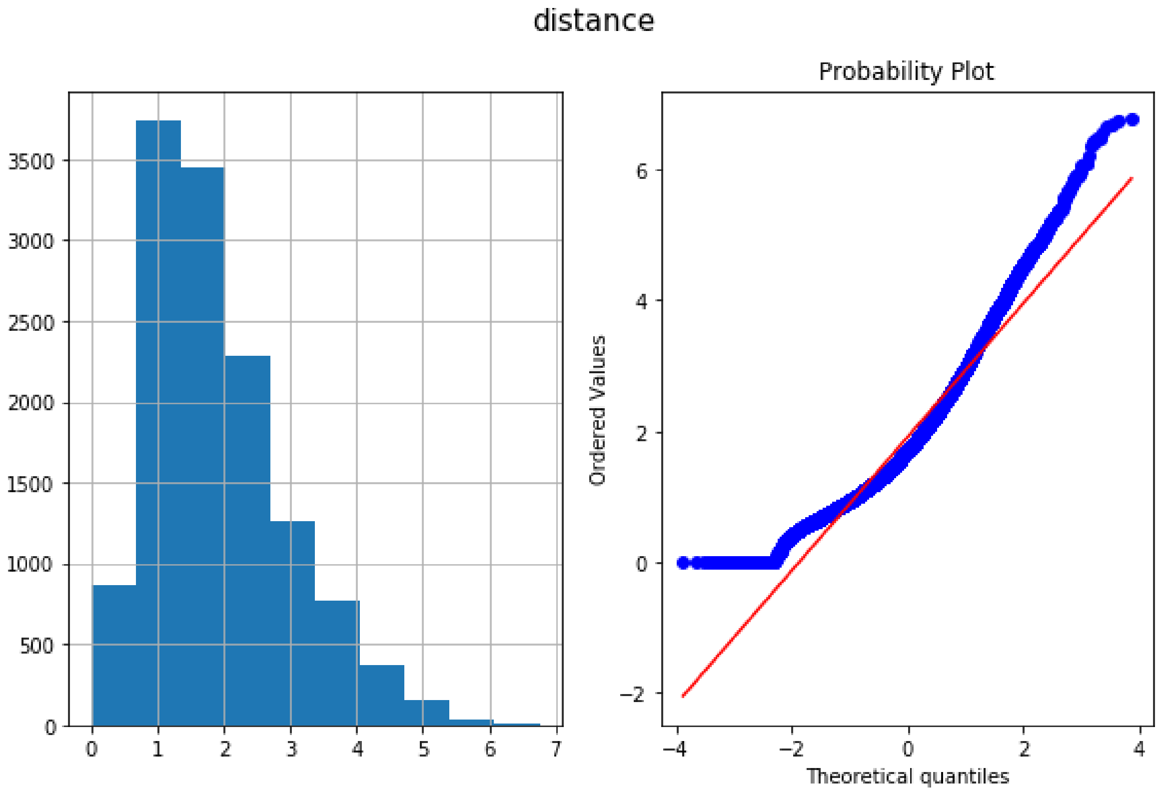
9.1.2 Reciprocal Transformation



9.1.3 Square Root Transformation

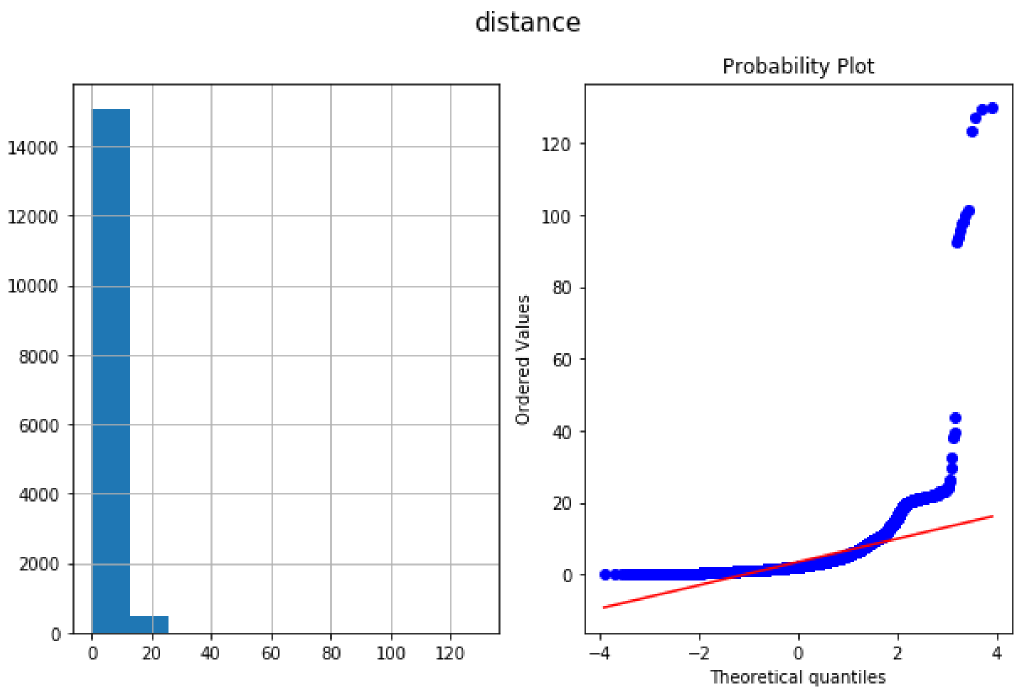


9.1.4 Exponential Transformation



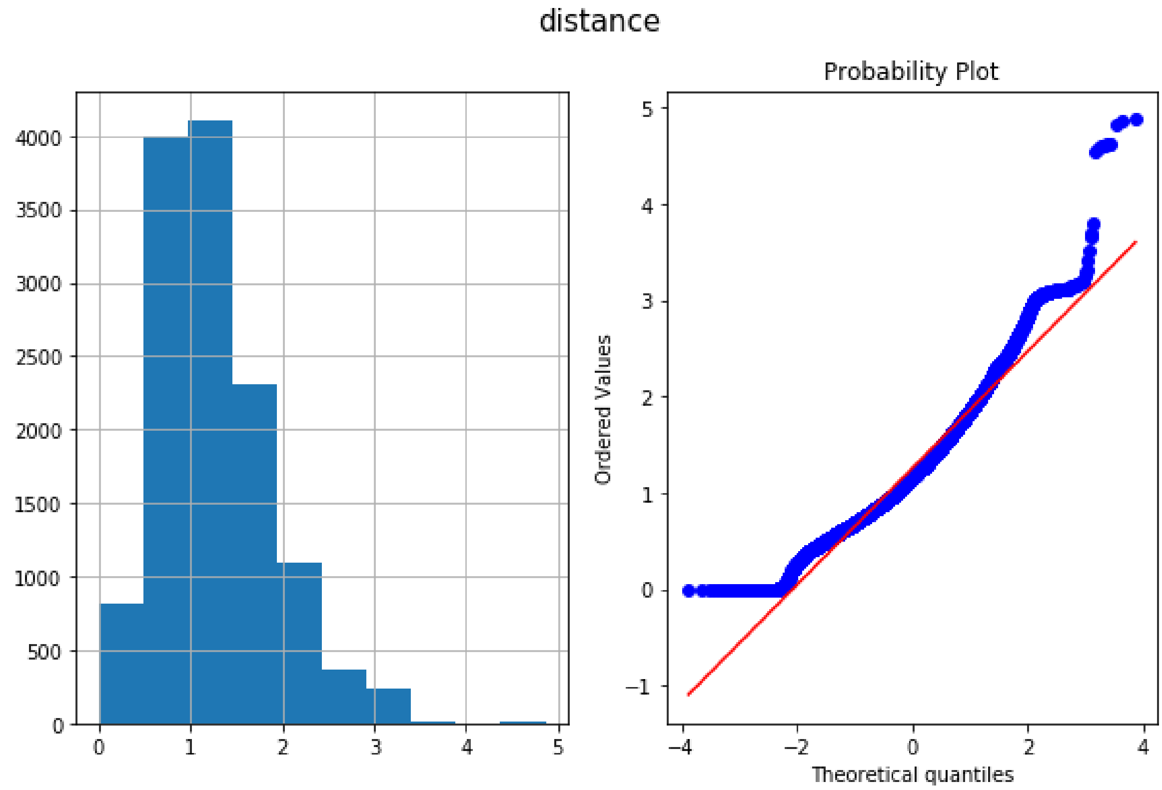
