```
In [1]: #IMPORTING REQUIRED LIBRARY

        import os
        import pandas as pd
        import numpy as np
        import matplotlib as plt
        import datetime as dt
        import seaborn as sns
```

```
In [27]: import import_ipynb
         import matplotlib.pyplot as plt
```

```
In [250]: %matplotlib inline
          import sklearn
```

```
In [15]: from sklearn.model_selection import train_test_split
         from sklearn.tree import DecisionTreeRegressor
```

```
In [105]: from sklearn.ensemble import RandomForestRegressor
```

```
In [18]: import statsmodels.api as sm
```

```
In [19]: from sklearn.neighbors import KNeighborsClassifier
```

```
In [28]: #SETTING WORKING DIRECTORY
         os.chdir("E:/data science and machine learning/CAB project 1/Python")
```

```
In [29]: os.getcwd()

Out[29]: 'E:\\data science and machine learning\\CAB project 1\\Python'
```

```
In [35]: #GETTING THE FILE FROM HDD
         cabdf=pd.read_csv("train_cab.csv",sep=',')
```

```
In [31]: cabdf.shape

Out[31]: (16067, 7)
```

```
In [32]: type(cabdf)

Out[32]: pandas.core.frame.DataFrame
```

```
In [33]: cabdf.columns

Out[33]: Index(['fare_amount', 'pickup_datetime', 'pickup_longitude', 'pickup_latitud
         e',
                'dropoff_longitude', 'dropoff_latitude', 'passenger_count'],
               dtype='object')
```

In [36]:
```
cabdf["fare_amount"]=cabdf["fare_amount"].convert_objects(convert_numeric=True
)
cabdf['pickup_datetime']=pd.to_datetime(cabdf['pickup_datetime'],errors='coerc
e')
cabdf['pickup_year']=cabdf['pickup_datetime'].apply(lambda x:x.year)
cabdf['pickup_month']=cabdf['pickup_datetime'].apply(lambda x:x.month)
cabdf['pickup_wday']=cabdf['pickup_datetime'].dt.strftime("%u")
cabdf=cabdf.drop(columns='pickup_datetime')
```

```
C:\Users\user\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: FutureWarn
ing: convert_objects is deprecated.  To re-infer data dtypes for object colum
ns, use Series.infer_objects()
For all other conversions use the data-type specific converters pd.to_datetim
e, pd.to_timedelta and pd.to_numeric.
  """Entry point for launching an IPython kernel.
```

In [37]:
```
cabdf()
```

Out[37]:

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passe
0	4.5	-73.844311	40.721319	-73.841610	40.712278	
1	16.9	-74.016048	40.711303	-73.979268	40.782004	
2	5.7	-73.982738	40.761270	-73.991242	40.750562	
3	7.7	-73.987130	40.733143	-73.991567	40.758092	
4	5.3	-73.968095	40.768008	-73.956655	40.783762	
5	12.1	-74.000964	40.731630	-73.972892	40.758233	
6	7.5	-73.980002	40.751662	-73.973802	40.764842	
7	16.5	-73.951300	40.774138	-73.990095	40.751048	
8	NaN	-74.006462	40.726713	-73.993078	40.731628	
9	8.9	-73.980658	40.733873	-73.991540	40.758138	
10	5.3	-73.996335	40.737142	-73.980721	40.733559	
11	5.5	0.000000	0.000000	0.000000	0.000000	
12	4.1	-73.991601	40.744712	-73.983081	40.744682	
13	7.0	-74.005360	40.728867	-74.008913	40.710907	
14	7.7	-74.001821	40.737547	-73.998060	40.722788	
15	5.0	0.000000	0.000000	0.000000	0.000000	
16	12.5	-73.986430	40.760465	-73.988990	40.737075	
17	5.3	-73.981060	40.737690	-73.994177	40.728412	
18	5.3	-73.969505	40.784843	-73.958732	40.783357	
19	4.0	-73.979815	40.751902	-73.979446	40.755481	
20	10.5	-73.985382	40.747858	-73.978377	40.762070	
21	11.5	-73.957954	40.779252	-73.961250	40.758787	
22	4.5	-73.991707	40.770505	-73.985459	40.763671	
23	4.9	-74.000632	40.747473	-73.986672	40.740577	
24	6.1	-73.969622	40.756973	-73.981152	40.759712	
25	7.3	-73.991875	40.754437	-73.977230	40.774323	
26	NaN	0.000000	0.000000	0.000000	0.000000	
27	4.5	-73.988893	40.760160	-73.986445	40.757857	
28	9.3	-73.989258	40.690835	-74.004133	40.725690	
29	4.5	-73.981020	40.737760	-73.980668	40.730497	
...	...	...	...	...	...	
16037	6.5	-73.992618	40.723878	-73.977073	40.733778	
16038	5.7	-73.990336	40.718973	-73.956060	40.713974	
16039	12.9	-73.936462	40.794292	-73.948747	40.779097	
16040	6.5	-73.980597	40.744267	-73.979330	40.731205	

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passe
16041	11.0	-73.983610	40.747090	-73.961310	40.770980	
16042	8.5	-73.991425	40.749832	-74.000107	40.727898	
16043	8.5	-73.973961	40.764055	-73.986807	40.751617	
16044	16.5	-73.982785	40.731421	-74.011358	40.713788	
16045	6.5	-73.995227	40.733475	-73.984030	40.743287	
16046	6.0	-73.976298	40.753948	-73.993062	40.744550	
16047	6.1	-73.970733	40.758193	-73.979457	40.755830	
16048	9.7	-73.988040	40.774902	-74.005265	40.744157	
16049	15.7	-74.008657	40.715975	-73.975653	40.751233	
16050	8.5	-73.996715	40.742504	-73.977987	40.751805	
16051	11.5	-73.975540	40.755590	-73.944780	40.780050	
16052	10.0	-73.987298	40.722007	-74.000267	40.730342	
16053	4.0	-73.954977	40.788582	-73.964227	40.792305	
16054	5.3	-73.993929	40.756944	-73.993044	40.744088	
16055	48.3	-73.994077	40.741242	-73.830257	40.763645	
16056	38.3	0.000000	0.000000	0.000000	0.000000	
16057	5.0	-73.963582	40.774242	-73.956525	40.783952	
16058	5.5	-73.974265	40.756048	-73.980885	40.746838	
16059	5.3	-73.973297	40.743768	-73.986060	40.730768	
16060	22.0	-73.954582	40.778047	-74.005982	40.742117	
16061	10.9	-73.994191	40.751138	-73.962769	40.769719	
16062	6.5	-74.008820	40.718757	-73.998865	40.719987	
16063	16.1	-73.981310	40.781695	-74.014392	40.715527	
16064	8.5	-73.972507	40.753417	-73.979577	40.765495	
16065	8.1	-73.957027	40.765945	-73.981983	40.779560	
16066	8.5	-74.002111	40.729755	-73.983877	40.761975	

```
16067 rows × 9 columns
```

