

Madhur Jaripatke

Roll No. 52

BE A Computer

RMDSSOE, Warje, Pune

Predict the price of the Uber ride from a given pickup point to the agreed drop-off location. Perform the following tasks:

1. Pre-process the dataset.
2. Identify outliers.
3. Check the correlation.
4. Implement linear regression and random forest regression models.
5. Evaluate the models and compare their respective scores like R2, RMSE, etc.

Dataset link: <https://www.kaggle.com/datasets/yasserh/uber-fares-dataset>

Importing Libraries

```
In [1]: import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import haversine as hs
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error, root_mean_squared_error
```

Loading The Dataset

```
In [9]: df = pd.read_csv('./Datasets/uber.csv')
df
```

Out[9]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup
0	24238194	2015-05-07 19:52:06.00000003	7.5	2015-05-07 19:52:06 UTC	
1	27835199	2009-07-17 20:04:56.00000002	7.7	2009-07-17 20:04:56 UTC	
2	44984355	2009-08-24 21:45:00.000000061	12.9	2009-08-24 21:45:00 UTC	
3	25894730	2009-06-26 08:22:21.00000001	5.3	2009-06-26 08:22:21 UTC	
4	17610152	2014-08-28 17:47:00.000000188	16.0	2014-08-28 17:47:00 UTC	
...
199995	42598914	2012-10-28 10:49:00.000000053	3.0	2012-10-28 10:49:00 UTC	
199996	16382965	2014-03-14 01:09:00.00000008	7.5	2014-03-14 01:09:00 UTC	
199997	27804658	2009-06-29 00:42:00.000000078	30.9	2009-06-29 00:42:00 UTC	
199998	20259894	2015-05-20 14:56:25.00000004	14.5	2015-05-20 14:56:25 UTC	
199999	11951496	2010-05-15 04:08:00.000000076	14.1	2010-05-15 04:08:00 UTC	

200000 rows × 9 columns

In [3]: `df.head()`

Out[3]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longi
0	24238194	2015-05-07 19:52:06.00000003	7.5	2015-05-07 19:52:06 UTC	-73.99
1	27835199	2009-07-17 20:04:56.00000002	7.7	2009-07-17 20:04:56 UTC	-73.99
2	44984355	2009-08-24 21:45:00.000000061	12.9	2009-08-24 21:45:00 UTC	-74.00
3	25894730	2009-06-26 08:22:21.00000001	5.3	2009-06-26 08:22:21 UTC	-73.97
4	17610152	2014-08-28 17:47:00.000000188	16.0	2014-08-28 17:47:00 UTC	-73.92

In [4]: `df.info()`

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200000 entries, 0 to 199999
Data columns (total 9 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Unnamed: 0            200000 non-null  int64
1   key                   200000 non-null  object
2   fare_amount          200000 non-null  float64
3   pickup_datetime      200000 non-null  object
4   pickup_longitude     200000 non-null  float64
5   pickup_latitude      200000 non-null  float64
6   dropoff_longitude    199999 non-null  float64
7   dropoff_latitude     199999 non-null  float64
8   passenger_count      200000 non-null  int64
dtypes: float64(5), int64(2), object(2)
memory usage: 13.7+ MB

```

```
In [5]: df.columns
```

```

Out[5]: Index(['Unnamed: 0', 'key', 'fare_amount', 'pickup_datetime',
              'pickup_longitude', 'pickup_latitude', 'dropoff_longitude',
              'dropoff_latitude', 'passenger_count'],
             dtype='object')

```

Data Preprocessing

```
In [6]: df = df.drop(['Unnamed: 0', 'key'], axis = 1)
```

```
In [7]: df.shape
```

```
Out[7]: (200000, 7)
```

```
In [8]: df.info()
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200000 entries, 0 to 199999
Data columns (total 7 columns):
#   Column                Non-Null Count  Dtype
---  -
0   fare_amount          200000 non-null  float64
1   pickup_datetime      200000 non-null  object
2   pickup_longitude     200000 non-null  float64
3   pickup_latitude      200000 non-null  float64
4   dropoff_longitude    199999 non-null  float64
5   dropoff_latitude     199999 non-null  float64
6   passenger_count      200000 non-null  int64
dtypes: float64(5), int64(1), object(1)
memory usage: 10.7+ MB

```