Based on Q-Q plot we select square root transformation for distance variable.

9.2 case 2

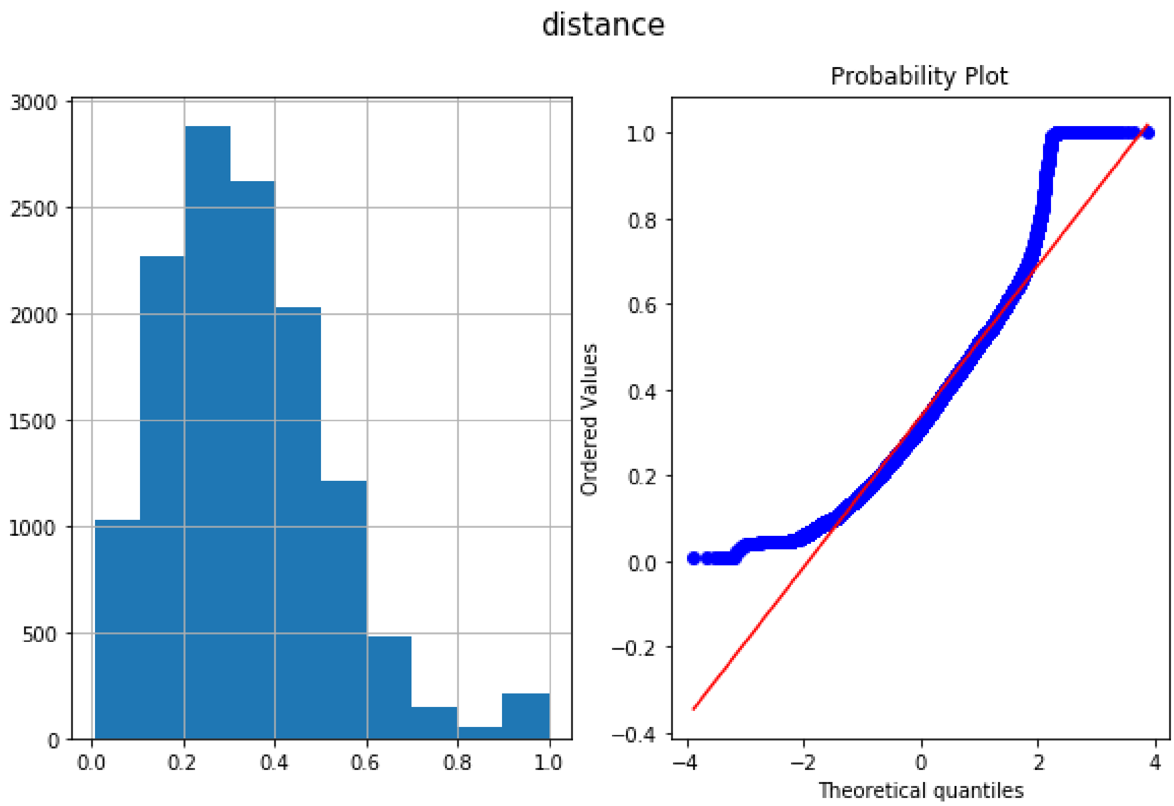


Based on Q-Q plot distance is not normally distributed.