In [38]:
```
cabdf.dtypes
```

Out[38]:
```
fare_amount          float64
pickup_longitude     float64
pickup_latitude      float64
dropoff_longitude    float64
dropoff_latitude     float64
passenger_count      float64
pickup_year          float64
pickup_month         float64
pickup_wday           object
dtype: object
```

In [39]:
```
cabdf["pickup_wday"]=cabdf["pickup_wday"].convert_objects(convert_numeric=True
)
```

```
C:\Users\user\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: FutureWarn
ing: convert_objects is deprecated.  To re-infer data dtypes for object colum
ns, use Series.infer_objects()
For all other conversions use the data-type specific converters pd.to_datetim
e, pd.to_timedelta and pd.to_numeric.
  """Entry point for launching an IPython kernel.
```

In [40]:
```
#MISSING VALUE ANALYSIS
missing_val=pd.DataFrame(cabdf.isnull().sum())
```

In [41]:
```
missing_val
```

Out[41]:

	0
fare_amount	25
pickup_longitude	0
pickup_latitude	0
dropoff_longitude	0
dropoff_latitude	0
passenger_count	55
pickup_year	1
pickup_month	1
pickup_wday	1

In [42]:
```
missing_val=missing_val.reset_index()
```

In [43]:
```
missing_val=missing_val.rename(columns={'index':'Variables',0:'Missing Value'
})
```

In [44]:
```
missing_val
```

Out[44]:

	Variables	Missing Value
0	fare_amount	25
1	pickup_longitude	0
2	pickup_latitude	0
3	dropoff_longitude	0
4	dropoff_latitude	0
5	passenger_count	55
6	pickup_year	1
7	pickup_month	1
8	pickup_wday	1

In [45]:
```
#MEAN METHOD for fare amount
cabdf['fare_amount']=cabdf['fare_amount'].fillna(cabdf['fare_amount'].mean())
```

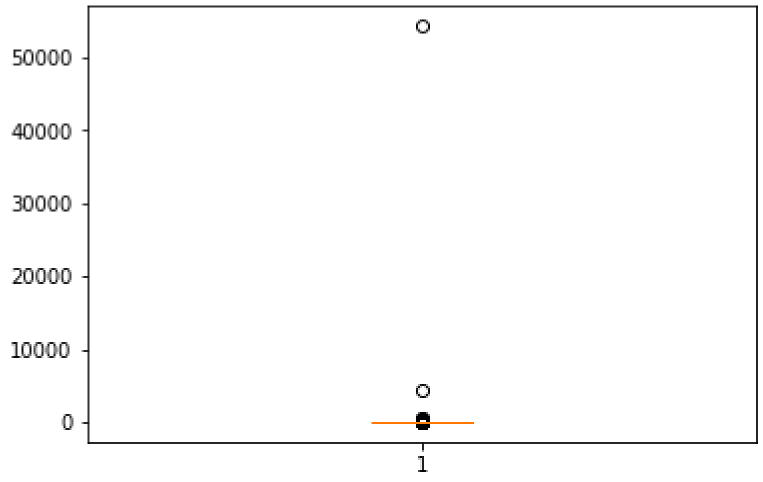
In [46]:
```
#MEDIAN METHOD for passenger count, pickup year,month & wday
cabdf['passenger_count']=cabdf['passenger_count'].fillna(cabdf['passenger_coun
t'].median())

cabdf['pickup_year']=cabdf['pickup_year'].fillna(cabdf['pickup_year'].median
())
cabdf['pickup_month']=cabdf['pickup_month'].fillna(cabdf['pickup_month'].media
n())
cabdf['pickup_wday']=cabdf['pickup_wday'].fillna(cabdf['pickup_wday'].median
())
```

In [47]:
```
#MISSING VALUE
missing_val=pd.DataFrame(cabdf.isnull().sum())
missing_val
```

Out[47]:

	0
fare_amount	0
pickup_longitude	0
pickup_latitude	0
dropoff_longitude	0
dropoff_latitude	0
passenger_count	0
pickup_year	0
pickup_month	0
pickup_wday	0

In [48]:
```
#BOX PLOT TO CHECK OUTLIERS OF EVERY VARIABLE
plt.boxplot(cabdf['fare_amount'])
cnames=['fare_amount', 'pickup_longitude', 'pickup_latitude',
       'dropoff_longitude', 'dropoff_latitude', 'passenger_count','pickup_yea
r','pickup_month','pickup_wday']
```



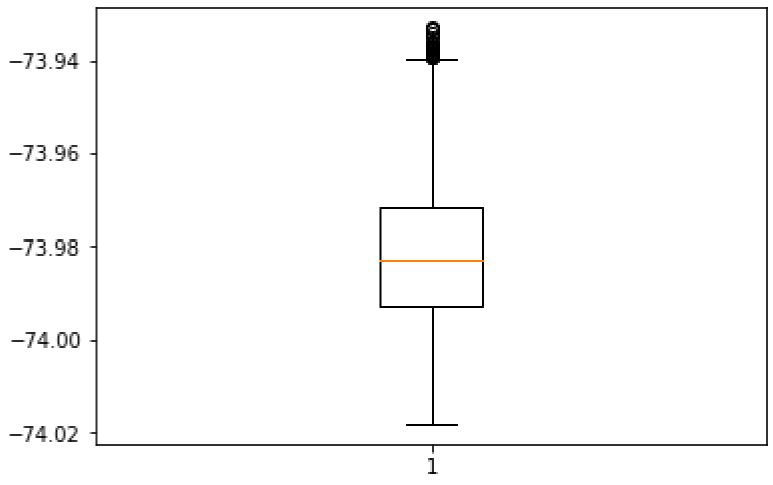
In [49]:
```
for i in cnames:
    q75,q25=np.percentile(cabdf.loc[:,i],[75,25])
    iqr=q75-q25
    min=q25-(iqr*1.5)
    max=q75+(iqr*1.5)

    cabdf=cabdf.drop(cabdf[cabdf.loc[:,i]<min].index)
    cabdf=cabdf.drop(cabdf[cabdf.loc[:,i]>max].index)
```