```
In [9]: df.describe()
```

Out[9]:

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dr
count	200000.000000	200000.000000	200000.000000	199999.000000	
mean	11.359955	-72.527638	39.935885	-72.525292	
std	9.901776	11.437787	7.720539	13.117408	
min	-52.000000	-1340.648410	-74.015515	-3356.666300	
25%	6.000000	-73.992065	40.734796	-73.991407	
50%	8.500000	-73.981823	40.752592	-73.980093	
75%	12.500000	-73.967154	40.767158	-73.963658	
max	499.000000	57.418457	1644.421482	1153.572603	

In [10]: `df.isnull().sum()`

Out[10]:

```

fare_amount      0
pickup_datetime  0
pickup_longitude  0
pickup_latitude   0
dropoff_longitude 1
dropoff_latitude  1
passenger_count   0
dtype: int64

```

In [11]: `df.dtypes`

Out[11]:

```

fare_amount      float64
pickup_datetime   object
pickup_longitude  float64
pickup_latitude   float64
dropoff_longitude  float64
dropoff_latitude  float64
passenger_count   int64
dtype: object

```

In [12]: `df.pickup_datetime = pd.to_datetime(df.pickup_datetime, errors='coerce')`

In [13]: `df.dtypes`

Out[13]:

```

fare_amount      float64
pickup_datetime  datetime64[ns, UTC]
pickup_longitude  float64
pickup_latitude   float64
dropoff_longitude  float64
dropoff_latitude  float64
passenger_count   int64
dtype: object

```

In [14]: `df = df.assign(hour = df.pickup_datetime.dt.hour,
 day = df.pickup_datetime.dt.day,
 month = df.pickup_datetime.dt.month,`

```
year = df.pickup_datetime.dt.year,
dayofweek = df.pickup_datetime.dt.dayofweek)
```

```
In [15]: df.head()
```

```
Out[15]:
```

	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude
0	7.5	2015-05-07 19:52:06+00:00	-73.999817	40.738354	-73.999817
1	7.7	2009-07-17 20:04:56+00:00	-73.994355	40.728225	-73.994355
2	12.9	2009-08-24 21:45:00+00:00	-74.005043	40.740770	-74.005043
3	5.3	2009-06-26 08:22:21+00:00	-73.976124	40.790844	-73.976124
4	16.0	2014-08-28 17:47:00+00:00	-73.925023	40.744085	-73.925023

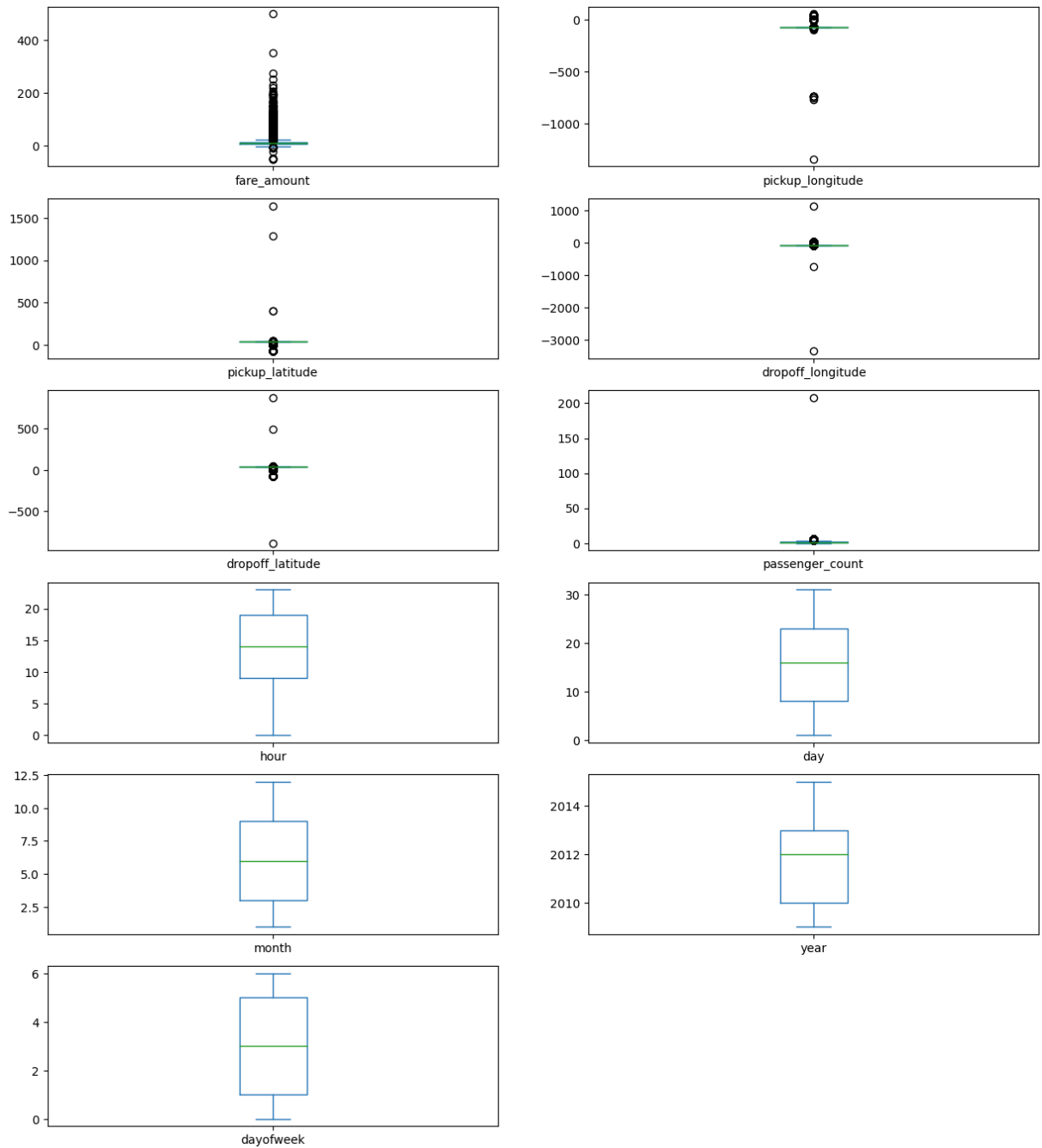
```
In [16]: df = df.drop('pickup_datetime',axis=1)
df.dtypes
```

