Machine Learning Ass. 1

BE_34_Samruddhi Khairnar

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

Dataset link:https://www.kaggle.com/datasets/yasserh/uber-fares-dataset (https://www.kaggle.com/datasets/yasserh/uber-fares-dataset)

In [408]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, c
all drive.mount("/content/drive", force_remount=True).
```

In [409]:
```
%cd /content/drive/MyDrive/BE_Datasets/
```

```
/content/drive/MyDrive/BE_Datasets
```

In [410]:
```
%ls
```

```
6_dataset.csv  emails.csv   locations.txt          uber.csv
diabetes.csv   lat_lon.txt  sales_data_sample.csv
```

1. Pre-process the dataset.

In [411]:
```
import pandas as pd
df = pd.read_csv('uber.csv')
```

In [412]:
```
df.head()
```

Out[412]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_lat
0	24238194	2015-05-07 19:52:06.0000003	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.73
1	27835199	2009-07-17 20:04:56.0000002	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.72
2	44984355	2009-08-24 21:45:00.00000061	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.74
3	25894730	2009-06-26 08:22:21.0000001	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.79
4	17610152	2014-08-28 17:47:00.000000188	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.74

In [413]:
```
df.columns
```

Out[413]:
```
Index(['Unnamed: 0', 'key', 'fare_amount', 'pickup_datetime',
       'pickup_longitude', 'pickup_latitude', 'dropoff_longitude',
       'dropoff_latitude', 'passenger_count'],
      dtype='object')
```

In [414]:
```
df.isnull().sum()
```

Out[414]:
```
Unnamed: 0           0
key                  0
fare_amount          0
pickup_datetime      0
pickup_longitude     0
pickup_latitude      0
dropoff_longitude    0
dropoff_latitude     0
passenger_count      1
dtype: int64
```

In [415]:
```
df.dropna(inplace=True)
```

In [416]:
```
df.drop('Unnamed: 0', inplace=True, axis='columns')
```

In [417]:
```
df.isnull().sum().sum()
```

Out[417]:
```
0
```

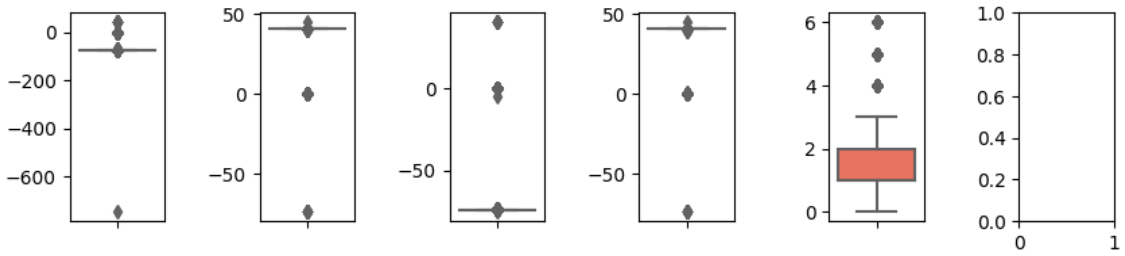
In [418]:
```
df.dtypes
```

Out[418]:
```
key                 object
fare_amount        float64