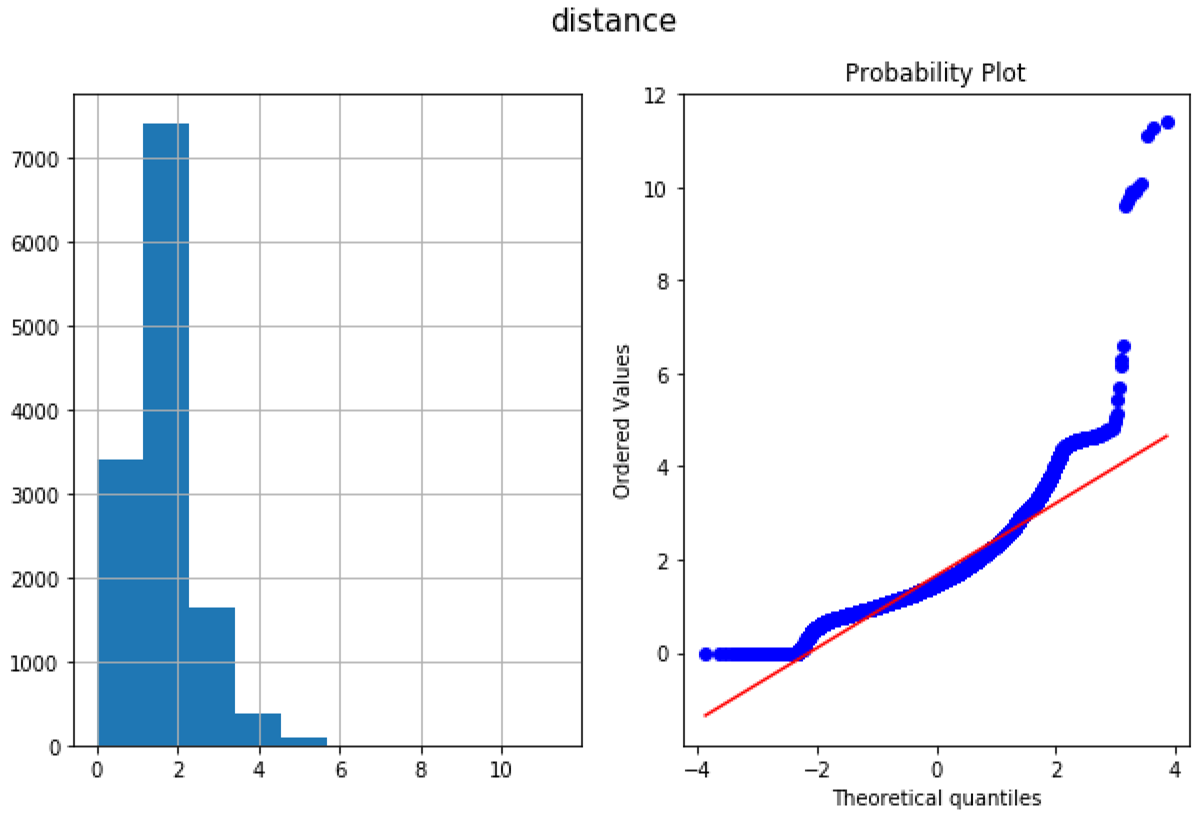
9.2.1 Logarithmic Transformation



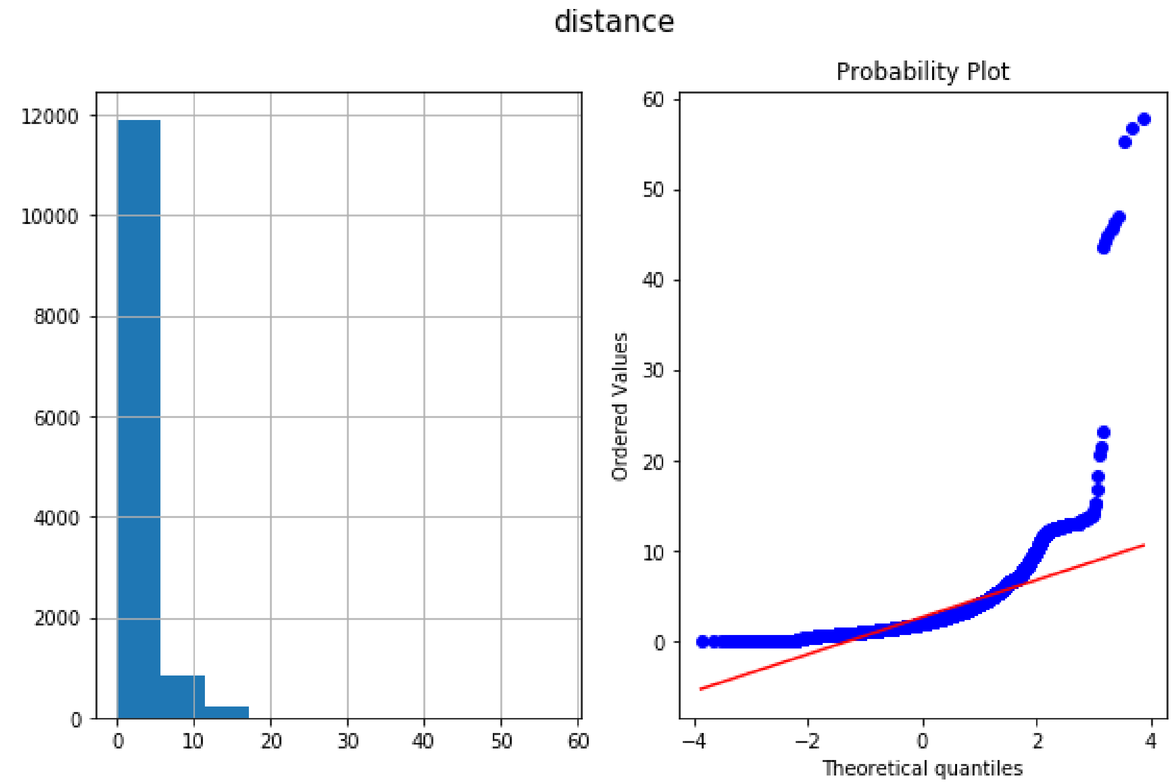
9.2.2 Reciprocal Transformation



9.2.3 Square Root Transformation

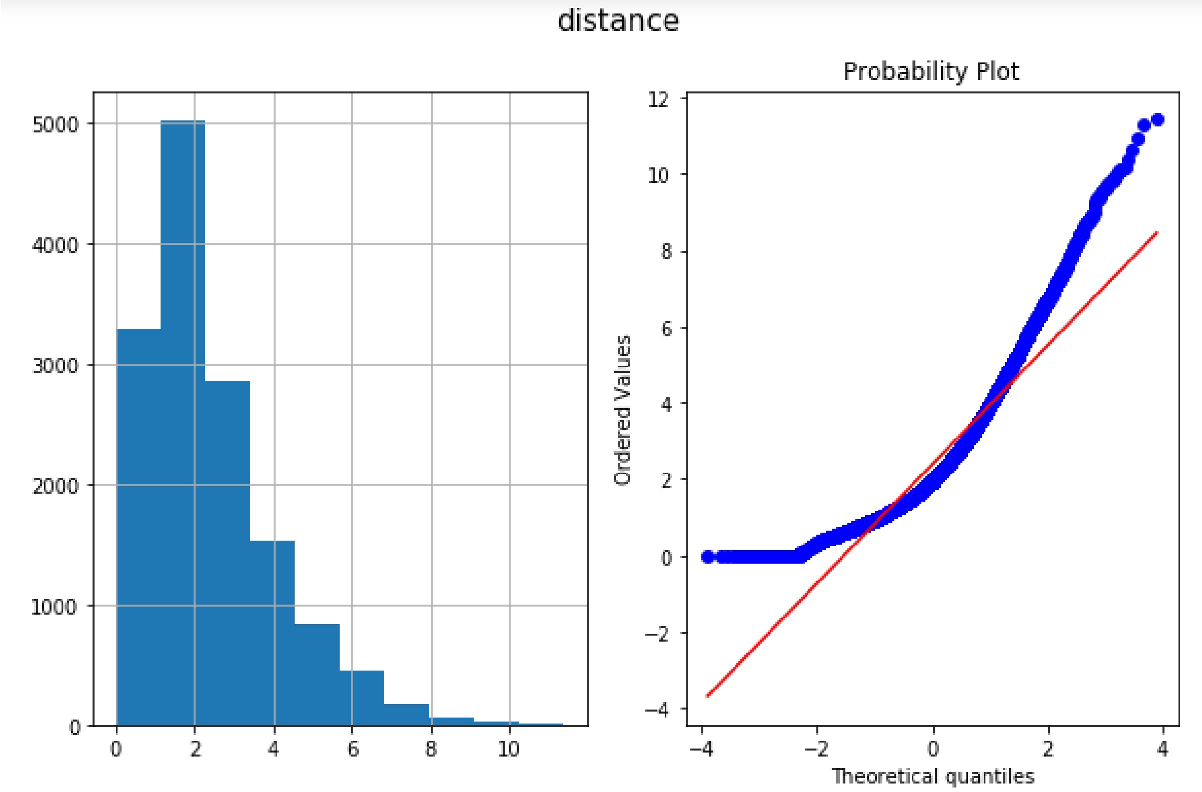


9.2.4 Exponential Transformation