```
Out[16]: fare_amount      float64
pickup_longitude    float64
pickup_latitude     float64
dropoff_longitude   float64
dropoff_latitude    float64
passenger_count     int64
hour                int32
day                 int32
month               int32
year                int32
dayofweek           int32
dtype: object
```

Exploratory Data Analysis

```
In [17]: df.plot(kind = "box",subplots = True,layout = (7,2),figsize=(15,20))
```

```
Out[17]: fare_amount      AxesSubplot(0.125,0.786098;0.352273x0.0939024)
pickup_longitude    AxesSubplot(0.547727,0.786098;0.352273x0.0939024)
pickup_latitude     AxesSubplot(0.125,0.673415;0.352273x0.0939024)
dropoff_longitude   AxesSubplot(0.547727,0.673415;0.352273x0.0939024)
dropoff_latitude    AxesSubplot(0.125,0.560732;0.352273x0.0939024)
passenger_count     AxesSubplot(0.547727,0.560732;0.352273x0.0939024)
hour                AxesSubplot(0.125,0.448049;0.352273x0.0939024)
day                 AxesSubplot(0.547727,0.448049;0.352273x0.0939024)
month               AxesSubplot(0.125,0.335366;0.352273x0.0939024)
year                AxesSubplot(0.547727,0.335366;0.352273x0.0939024)
dayofweek           AxesSubplot(0.125,0.222683;0.352273x0.0939024)
dtype: object
```



```
In [18]: def remove_outlier(df1 , col):
    Q1 = df1[col].quantile(0.25)
    Q3 = df1[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_whisker = Q1-1.5*IQR
    upper_whisker = Q3+1.5*IQR
    df[col] = np.clip(df1[col] , lower_whisker , upper_whisker)
    return df1

def treat_outliers_all(df1 , col_list):
    for c in col_list:
        df1 = remove_outlier(df , c)
    return df1
```

```
In [19]: df = treat_outliers_all(df , df.iloc[: , 0::])
```

```
In [20]: travel_dist = []
for pos in range(len(df['pickup_longitude'])):
    long1,lati1,long2,lati2 = [df['pickup_longitude'][pos],df['pickup_latitude'][pos],df['dropoff_longitude'][pos],df['dropoff_latitude'][pos]]
    loc1=(lati1,long1)
    loc2=(lati2,long2)
    c = hs.haversine(loc1,loc2)
    travel_dist.append(c)
print(travel_dist)
df['dist_travel_km'] = travel_dist
df.head()
```

IOPub data rate exceeded.
The notebook server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--NotebookApp.iopub_data_rate_limit`.

Current values:

NotebookApp.iopub_data_rate_limit=1000000.0 (bytes/sec)
NotebookApp.rate_limit_window=3.0 (secs)