In [50]:
```
cabdf.shape
```

Out[50]:
```
(11878, 9)
```

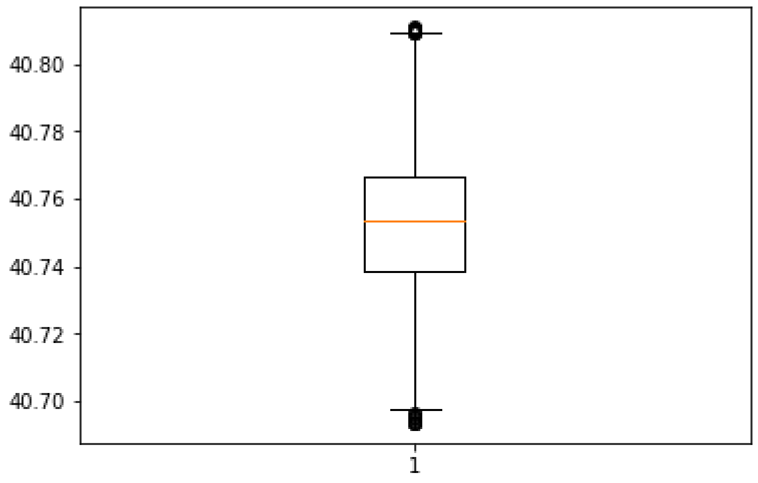
```
In [52]: plt.boxplot(cabdf['pickup_longitude'])
```

```
Out[52]: {'whiskers': [<matplotlib.lines.Line2D at 0x187da0deac8>,
           <matplotlib.lines.Line2D at 0x187da0dee10>],
          'caps': [<matplotlib.lines.Line2D at 0x187da0deef0>,
           <matplotlib.lines.Line2D at 0x187da0d24e0>],
          'boxes': [<matplotlib.lines.Line2D at 0x187da0de710>],
          'medians': [<matplotlib.lines.Line2D at 0x187da0d2828>],
          'fliers': [<matplotlib.lines.Line2D at 0x187da0d2b70>],
          'means': []}
```



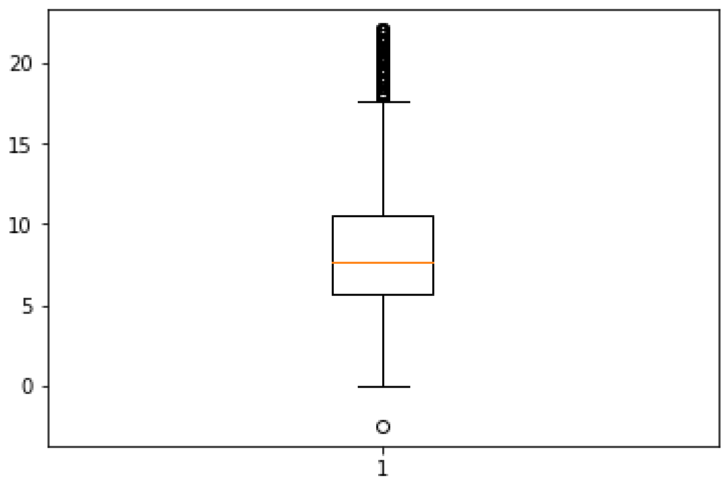
```
In [53]: plt.boxplot(cabdf['pickup_latitude'])
```

```
Out[53]: {'whiskers': [<matplotlib.lines.Line2D at 0x187da028d30>,
           <matplotlib.lines.Line2D at 0x187da1a14e0>],
          'caps': [<matplotlib.lines.Line2D at 0x187da1a1400>,
           <matplotlib.lines.Line2D at 0x187da002630>],
          'boxes': [<matplotlib.lines.Line2D at 0x187da028e10>],
          'medians': [<matplotlib.lines.Line2D at 0x187da002cc0>],
          'fliers': [<matplotlib.lines.Line2D at 0x187da0024a8>],
          'means': []}
```



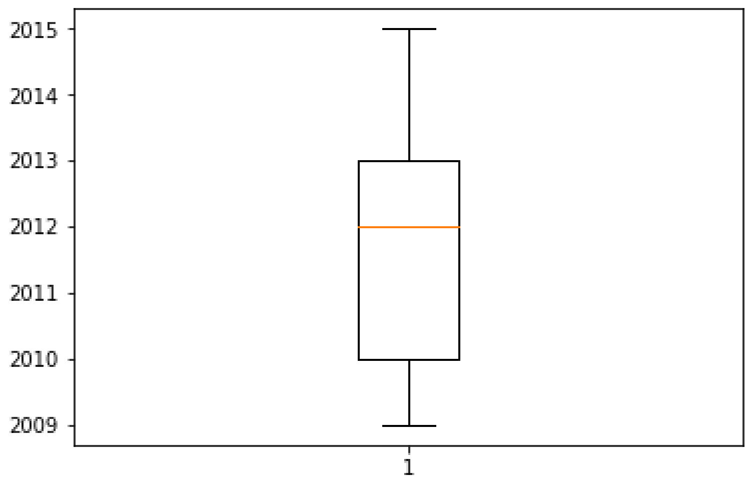
In [54]:
```
plt.boxplot(cabdf['fare_amount'])
```

Out[54]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x187da003e10>,
  <matplotlib.lines.Line2D at 0x187da1d0438>],
 'caps': [<matplotlib.lines.Line2D at 0x187da1d0780>,
  <matplotlib.lines.Line2D at 0x187da1d0ac8>],
 'boxes': [<matplotlib.lines.Line2D at 0x187da003cf8>],
 'medians': [<matplotlib.lines.Line2D at 0x187da1d0e10>],
 'fliers': [<matplotlib.lines.Line2D at 0x187da1d0ef0>],
 'means': []}
```



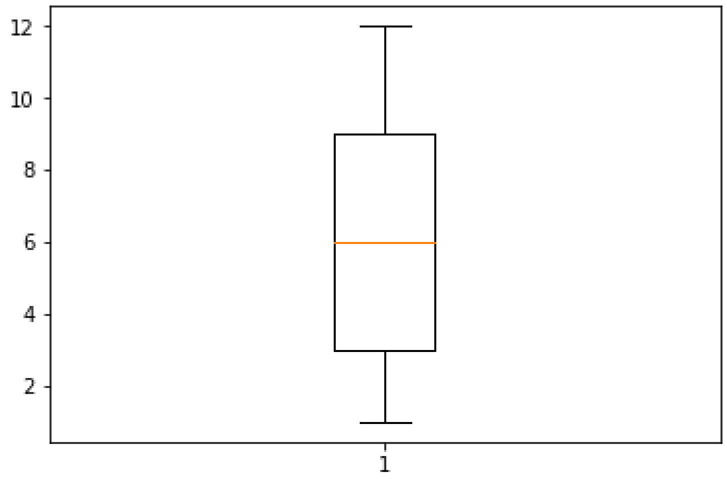
In [55]:
```
plt.boxplot(cabdf['pickup_year'])
```

Out[55]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x187da103f28>,
  <matplotlib.lines.Line2D at 0x187da0f6550>],
 'caps': [<matplotlib.lines.Line2D at 0x187da0f6898>,
  <matplotlib.lines.Line2D at 0x187da0f6be0>],
 'boxes': [<matplotlib.lines.Line2D at 0x187da103e10>],
 'medians': [<matplotlib.lines.Line2D at 0x187da0f6f28>],
 'fliers': [<matplotlib.lines.Line2D at 0x187da12c2b0>],
 'means': []}
```