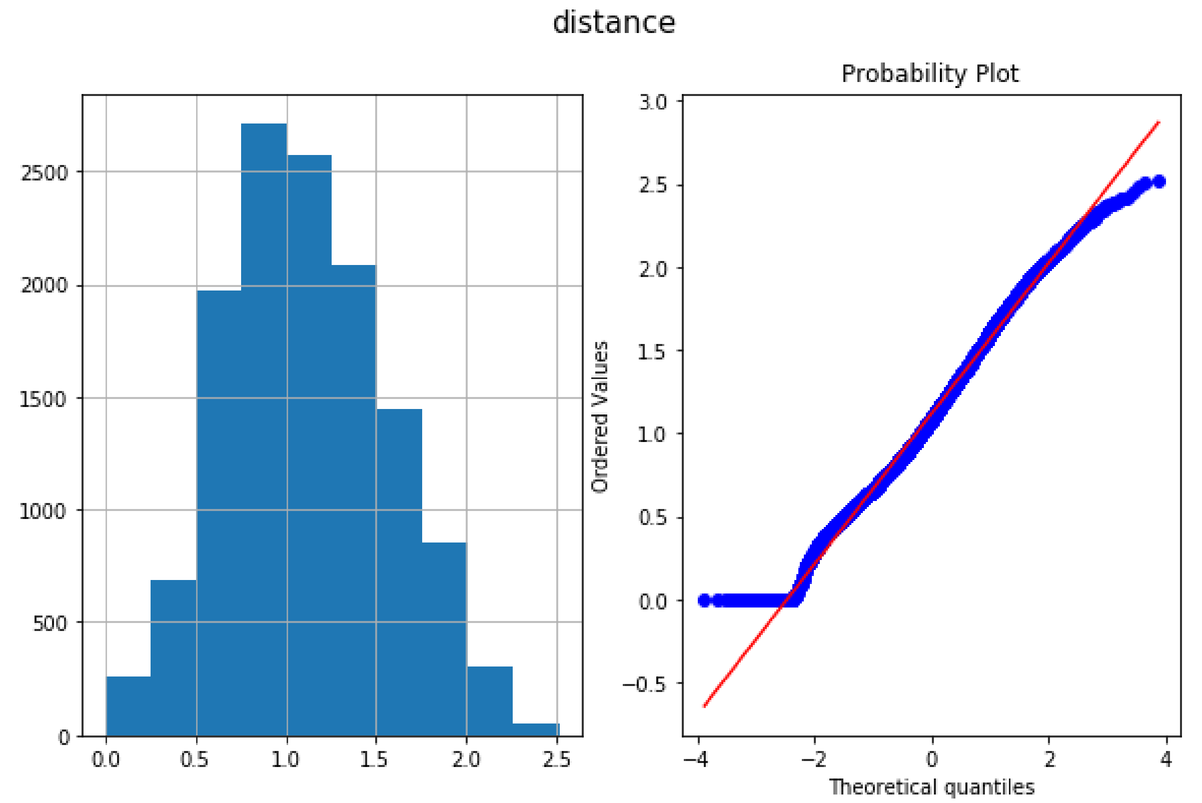
Based on Q-Q plot we select Logarithmic transformation for distance variable.

9.3 case 3

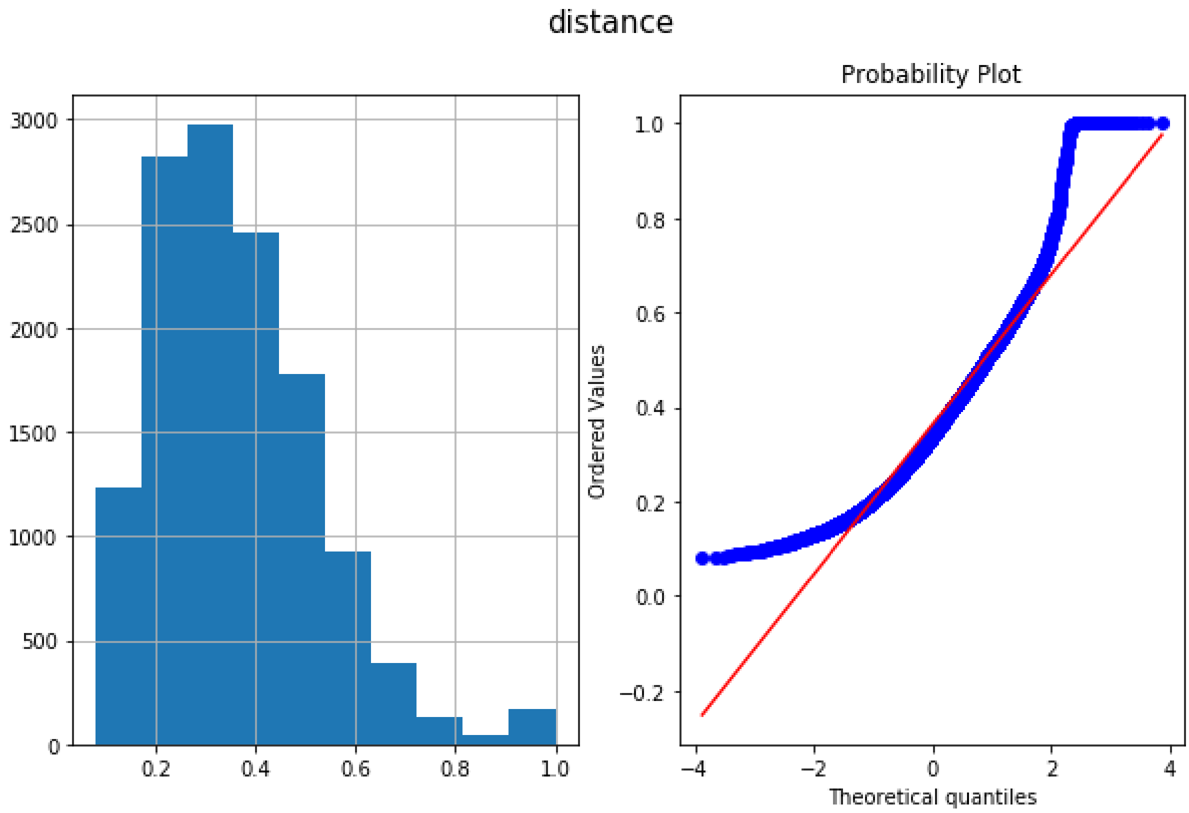


Based on Q-Q plot distance is not normally distributed.