```
Out[20]:
```

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude
0	7.5	-73.999817	40.738354	-73.999512	40.738354
1	7.7	-73.994355	40.728225	-73.994710	40.728225
2	12.9	-74.005043	40.740770	-73.962565	40.740770
3	5.3	-73.976124	40.790844	-73.965316	40.790844
4	16.0	-73.929786	40.744085	-73.973082	40.744085

```
In [21]: df= df.loc[(df.dist_travel_km >= 1) | (df.dist_travel_km <= 130)]
print('Observations left in the dataset:', df.shape)
```

Observations left in the dataset: (199999, 12)

```
In [22]: incorrect_coordinates = df.loc[(df.pickup_latitude > 90) | (df.pickup_latitude < -90)]
```

```
In [23]: df.drop(incorrect_coordinates, inplace = True, errors = 'ignore')
```

/tmp/ipykernel_40972/2959381030.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

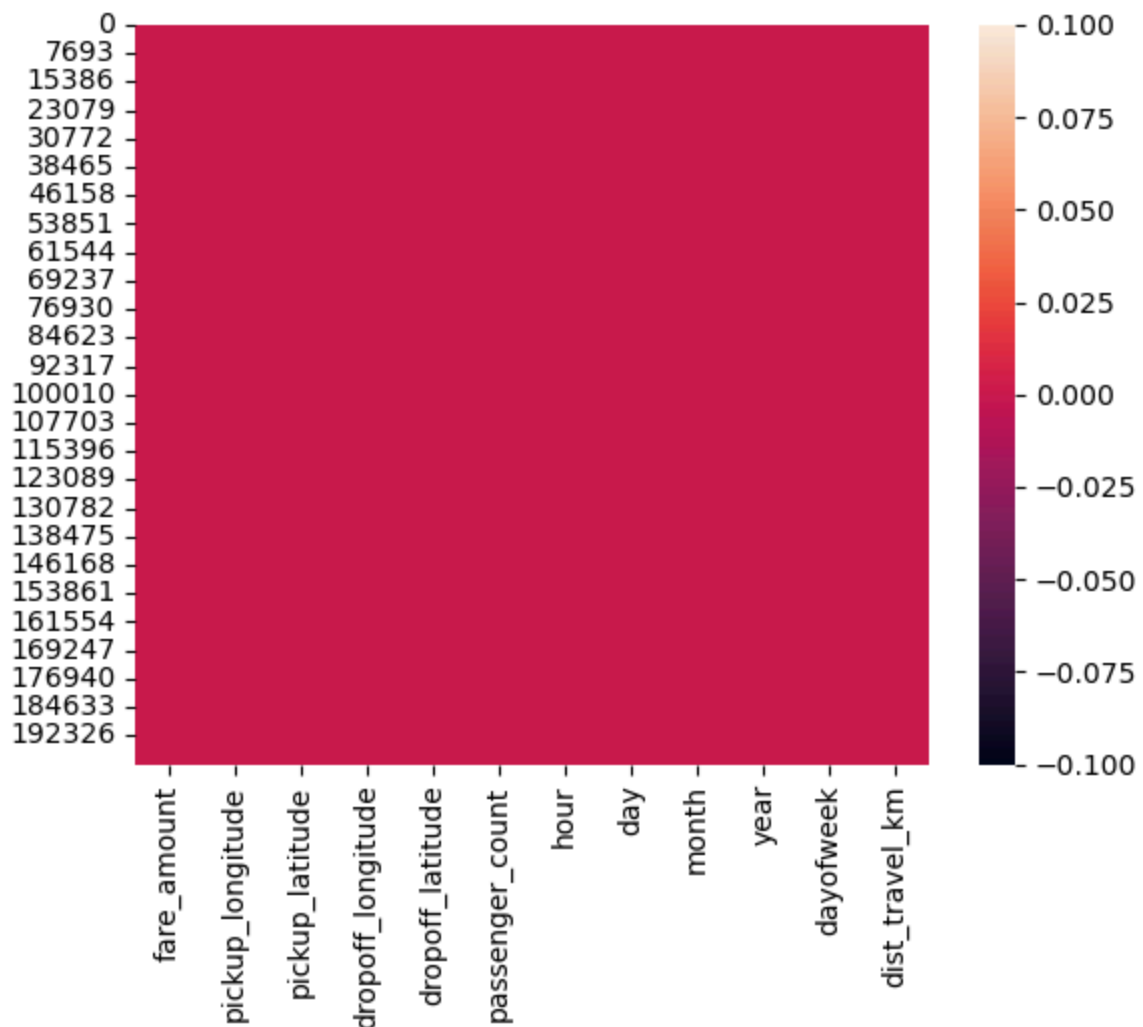
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
df.drop(incorrect_coordinates, inplace = True, errors = 'ignore')