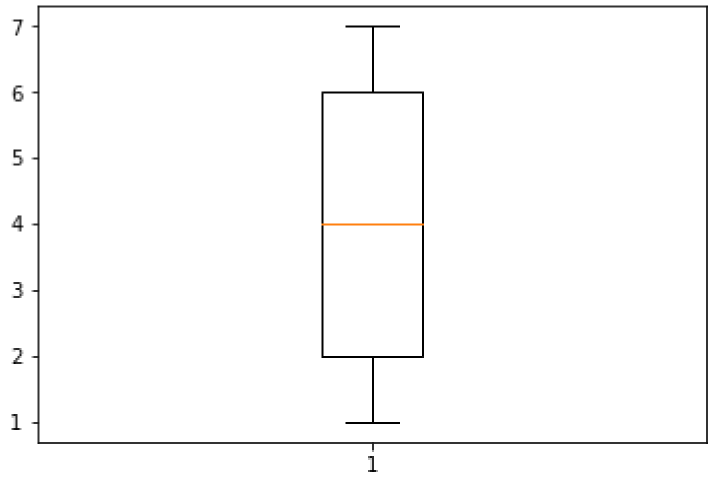
In [56]:
```
plt.boxplot(cabdf['pickup_month'])
```

Out[56]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x187da1f5eb8>,
  <matplotlib.lines.Line2D at 0x187da1f5f98>],
 'caps': [<matplotlib.lines.Line2D at 0x187da200588>,
  <matplotlib.lines.Line2D at 0x187da2008d0>],
 'boxes': [<matplotlib.lines.Line2D at 0x187da1f5b00>],
 'medians': [<matplotlib.lines.Line2D at 0x187da200c18>],
 'fliers': [<matplotlib.lines.Line2D at 0x187da200f60>],
 'means': []}
```



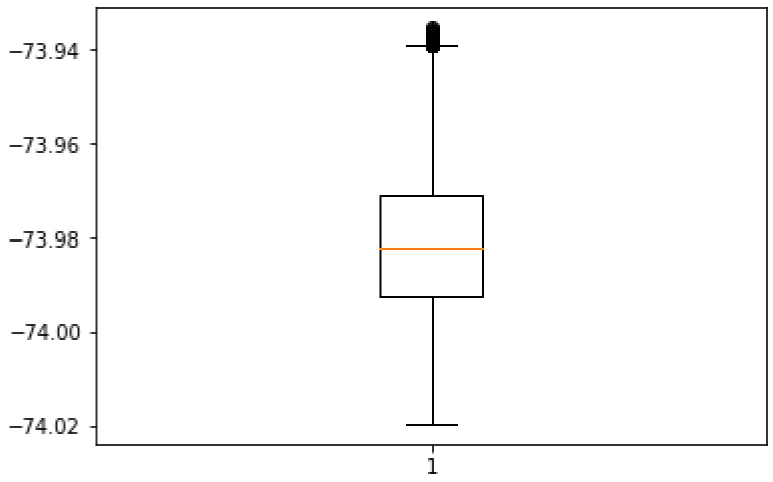
In [57]:
```
plt.boxplot(cabdf['pickup_wday'])
```

Out[57]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x187da2535c0>,
  <matplotlib.lines.Line2D at 0x187da253908>],
 'caps': [<matplotlib.lines.Line2D at 0x187da253c50>,
  <matplotlib.lines.Line2D at 0x187da253f98>],
 'boxes': [<matplotlib.lines.Line2D at 0x187da253208>],
 'medians': [<matplotlib.lines.Line2D at 0x187da25d320>],
 'fliers': [<matplotlib.lines.Line2D at 0x187da25d668>],
 'means': []}
```



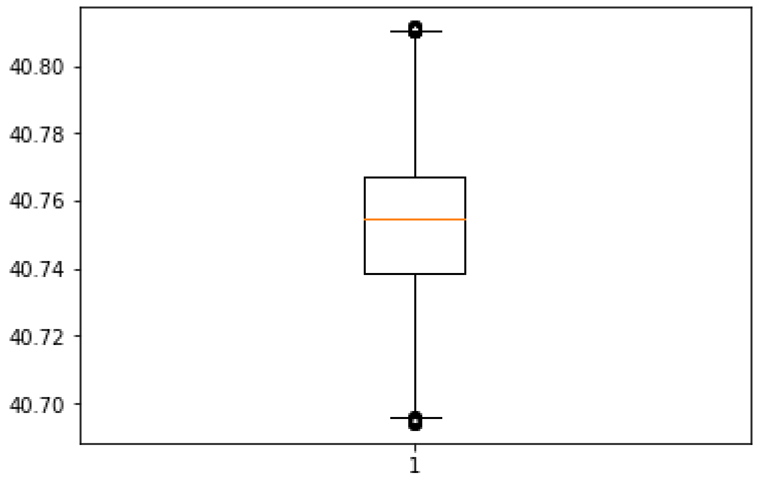
In [58]:
```
plt.boxplot(cabdf['dropoff_longitude'])
```