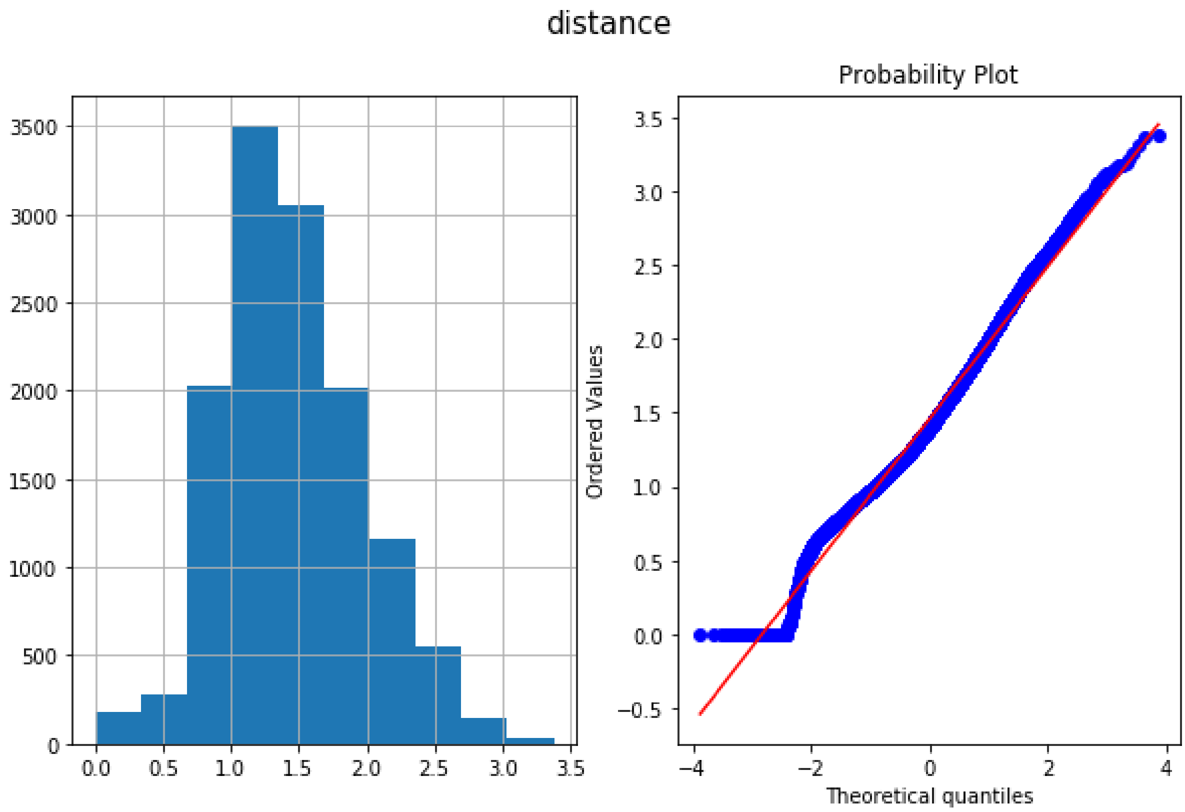
9.3.1 Logarithmic Transformation



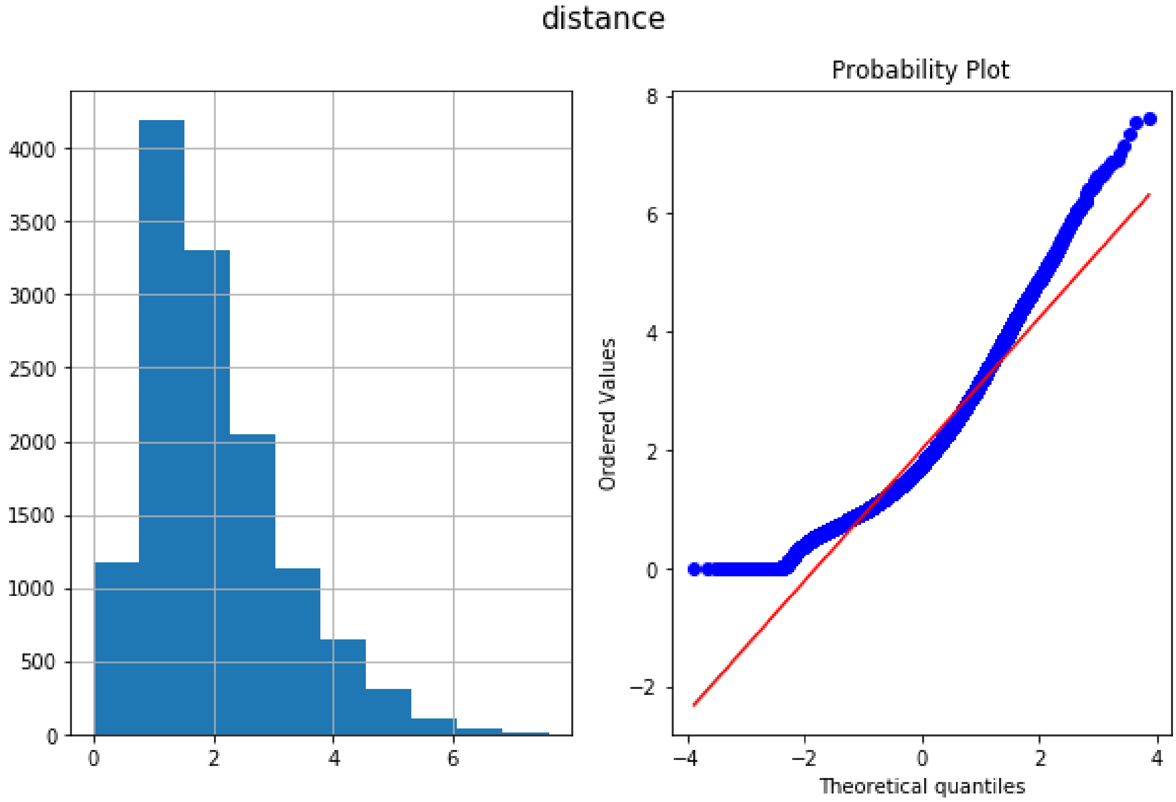
9.3.2 Reciprocal Transformation



9.3.3 Square Root Transformation