```
In [24]: df.isnull().sum()
```

```
Out[24]: fare_amount      0
pickup_longitude      0
pickup_latitude      0
dropoff_longitude      0
dropoff_latitude      0
passenger_count      0
hour      0
day      0
month      0
year      0
dayofweek      0
dist_travel_km      0
dtype: int64
```

```
In [25]: sns.heatmap(df.isnull())
```

```
Out[25]: <AxesSubplot: >
```



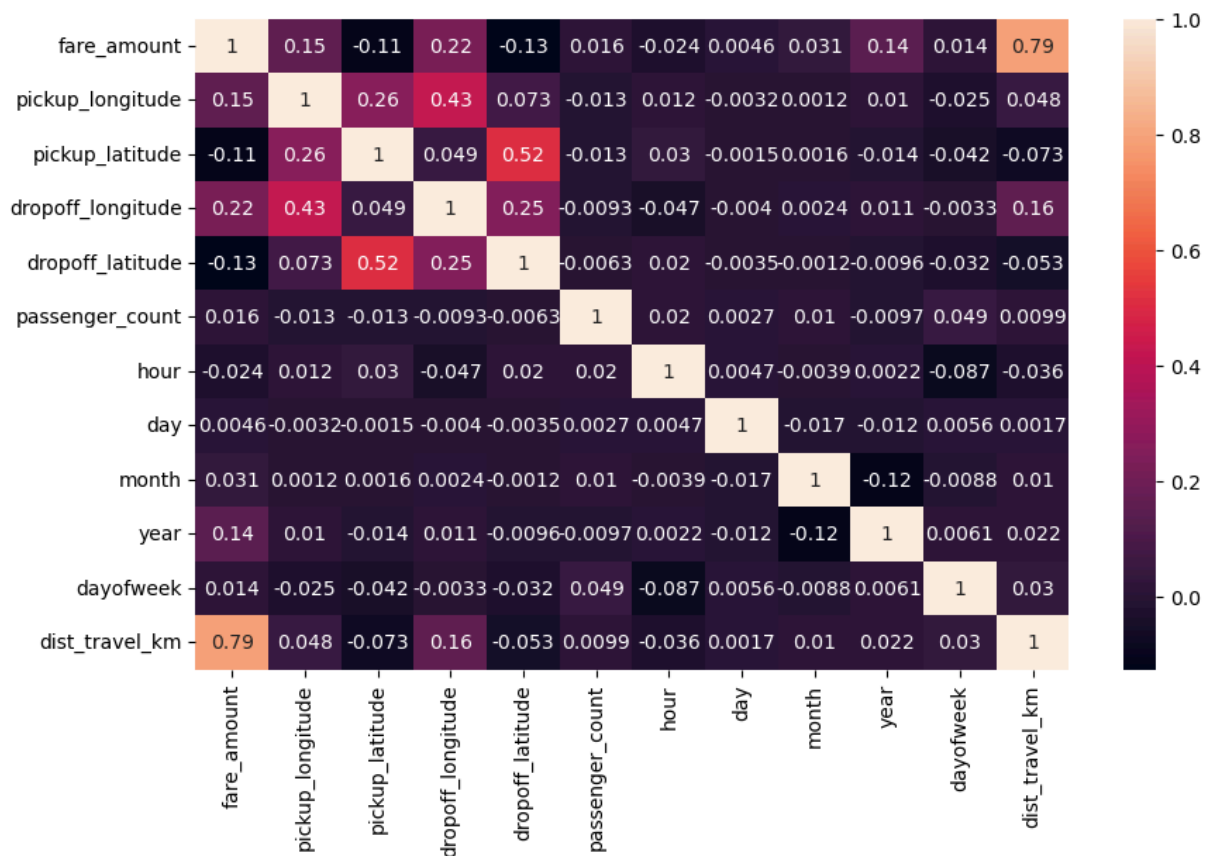
```
In [26]: corr = df.corr()
corr
```


Out[26]:

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude
fare_amount	1.000000	0.154056	-0.110856	0.218681
pickup_longitude	0.154056	1.000000	0.259492	0.425622
pickup_latitude	-0.110856	0.259492	1.000000	0.048889
dropoff_longitude	0.218681	0.425622	0.048889	1.000000
dropoff_latitude	-0.125874	0.073309	0.515736	0.073309
passenger_count	0.015798	-0.013202	-0.012879	-0.012879
hour	-0.023605	0.011590	0.029691	-0.023605
day	0.004552	-0.003194	-0.001544	-0.003194
month	0.030815	0.001168	0.001561	0.001168
year	0.141271	0.010193	-0.014247	0.010193
dayofweek	0.013664	-0.024645	-0.042304	-0.024645
dist_travel_km	0.786381	0.048423	-0.073385	0.048423