Out[58]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x187da2a7dd8>,
  <matplotlib.lines.Line2D at 0x187db5e0400>],
 'caps': [<matplotlib.lines.Line2D at 0x187db5e0748>,
  <matplotlib.lines.Line2D at 0x187db5e0a90>],
 'boxes': [<matplotlib.lines.Line2D at 0x187da2a7cc0>],
 'medians': [<matplotlib.lines.Line2D at 0x187db5e0dd8>],
 'fliers': [<matplotlib.lines.Line2D at 0x187db5e0eb8>],
 'means': []}
```

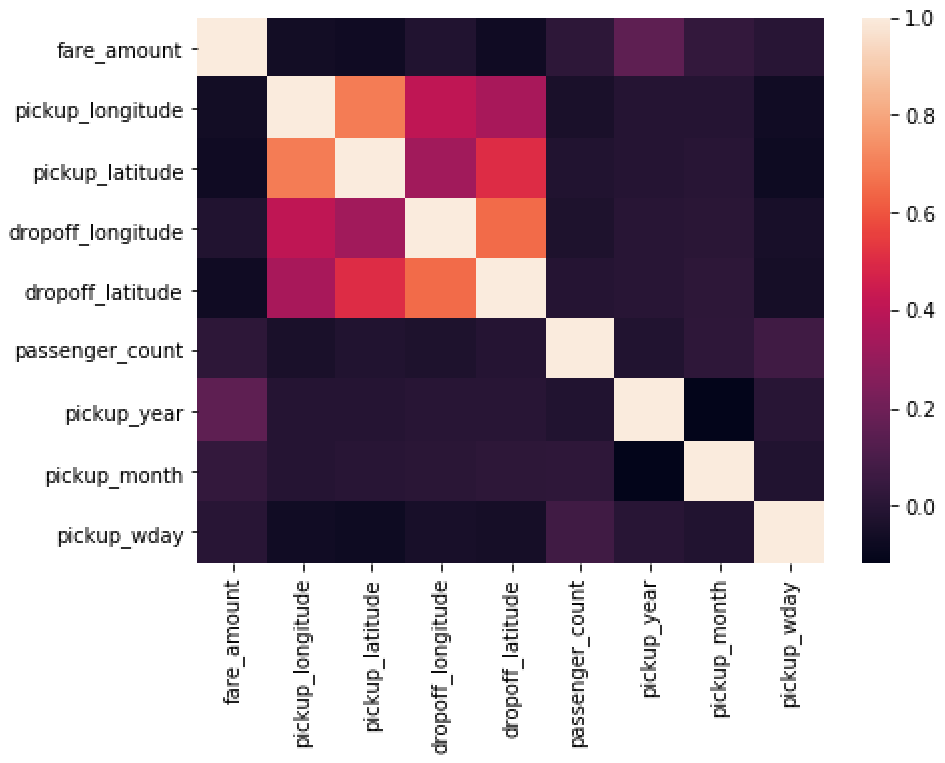


In [59]:
```
plt.boxplot(cabdf['dropoff_latitude'])
```

Out[59]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x187da1a3dd8>,
  <matplotlib.lines.Line2D at 0x187db6376d8>],
 'caps': [<matplotlib.lines.Line2D at 0x187db637a20>,
  <matplotlib.lines.Line2D at 0x187db637d68>],
 'boxes': [<matplotlib.lines.Line2D at 0x187db637048>],
 'medians': [<matplotlib.lines.Line2D at 0x187db637e48>],
 'fliers': [<matplotlib.lines.Line2D at 0x187db641438>],
 'means': []}
```



In [62]:
```
#FEATURE SELECTION
cabdf_corr=cabdf.loc[:,cnames]

f,ax=plt.subplots(figsize=(7,5))
corr=cabdf_corr.corr()
ax = sns.heatmap(corr)
```



In [64]:
```
#SAMPLING
train, test = train_test_split(cabdf, test_size=0.2)
cabdf
```

Out[64]:

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passe
1	16.900000	-74.016048	40.711303	-73.979268	40.782004	
2	5.700000	-73.982738	40.761270	-73.991242	40.750562	
3	7.700000	-73.987130	40.733143	-73.991567	40.758092	
4	5.300000	-73.968095	40.768008	-73.956655	40.783762	
5	12.100000	-74.000964	40.731630	-73.972892	40.758233	
6	7.500000	-73.980002	40.751662	-73.973802	40.764842	
7	16.500000	-73.951300	40.774138	-73.990095	40.751048	
8	15.015004	-74.006462	40.726713	-73.993078	40.731628	
9	8.900000	-73.980658	40.733873	-73.991540	40.758138	
10	5.300000	-73.996335	40.737142	-73.980721	40.733559	
12	4.100000	-73.991601	40.744712	-73.983081	40.744682	
13	7.000000	-74.005360	40.728867	-74.008913	40.710907	
14	7.700000	-74.001821	40.737547	-73.998060	40.722788	
16	12.500000	-73.986430	40.760465	-73.988990	40.737075	
17	5.300000	-73.981060	40.737690	-73.994177	40.728412	
18	5.300000	-73.969505	40.784843	-73.958732	40.783357	
19	4.000000	-73.979815	40.751902	-73.979446	40.755481	
20	10.500000	-73.985382	40.747858	-73.978377	40.762070	
21	11.500000	-73.957954	40.779252	-73.961250	40.758787	
22	4.500000	-73.991707	40.770505	-73.985459	40.763671	
23	4.900000	-74.000632	40.747473	-73.986672	40.740577	
24	6.100000	-73.969622	40.756973	-73.981152	40.759712	
25	7.300000	-73.991875	40.754437	-73.977230	40.774323	
27	4.500000	-73.988893	40.760160	-73.986445	40.757857	
29	4.500000	-73.981020	40.737760	-73.980668	40.730497	
30	5.500000	-73.976075	40.752422	-73.981082	40.759285	
33	5.700000	-73.976162	40.744988	-73.990002	40.738323	
36	4.500000	-73.990173	40.756447	-73.985619	40.762829	
37	5.300000	-73.995199	40.754740	-74.005416	40.751084	
39	9.800000	-73.972673	40.759186	-73.969897	40.791367	
...	...	...	...	...	...	
16031	5.500000	-73.971260	40.795307	-73.966110	40.806538	
16032	6.500000	-74.004509	40.724190	-74.005814	40.740253	
16033	8.500000	-73.961395	40.780161	-73.976850	40.758319	
16034	5.300000	-73.978507	40.788207	-73.968442	40.799108	

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passe
16035	9.000000	-74.000935	40.752112	-73.976602	40.760152	
16036	10.500000	-73.990103	40.729750	-73.978462	40.762383	
16037	6.500000	-73.992618	40.723878	-73.977073	40.733778	
16038	5.700000	-73.990336	40.718973	-73.956060	40.713974	
16040	6.500000	-73.980597	40.744267	-73.979330	40.731205	
16041	11.000000	-73.983610	40.747090	-73.961310	40.770980	
16042	8.500000	-73.991425	40.749832	-74.000107	40.727898	
16043	8.500000	-73.973961	40.764055	-73.986807	40.751617	
16044	16.500000	-73.982785	40.731421	-74.011358	40.713788	
16045	6.500000	-73.995227	40.733475	-73.984030	40.743287	
16046	6.000000	-73.976298	40.753948	-73.993062	40.744550	
16047	6.100000	-73.970733	40.758193	-73.979457	40.755830	
16048	9.700000	-73.988040	40.774902	-74.005265	40.744157	
16050	8.500000	-73.996715	40.742504	-73.977987	40.751805	
16051	11.500000	-73.975540	40.755590	-73.944780	40.780050	
16053	4.000000	-73.954977	40.788582	-73.964227	40.792305	
16054	5.300000	-73.993929	40.756944	-73.993044	40.744088	
16058	5.500000	-73.974265	40.756048	-73.980885	40.746838	
16059	5.300000	-73.973297	40.743768	-73.986060	40.730768	
16060	22.000000	-73.954582	40.778047	-74.005982	40.742117	
16061	10.900000	-73.994191	40.751138	-73.962769	40.769719	
16062	6.500000	-74.008820	40.718757	-73.998865	40.719987	
16063	16.100000	-73.981310	40.781695	-74.014392	40.715527	
16064	8.500000	-73.972507	40.753417	-73.979577	40.765495	
16065	8.100000	-73.957027	40.765945	-73.981983	40.779560	
16066	8.500000	-74.002111	40.729755	-73.983877	40.761975	