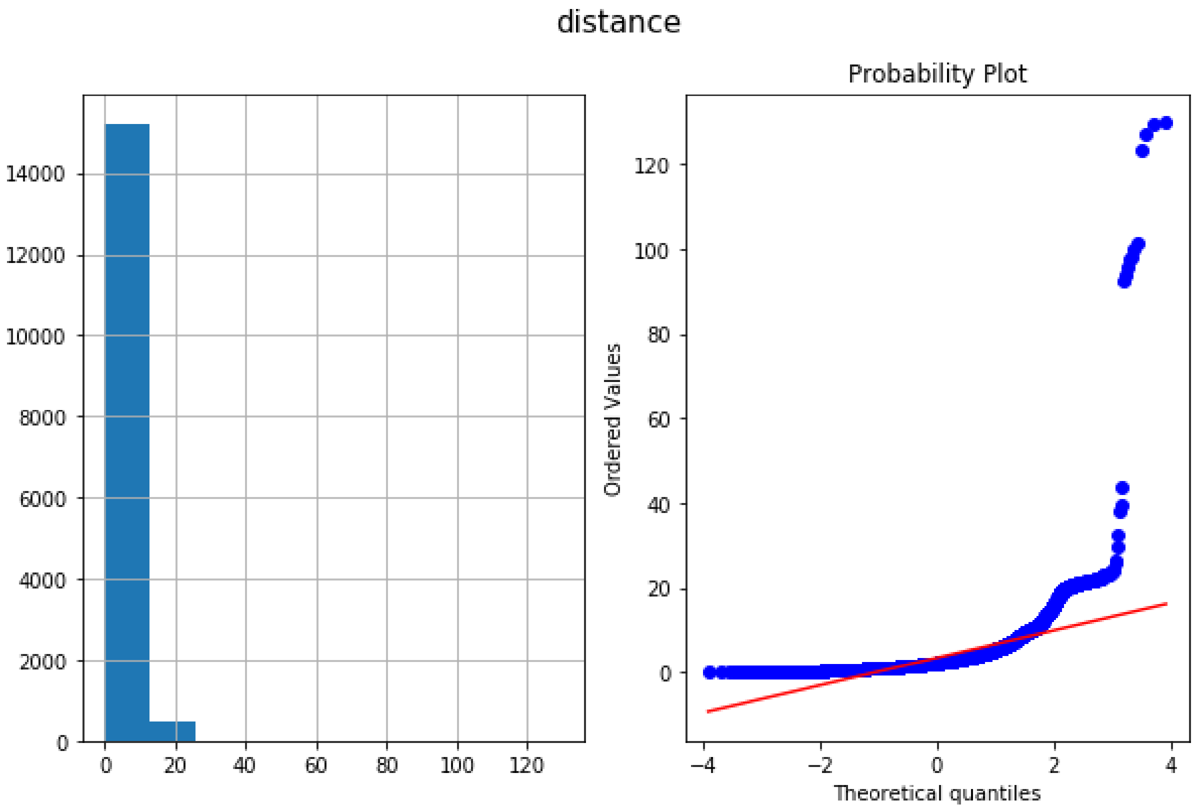


9.3.4 Exponential Transformation



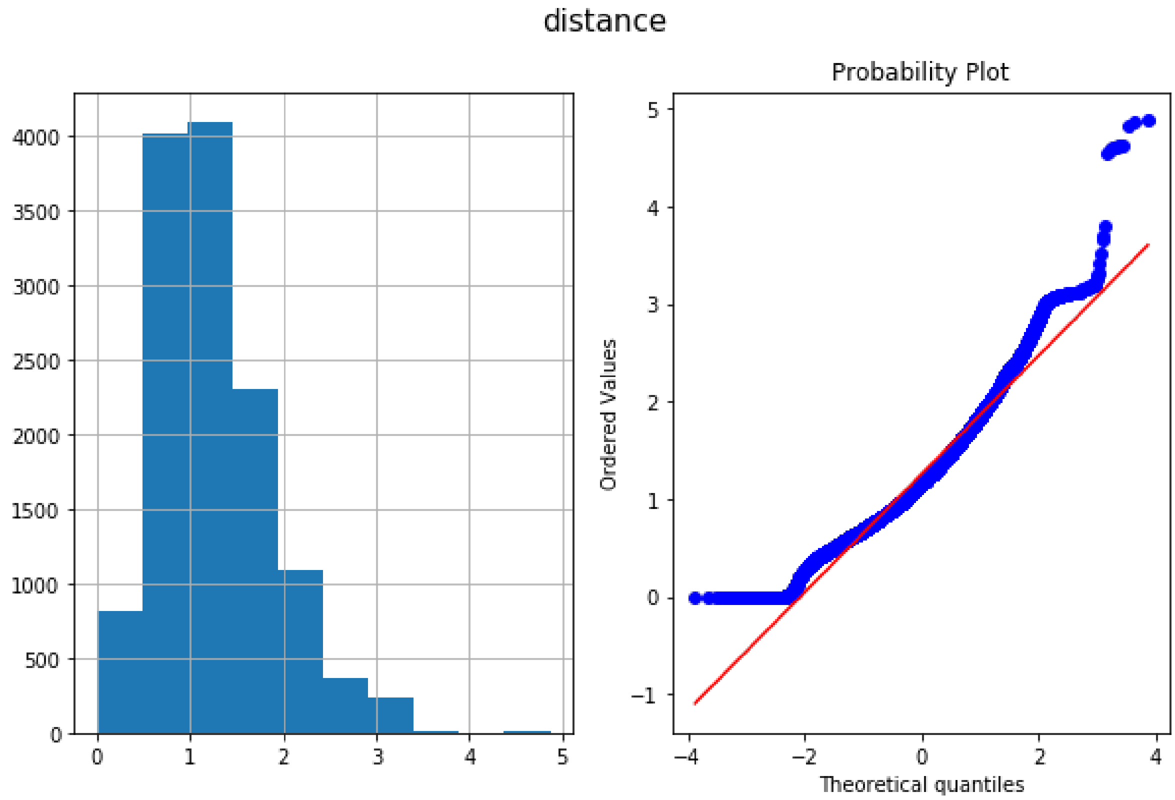
Based on Q-Q plot we select square root transformation for distance variable.