```
In [27]: fig,axis = plt.subplots(figsize = (10,6))
sns.heatmap(df.corr(),annot = True)
```

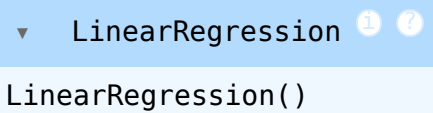
Out[27]: <AxesSubplot: >



Model Building

```
In [28]: x = df[['pickup_longitude', 'pickup_latitude', 'dropoff_longitude', 'dropoff_latitude']  
y = df['fare_amount']
```

```
In [29]: x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.33)  
regression = LinearRegression()  
regression.fit(x_train, y_train)
```

```
Out[29]:  LinearRegression()  
LinearRegression()
```

```
In [30]: regression.intercept_
```

```
Out[30]: 3647.6226066990603
```

```
In [31]: regression.coef_
```

```
Out[31]: array([ 2.47615446e+01, -7.13551346e+00,  2.03952851e+01, -1.83503695e+01,  
                7.59135481e-02,  6.73761284e-03,  4.06398003e-03,  6.05257343e-02,  
                3.65797973e-01, -3.64125042e-02,  1.84863771e+00])
```

```
In [32]: prediction = regression.predict(x_test)  
print('Prediction for x:\n', prediction, '\n')  
print('Fare Amount test data:\n', y_test)
```

Prediction for x:

```
[ 6.49965238 12.86669939  5.21442901 ...  6.90707433  7.78411418  
 9.19531409]
```

Fare Amount test data:

```
19721      5.0  
111803     12.1  
158381      5.3  
150702     17.5  
15501      6.5
```

```
...  
29975     12.0  
19243     11.0  
173765      4.1  
55536      6.9  
175297      6.5
```

Name: fare_amount, Length: 66000, dtype: float64

```
In [33]: print('R2 Score:\n', r2_score(y_test, prediction))
```

R2 Score:

```
0.6614484665343205
```

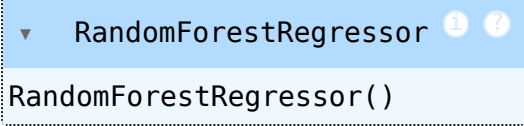
```
In [34]: MSE = mean_squared_error(y_test, prediction)  
print('Mean Squared Error:\n', MSE)
```

Mean Squared Error:
9.989764338740565

```
In [35]: RMSE = root_mean_squared_error(y_test,prediction)
print('Root Mean Squared Error:\n',RMSE)
```

Root Mean Squared Error:
3.1606588456745164

```
In [36]: rf = RandomForestRegressor(n_estimators=100)
rf.fit(x_train,y_train)
```

Out[36]: 

RandomForestRegressor()

Results

```
In [37]: y_pred = rf.predict(x_test)
print('Predictions for Fare Amount:\n',y_pred)
```

Predictions for Fare Amount:
[6.345 12.452 6.224 ... 7.509 8.254 8.116]

```
In [38]: R2_Random = r2_score(y_test,y_pred)
print('Random R2 Score:\n',R2_Random)
```

Random R2 Score:
0.7933139695621463

```
In [39]: MSE_Random = mean_squared_error(y_test,y_pred)
print('Random Mean Squared Error:\n',MSE_Random)
```

Random Mean Squared Error:
6.098760549236241

```
In [40]: RMSE_Random = root_mean_squared_error(y_test,y_pred)
print('Random Root Mean Squared Error:\n',RMSE_Random)
```

Random Root Mean Squared Error:
2.4695668748256727