11878 rows × 9 columns

In [1]:
```
#LINEAR REGRESSION

model = sm.OLS(train.iloc[:,0], train.iloc[:,1:9]).fit()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-a893dc25ec92> in <module>
      1 #LINEAR REGRESSION
      2
----> 3 model = sm.OLS(train.iloc[:,0], train.iloc[:,1:9]).fit()
      4 model

NameError: name 'sm' is not defined
```

In [171]:
```
predictions_LR = model.predict(test.iloc[:,1:9])
```

In [ ]:
```
```

In [244]:
```
from sklearn.metrics import mean_squared_error
from math import sqrt

rms_LR = sqrt(mean_squared_error(test['fare_amount'],predictions_LR))
```

In [173]:
```
rms_LR
#RMSE_LR=3.6611025961214114
```

Out[173]:
```
3.6611025961214114
```

In [174]:
```
#DECISIONS TREE
fit_DT = DecisionTreeRegressor(max_depth=2).fit(train.iloc[:,1:9], train.iloc
[:,0])
```

In [166]:
```
predictions_DT = fit_DT.predict(test.iloc[:,1:9])
```

In [167]:
```
predictions_DT
```

Out[167]:
```
array([ 8.22769841,  8.22769841,  8.22769841, ..., 15.63288889,
        8.22769841,  8.22769841])
```

In [168]:
```
rms_DT = sqrt(mean_squared_error(test['fare_amount'],predictions_DT))
```

In [169]:
```
rms_DT
```

Out[169]:
```
3.5198987943291846
```

In [100]:
```
#RMSE_DT=3.5198987943291846
```

In [175]:
```
#RANDOM FOREST
RF_model = RandomForestRegressor(n_estimators = 500).fit(train.iloc[:,1:9], tr
ain.iloc[:,0])
```

In [183]:
```
predictions_RF = RF_model.predict(test.iloc[:,1:9])
```

In [179]:
```
predictions_RF
```

Out[179]:
```
array([ 9.4918    ,  7.2736    ,  8.61561024, ..., 17.9044    ,
        7.409     ,  7.7328    ])
```

In [180]:
```
rms_RF = sqrt(mean_squared_error(test['fare_amount'],predictions_RF))
```

In [181]:
```
rms_RF
```

Out[181]:
```
2.066732631353832
```

In [118]:
```
#RMSE_RF=2.066732631353832
```

In [121]:
```
#DEPLOYING FINAL MODEL TO FINAL DATASET

cabdf_final=pd.read_csv("test.csv",sep=',')
```

In [156]:
```
type(cabdf_final)
```

Out[156]:
```
pandas.core.frame.DataFrame
```

In [124]:
```
cabdf_final['pickup_datetime']=pd.to_datetime(cabdf_final['pickup_datetime'],e
rrors='coerce')
cabdf_final['pickup_year']=cabdf_final['pickup_datetime'].apply(lambda x:x.yea
r)
cabdf_final['pickup_month']=cabdf_final['pickup_datetime'].apply(lambda x:x.mo
nth)
cabdf_final['pickup_wday']=cabdf_final['pickup_datetime'].dt.strftime("%u")
cabdf_final=cabdf_final.drop(columns='pickup_datetime')
```

In [157]:
```
cabdf_final.dtypes
```

Out[157]:
```
pickup_longitude     float64
pickup_latitude      float64
dropoff_longitude    float64
dropoff_latitude     float64
passenger_count        int64
pickup_year            int64
pickup_month           int64
pickup_wday            int64
dtype: object
```

In [129]:
```
cabdf_final["pickup_wday"]=cabdf_final["pickup_wday"].convert_objects(convert_
numeric=True)
```

```
C:\Users\user\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: FutureWarn
ing: convert_objects is deprecated.  To re-infer data dtypes for object colum
ns, use Series.infer_objects()
For all other conversions use the data-type specific converters pd.to_datetim
e, pd.to_timedelta and pd.to_numeric.
  """Entry point for launching an IPython kernel.
```

In [130]:
```
#MISSING VALUE
missing_val1=pd.DataFrame(cabdf_final.isnull().sum())
```

In [131]:
```
missing_val1
```

Out[131]:

	0
pickup_longitude	0
pickup_latitude	0
dropoff_longitude	0
dropoff_latitude	0
passenger_count	0
pickup_year	0
pickup_month	0
pickup_wday	0

In [132]:
```
#CHANGING INDEX & GIVING NAME TO THE COLUMN
missing_val1=missing_val1.reset_index()

missing_val1=missing_val1.rename(columns={'index':'Variables',0:'Missing Valu
e'})
```

In [133]:
```
missing_val1
```

Out[133]:

	Variables	Missing Value
0	pickup_longitude	0
1	pickup_latitude	0
2	dropoff_longitude	0
3	dropoff_latitude	0
4	passenger_count	0
5	pickup_year	0
6	pickup_month	0
7	pickup_wday	0

```
In [ ]:
```

```
In [185]: #DEPLOYMENT OF BEST MODEL into FINAL DATASET
          predictions_RF1 = RF_model.predict(cabdf_final.iloc[:,0:8])
```

```
In [186]: predictions_RF1
```

```
Out[186]: array([ 9.2688    ,  8.933     ,  5.1536    , ..., 12.8448    ,
                 16.84259001,  7.5042    ])
```

```
In [238]: #READING THE ORIGINAL FILE IN SAME FORMAT
          cabdf_final1=pd.read_csv("test.csv",sep=',')
          #ADDING THE PREICTED VALUE TO THE DATASET AFTER CONVERTING ARRAY TO DATAFRAME
          predictions_RF1=pd.DataFrame(predictions_RF1)
          cabdf_final1=pd.concat([cabdf_final1.reset_index(drop=True),predictions_RF1],a
          xis=1)
```

In [239]:
```
cabdf_final1
```

Out[239]:

	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	pas
0	2015-01-27 13:08:24 UTC	-73.973320	40.763805	-73.981430	40.743835	
1	2015-01-27 13:08:24 UTC	-73.986862	40.719383	-73.998886	40.739201	
2	2011-10-08 11:53:44 UTC	-73.982524	40.751260	-73.979654	40.746139	
3	2012-12-01 21:12:12 UTC	-73.981160	40.767807	-73.990448	40.751635	
4	2012-12-01 21:12:12 UTC	-73.966046	40.789775	-73.988565	40.744427	
5	2012-12-01 21:12:12 UTC	-73.960983	40.765547	-73.979177	40.740053	
6	2011-10-06 12:10:20 UTC	-73.949013	40.773204	-73.959622	40.770893	
7	2011-10-06 12:10:20 UTC	-73.777282	40.646636	-73.985083	40.759368	
8	2011-10-06 12:10:20 UTC	-74.014099	40.709638	-73.995106	40.741365	
9	2014-02-18 15:22:20 UTC	-73.969582	40.765519	-73.980686	40.770725	
10	2014-02-18 15:22:20 UTC	-73.989374	40.741973	-73.999300	40.722534	
11	2014-02-18 15:22:20 UTC	-74.001614	40.740893	-73.956387	40.767437	
12	2010-03-29 20:20:32 UTC	-73.991198	40.739937	-73.997166	40.735269	
13	2010-03-29 20:20:32 UTC	-73.982034	40.762723	-74.001867	40.761545	
14	2011-10-06 03:59:12 UTC	-73.992455	40.728701	-73.983397	40.750149	
15	2011-10-06 03:59:12 UTC	-73.983583	40.746993	-73.951178	40.785903	
16	2012-07-15 16:45:04 UTC	-74.006746	40.731721	-74.010204	40.732318	
17	2012-07-15 16:45:04 UTC	-73.976446	40.785598	-73.952220	40.772121	
18	2012-07-15 16:45:04 UTC	-73.973548	40.763349	-73.972096	40.756417	
19	2012-07-15 16:45:04 UTC	-73.970918	40.756025	-73.975954	40.755563	
20	2014-10-29 02:09:56 UTC	-73.926071	40.705866	-73.941741	40.714789	
21	2014-06-14 13:39:00 UTC	-73.970555	40.764702	-73.949132	40.771800	
22	2014-06-14 13:39:00 UTC	-73.989102	40.736360	-73.992767	40.747767	

	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	pas
23	2014-06-14 13:39:00 UTC	-74.003525	40.748480	-73.991520	40.762960	
24	2014-06-14 13:39:00 UTC	-73.990352	40.759992	-74.015665	40.711682	
25	2014-06-14 13:39:00 UTC	-73.989482	40.757450	-74.000850	40.762705	
26	2014-06-14 13:39:00 UTC	-73.870785	40.773722	-73.741922	40.689945	
27	2014-06-14 13:39:00 UTC	-73.992682	40.733877	-73.938852	40.808220	
28	2014-06-14 13:39:00 UTC	-73.954020	40.778705	-73.950277	40.768810	
29	2014-06-14 13:39:00 UTC	-73.972742	40.743432	-74.007125	40.710192	
...	...	...	...	...	...	
9884	2013-09-25 22:00:00 UTC	-73.790022	40.643817	-73.735688	40.773400	
9885	2013-09-25 22:00:00 UTC	-74.007878	40.722762	-73.965740	40.754505	
9886	2013-09-25 22:00:00 UTC	-73.978852	40.752837	-73.941152	40.812722	
9887	2013-09-25 22:00:00 UTC	-73.959087	40.783282	-73.978802	40.785655	
9888	2013-09-25 22:00:00 UTC	-73.956488	40.767512	-73.956488	40.767512	
9889	2013-09-25 22:00:00 UTC	-73.966650	40.714675	-73.971912	40.693667	
9890	2013-09-25 22:00:00 UTC	-73.976602	40.754152	-73.993297	40.730887	
9891	2013-09-25 22:00:00 UTC	-73.987185	40.760505	-73.938755	40.799507	
9892	2013-09-25 22:00:00 UTC	-73.969175	40.757770	-73.952318	40.781030	
9893	2013-09-25 22:00:00 UTC	-73.949657	40.796197	-73.911755	40.827672	
9894	2013-09-25 22:00:00 UTC	-74.002267	40.730415	-73.990360	40.756807	
9895	2013-09-25 22:00:00 UTC	-73.985840	40.731167	-73.953883	40.653937	
9896	2013-09-25 22:00:00 UTC	-73.955490	40.776862	-73.982162	40.769302	
9897	2015-02-20 11:08:29 UTC	-73.965782	40.805538	-73.982384	40.761600	
9898	2015-01-12 15:36:37 UTC	-73.979042	40.777515	-73.983658	40.781082	
9899	2015-06-07 00:38:14 UTC	-73.983238	40.764874	-73.922928	40.743458	

	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	pas
9900	2015-04-12 21:56:22 UTC	-73.962952	40.772480	-73.976051	40.786289	
9901	2015-04-10 11:56:54 UTC	-73.977943	40.762753	-73.976219	40.776451	
9902	2015-06-25 01:01:46 UTC	-73.905525	40.752655	-73.864151	40.737091	
9903	2015-05-29 10:02:42 UTC	-73.988403	40.738731	-73.992340	40.759193	
9904	2015-06-30 20:03:50 UTC	-73.776848	40.645035	-73.955460	40.652458	
9905	2015-02-27 19:36:02 UTC	-73.989647	40.767406	-73.941177	40.845696	
9906	2015-06-15 01:00:06 UTC	-73.988052	40.720776	-73.991043	40.718346	
9907	2015-02-03 09:00:58 UTC	-73.863457	40.769611	-73.980995	40.763241	
9908	2015-05-19 13:58:11 UTC	-73.987968	40.718922	-73.982124	40.732956	
9909	2015-05-10 12:37:51 UTC	-73.968124	40.796997	-73.955643	40.780388	
9910	2015-01-12 17:05:51 UTC	-73.945511	40.803600	-73.960213	40.776371	
9911	2015-04-19 20:44:15 UTC	-73.991600	40.726608	-73.789742	40.647011	
9912	2015-01-31 01:05:19 UTC	-73.985573	40.735432	-73.939178	40.801731	
9913	2015-01-18 14:06:23 UTC	-73.988022	40.754070	-74.000282	40.759220	

9914 rows × 7 columns

```
In [ ]:
```

```
In [240]: cabdf_final1.columns=[ 'pickup_datetime','pickup_longitude', 'pickup_latitude'
,
       'dropoff_longitude', 'dropoff_latitude', 'passenger_count','fare_amoun
t']
```

In [251]:
```
cabdf_final1
```

Out[251]:

	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	pas
0	2015-01-27 13:08:24 UTC	-73.973320	40.763805	-73.981430	40.743835	
1	2015-01-27 13:08:24 UTC	-73.986862	40.719383	-73.998886	40.739201	
2	2011-10-08 11:53:44 UTC	-73.982524	40.751260	-73.979654	40.746139	
3	2012-12-01 21:12:12 UTC	-73.981160	40.767807	-73.990448	40.751635	
4	2012-12-01 21:12:12 UTC	-73.966046	40.789775	-73.988565	40.744427	
5	2012-12-01 21:12:12 UTC	-73.960983	40.765547	-73.979177	40.740053	
6	2011-10-06 12:10:20 UTC	-73.949013	40.773204	-73.959622	40.770893	
7	2011-10-06 12:10:20 UTC	-73.777282	40.646636	-73.985083	40.759368	
8	2011-10-06 12:10:20 UTC	-74.014099	40.709638	-73.995106	40.741365	
9	2014-02-18 15:22:20 UTC	-73.969582	40.765519	-73.980686	40.770725	
10	2014-02-18 15:22:20 UTC	-73.989374	40.741973	-73.999300	40.722534	
11	2014-02-18 15:22:20 UTC	-74.001614	40.740893	-73.956387	40.767437	
12	2010-03-29 20:20:32 UTC	-73.991198	40.739937	-73.997166	40.735269	
13	2010-03-29 20:20:32 UTC	-73.982034	40.762723	-74.001867	40.761545	
14	2011-10-06 03:59:12 UTC	-73.992455	40.728701	-73.983397	40.750149	
15	2011-10-06 03:59:12 UTC	-73.983583	40.746993	-73.951178	40.785903	
16	2012-07-15 16:45:04 UTC	-74.006746	40.731721	-74.010204	40.732318	
17	2012-07-15 16:45:04 UTC	-73.976446	40.785598	-73.952220	40.772121	
18	2012-07-15 16:45:04 UTC	-73.973548	40.763349	-73.972096	40.756417	
19	2012-07-15 16:45:04 UTC	-73.970918	40.756025	-73.975954	40.755563	
20	2014-10-29 02:09:56 UTC	-73.926071	40.705866	-73.941741	40.714789	
21	2014-06-14 13:39:00 UTC	-73.970555	40.764702	-73.949132	40.771800	
22	2014-06-14 13:39:00 UTC	-73.989102	40.736360	-73.992767	40.747767	

	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	pas
23	2014-06-14 13:39:00 UTC	-74.003525	40.748480	-73.991520	40.762960	
24	2014-06-14 13:39:00 UTC	-73.990352	40.759992	-74.015665	40.711682	
25	2014-06-14 13:39:00 UTC	-73.989482	40.757450	-74.000850	40.762705	
26	2014-06-14 13:39:00 UTC	-73.870785	40.773722	-73.741922	40.689945	
27	2014-06-14 13:39:00 UTC	-73.992682	40.733877	-73.938852	40.808220	
28	2014-06-14 13:39:00 UTC	-73.954020	40.778705	-73.950277	40.768810	
29	2014-06-14 13:39:00 UTC	-73.972742	40.743432	-74.007125	40.710192	
...	...	...	...	...	...	
9884	2013-09-25 22:00:00 UTC	-73.790022	40.643817	-73.735688	40.773400	
9885	2013-09-25 22:00:00 UTC	-74.007878	40.722762	-73.965740	40.754505	
9886	2013-09-25 22:00:00 UTC	-73.978852	40.752837	-73.941152	40.812722	
9887	2013-09-25 22:00:00 UTC	-73.959087	40.783282	-73.978802	40.785655	
9888	2013-09-25 22:00:00 UTC	-73.956488	40.767512	-73.956488	40.767512	
9889	2013-09-25 22:00:00 UTC	-73.966650	40.714675	-73.971912	40.693667	
9890	2013-09-25 22:00:00 UTC	-73.976602	40.754152	-73.993297	40.730887	
9891	2013-09-25 22:00:00 UTC	-73.987185	40.760505	-73.938755	40.799507	
9892	2013-09-25 22:00:00 UTC	-73.969175	40.757770	-73.952318	40.781030	
9893	2013-09-25 22:00:00 UTC	-73.949657	40.796197	-73.911755	40.827672	
9894	2013-09-25 22:00:00 UTC	-74.002267	40.730415	-73.990360	40.756807	
9895	2013-09-25 22:00:00 UTC	-73.985840	40.731167	-73.953883	40.653937	
9896	2013-09-25 22:00:00 UTC	-73.955490	40.776862	-73.982162	40.769302	
9897	2015-02-20 11:08:29 UTC	-73.965782	40.805538	-73.982384	40.761600	
9898	2015-01-12 15:36:37 UTC	-73.979042	40.777515	-73.983658	40.781082	
9899	2015-06-07 00:38:14 UTC	-73.983238	40.764874	-73.922928	40.743458	

	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	pas
9900	2015-04-12 21:56:22 UTC	-73.962952	40.772480	-73.976051	40.786289	
9901	2015-04-10 11:56:54 UTC	-73.977943	40.762753	-73.976219	40.776451	
9902	2015-06-25 01:01:46 UTC	-73.905525	40.752655	-73.864151	40.737091	
9903	2015-05-29 10:02:42 UTC	-73.988403	40.738731	-73.992340	40.759193	
9904	2015-06-30 20:03:50 UTC	-73.776848	40.645035	-73.955460	40.652458	
9905	2015-02-27 19:36:02 UTC	-73.989647	40.767406	-73.941177	40.845696	
9906	2015-06-15 01:00:06 UTC	-73.988052	40.720776	-73.991043	40.718346	
9907	2015-02-03 09:00:58 UTC	-73.863457	40.769611	-73.980995	40.763241	
9908	2015-05-19 13:58:11 UTC	-73.987968	40.718922	-73.982124	40.732956	
9909	2015-05-10 12:37:51 UTC	-73.968124	40.796997	-73.955643	40.780388	
9910	2015-01-12 17:05:51 UTC	-73.945511	40.803600	-73.960213	40.776371	
9911	2015-04-19 20:44:15 UTC	-73.991600	40.726608	-73.789742	40.647011	
9912	2015-01-31 01:05:19 UTC	-73.985573	40.735432	-73.939178	40.801731	
9913	2015-01-18 14:06:23 UTC	-73.988022	40.754070	-74.000282	40.759220	

9914 rows × 7 columns

```
In [ ]:
```

```
In [ ]:
```

```
In [242]: #Writing the final dataset into HDD
          cabdf_final1.to_csv("cab_fare_prediction_python.csv",index=False)
```

```
In [ ]:
```