9.4 case 4

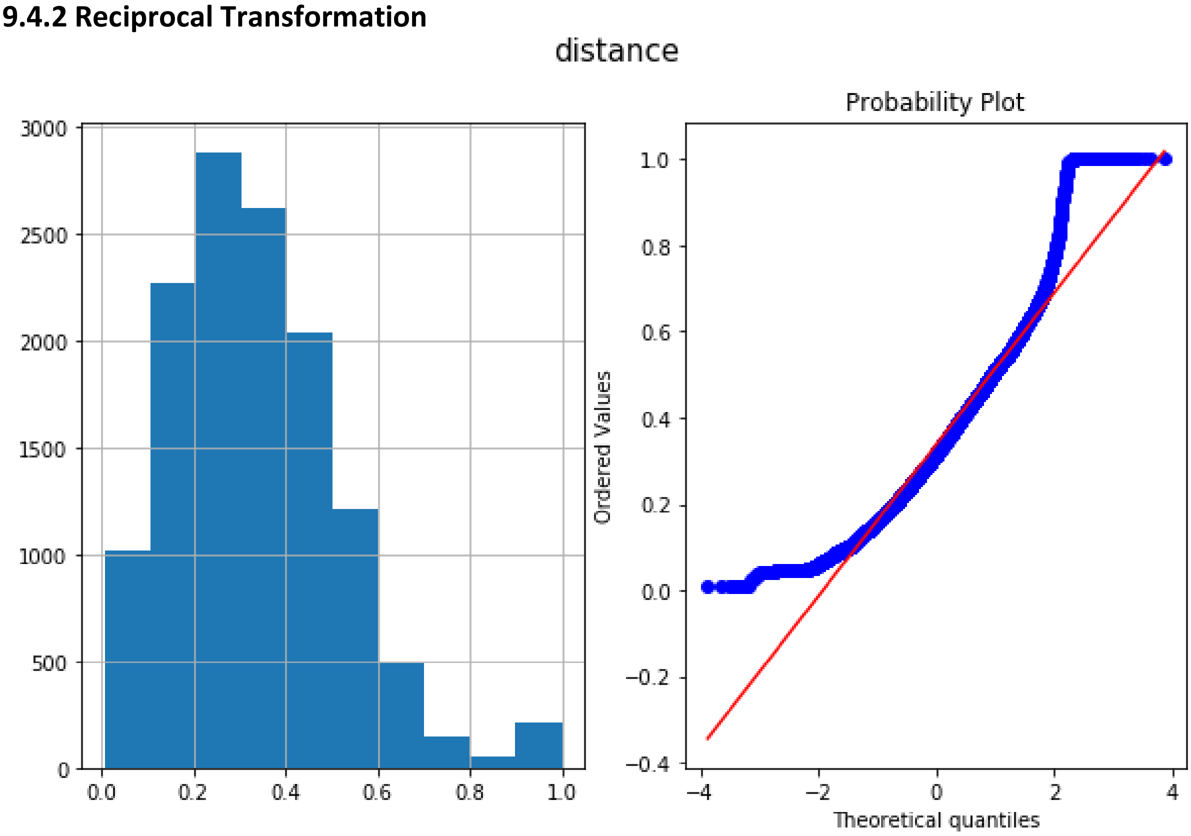


Based on Q-Q plot distance is not normally distributed.

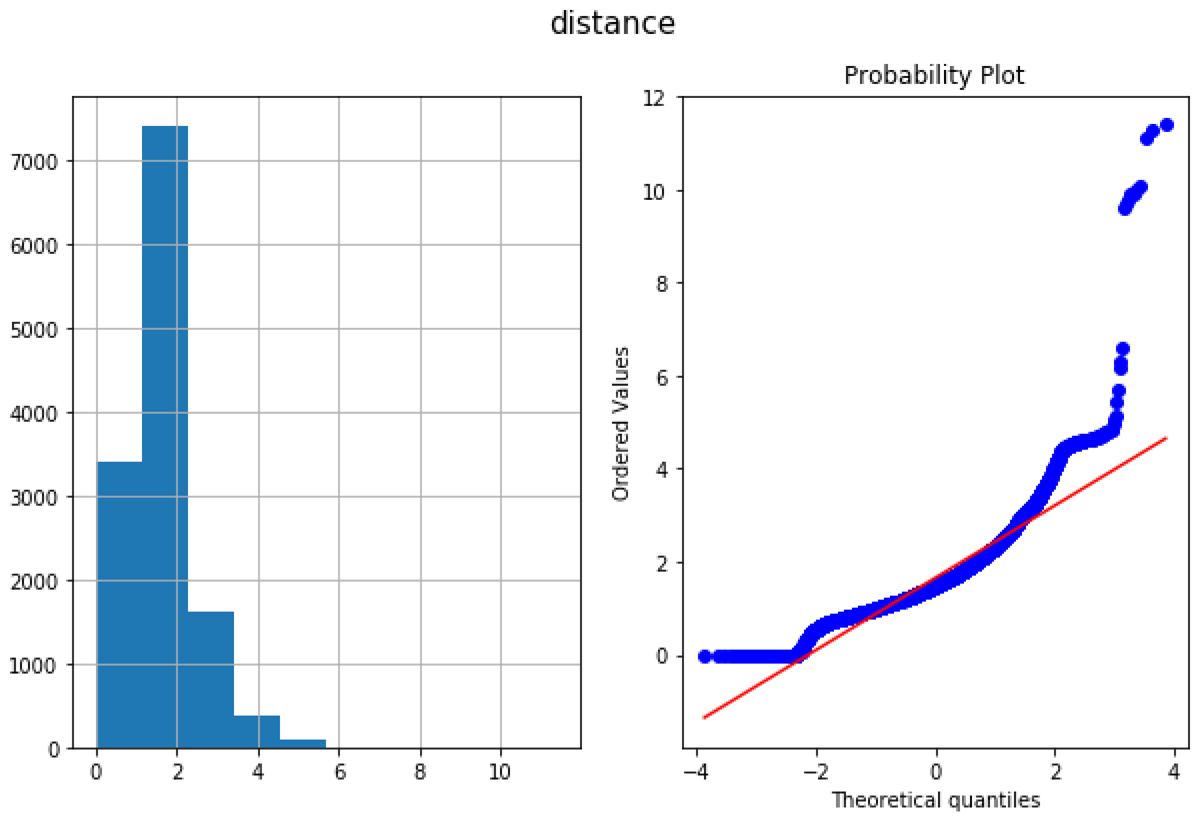
9.4.1 Logarithmic Transformation



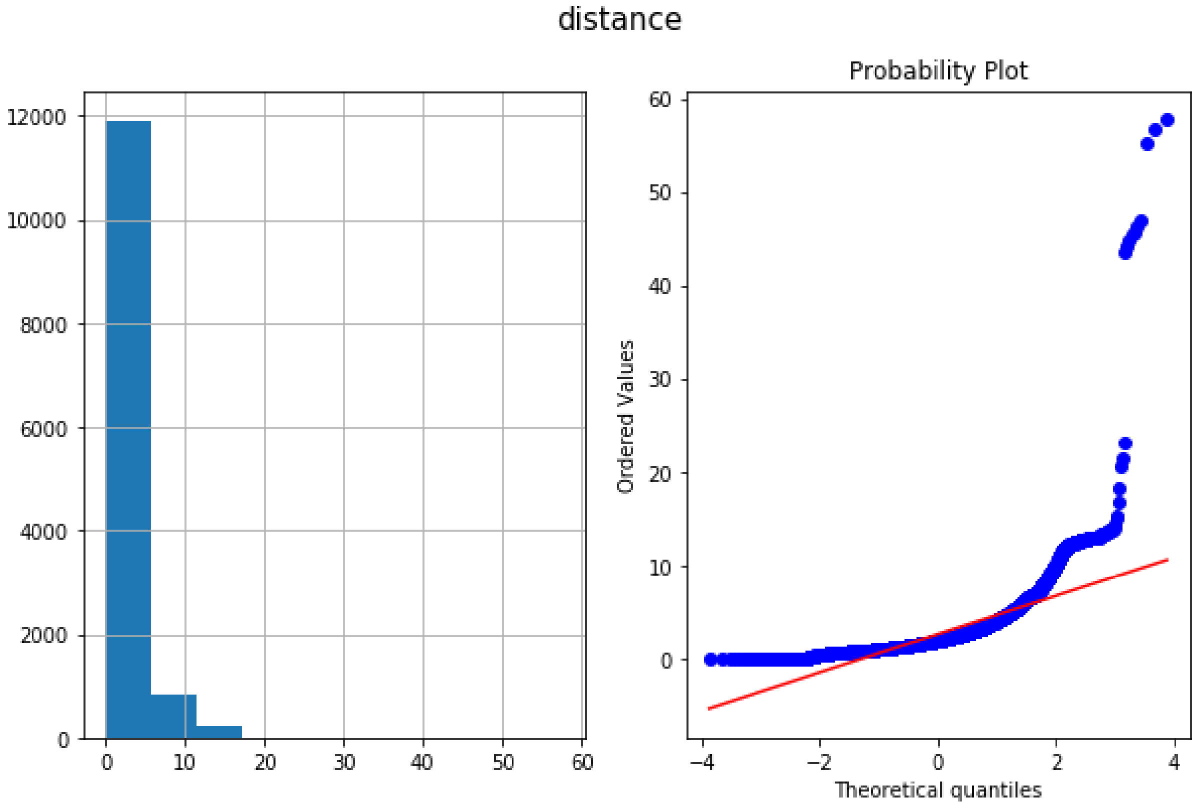
9.4.2 Reciprocal Transformation



9.4.3 Square Root Transformation



9.4.4 Exponential Transformation



Based on Q-Q plot we select Logarithmic transformation for distance variable.

10.Feature Scaling

Since we have transformed distance variable to normal distribution, we are performing standardisation on this variable.

z = (x-u)/sigma

11.Model Building

11.1 Train,Test splitting

Performed train ,test splitting from our train data, train_size= 0.7 and random_state=1234.

11.2 Models

Developed Linear Regression (LR) model, KNN Regression (KNN) model, Decision Tree Regressor (DT) model and Random Forest (RF) model each from four different cases. Hyper parameter tuning is performed for KNN, DT, RF models(only in Python). We preferred RMSE over MAPE in checking the model error is because in some cases, y_actual becomes zero and thereby division by zero cannot be calculated in case of MAPE.

11.3 Results

11.3.1 Within the test cases

11.3.1.1 Python

1. **case 1**

	case 1			
	LR_1	**KNN_1**	**DT_1**	**RF_1**
r^2 (train)	0.6965	0.9999	0.7299	0.7137
r^2 (test)	0.6811	0.568	0.6988	0.6931
adj r^2 (train)	0.6963	0.9999	0.7297	0.7135
adj r^2 (test)	0.6806	0.5673	0.6983	0.6926
RMSE	1.9716	2.2947	1.9161	1.934
Data loss %	19.49%			

Based on r^2 value, adjusted r^2 value and RMSE value, we select DT_1 as our best model. Lower values of RMSE & higher values of r^2 and adjusted r^2 are preferred.

2. **case 2**

	case 2			
	LR_2	**KNN_2**	**DT_2**	**RF_2**
r^2 (train)	0.6131	0.7301	0.7852	0.9669
r^2 (test)	0.6384	0.6344	0.8168	0.814
adj r^2 (train)	0.6128	0.7299	0.7851	0.9669
adj r^2 (test)	0.638	0.6339	0.8165	0.8138
RMSE	5.8624	5.8951	4.1729	4.2043
Data loss %	3.10%			

Based on r^2 value, adjusted r^2 value and RMSE value, we select RF_2 as our best model.

3. **case 3**

	case 3			
	LR_3	**KNN_3**	**DT_3**	**RF_3**
r^2 (train)	0.6273	0.9999	0.6544	0.9476
r^2 (test)	0.649	0.5409	0.6584	0.6445
adj r^2 (train)	0.627	0.9999	0.6542	0.9475
adj r^2 (test)	0.6485	0.5403	0.658	0.644
RMSE	2.4508	2.8027	2.4174	2.4662
Data loss %	11.34%			

Based on r^2 value, adjusted r^2 value and RMSE value, we select RF_3 as our best model.

4. **case 4**

	case 4			
	LR_4	**KNN_4**	**DT_4**	**RF_4**
r^2 (train)	0.6473	0.9999	0.8432	0.9735
r^2 (test)	0.5597	0.576	0.6995	0.718
adj r^2 (train)	0.6471	0.9999	0.8431	0.9735
adj r^2 (test)	0.5592	0.5754	0.6991	0.7177
RMSE	6.2453	6.1291	5.1592	4.9977
Data loss %	2.27%			

Based on r^2 value, adjusted r^2 value and RMSE value, we select RF_4 as our best model.

11.3.1.2 R

1. **case 1**

	case 1			
	LR_1	**KNN_1**	**DT_1**	**RF_1**
r^2 (train)	0.6981	0.6612	0.6723	0.9191
r^2 (test)	0.6775	0.5613	0.6409	0.7086
adj r^2 (train)	0.6979	0.661	0.6721	0.9191
adj r^2 (test)	0.677	0.5606	0.6403	0.7081
RMSE	1.9885	2.3193	2.0984	1.8902
Data loss %	19.49%			

Based on r^2 value, adjusted r^2 value and RMSE value, we select RF_1 as our best model.

2. **case 2**

	case 2			
	LR_2	**KNN_2**	**DT_2**	**RF_2**
r^2 (train)	0.6218	0.7265	0.751	0.938
r^2 (test)	0.6199	0.6032	0.7491	0.7797
adj r^2 (train)	0.6216	0.7263	0.7509	0.938
adj r^2 (test)	0.6194	0.6027	0.7487	0.7794
RMSE	6.0043	6.1349	4.8784	4.5705
Data loss %	3.10%			

Based on r^2 value, adjusted r^2 value and RMSE value, we select RF_2 as our best model.

3. **case 3**

	case 3			
	LR_3	**KNN_3**	**DT_3**	**RF_3**
r^2 (train)	0.6404	0.5943	0.6165	0.898
r^2 (test)	0.663	0.5309	0.6168	0.6812
adj r^2 (train)	0.6401	0.5941	0.6162	0.8979
adj r^2 (test)	0.6626	0.5303	0.6163	0.6807
RMSE	2.3974	2.8286	2.5565	2.332
Data loss %	11.34%			

Based on r^2 value, adjusted r^2 value and RMSE value, we select RF_3 as our best model.

4. **case 4**

	case 4			
	LR_4	**KNN_4**	**DT_4**	**RF_4**
r^2 (train)	0.6074	0.7357	0.7154	0.9311
r^2 (test)	0.6558	0.6271	0.7593	0.8257
adj r^2 (train)	0.6072	0.7356	0.7152	0.931
adj r^2 (test)	0.6554	0.6266	0.759	0.8255

RMSE	5.4981	5.7233	4.5978	3.9124
Data loss %	2.27%			

Based on r^2 value, adjusted r^2 value and RMSE value, we select RF_4 as our best model.

11.3.2 Between the test cases

11.3.2.1 Python

	DT_1	RF_2	RF_3	RF_4
RMSE	1.9161	4.2043	2.4662	4.9977

Since RMSE of DT_1 is lowest amongst all, we choose DT_1 model the best amongst all models.

11.3.2.2 R

	RF_1	RF_2	RF_3	RF_4
RMSE	1.9885	4.5705	2.332	3.9124

Since RMSE of RF_1 is lowest amongst all, we choose RF_1 model the best amongst all models.

12.Prediction of fare_amount

Using the best model that we found out, fare_amount in test.csv dataset is predicted in R and Python.