pickup_datetime     object
pickup_longitude   float64
pickup_latitude    float64
dropoff_longitude  float64
dropoff_latitude   float64
passenger_count    float64
dtype: object
```

2. Identify outliers.

In [419]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [420]:
```
fig, ax = plt.subplots(1,6, figsize=(10,2))
plt.subplots_adjust(wspace=1)
for i,j in enumerate(['pickup_longitude', 'pickup_latitude', 'dropoff_longi
  sns.boxplot(ax=ax[i], y=df[j].values, color='tomato')
plt.show();
```



In [421]:
```
df2 = df.drop(['key','pickup_datetime'], axis='columns')
q1,q3 = df2.quantile(0.25), df2.quantile(0.75)
iqr = q3 - q1
outliers = ((df2 < q1-1.5*iqr) | (df2 > q3+1.5*iqr))
outliers.sum()
```

Out[421]:
```
fare_amount          2166
pickup_longitude     1834
pickup_latitude      1234
dropoff_longitude    1893
dropoff_latitude     1615
passenger_count      2941
dtype: int64
```

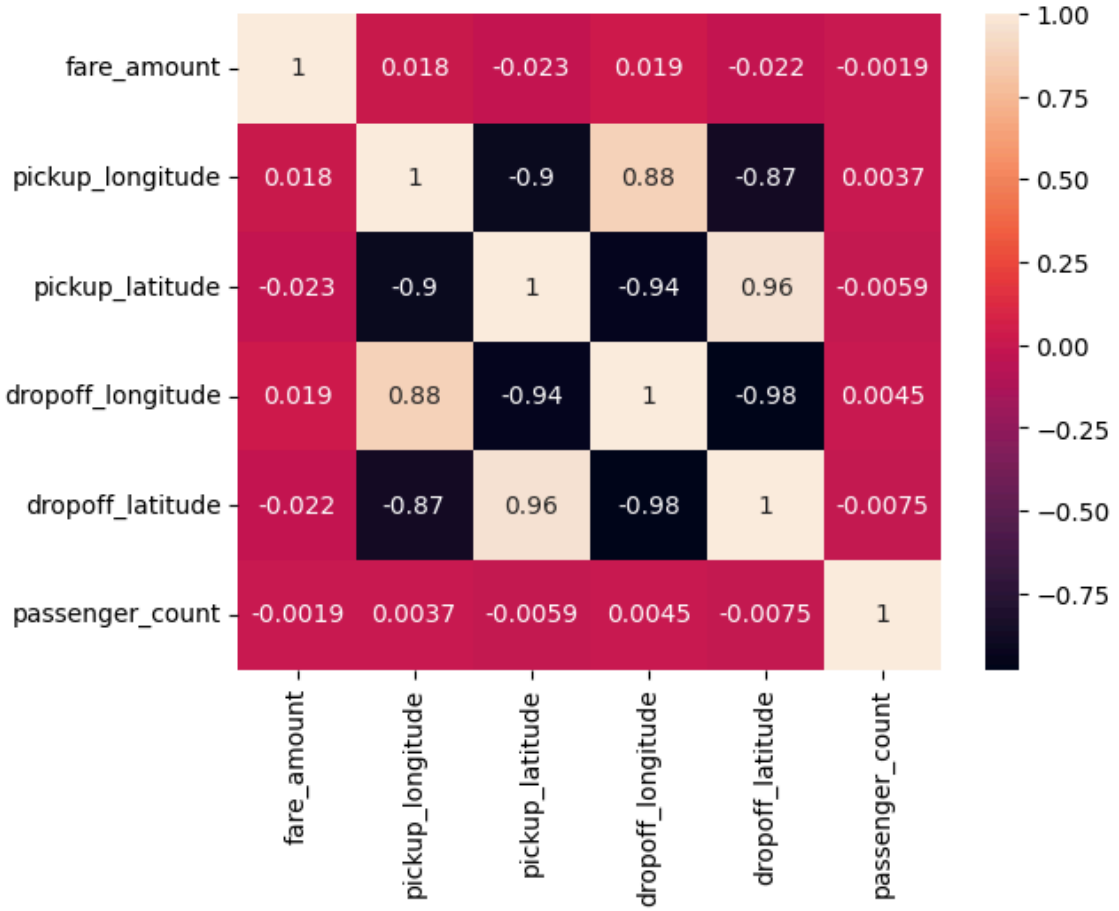
In [422]:
```
import numpy as np
df2.where(outliers, df2.mean(), inplace=True, axis=1)
```

In [423]:
```
df2['key'] = df.key
df2['pickup_datetime'] = df.pickup_datetime
df = df2
```

3. Check the correlation.

In [424]:
```
sns.heatmap(df.corr(), annot = True);
```

```
<ipython-input-424-ed4eb9922c2f>:1: FutureWarning: The default value of nu
meric_only in DataFrame.corr is deprecated. In a future version, it will d
efault to False. Select only valid columns or specify the value of numeric
_only to silence this warning.
  sns.heatmap(df.corr(), annot = True);
```



In [425]:
```
# Calculate distances - do not use the normal formula, use Haversine instea
# df['Distance'] = ( (df['pickup_latitude']-df['dropoff_latitude'])**2 + (d
```

Haversine Formula

Obtaining the distance between two points on Earth – distance between coordinates.

If we apply the haversine formula for a **central angle** (i.e., the angle between two points along a great circle on a sphere with radius **R**) and solve for distance, we obtain:

$$\mathbf{d = 2R \times \sin^{-1}(\sqrt{[\sin^2((\theta_2 - \theta_1)/2) + \cos\theta_1 \times \cos\theta_2 \times \sin^2((\varphi_2 - \varphi_1)/2)]})}$$

where:

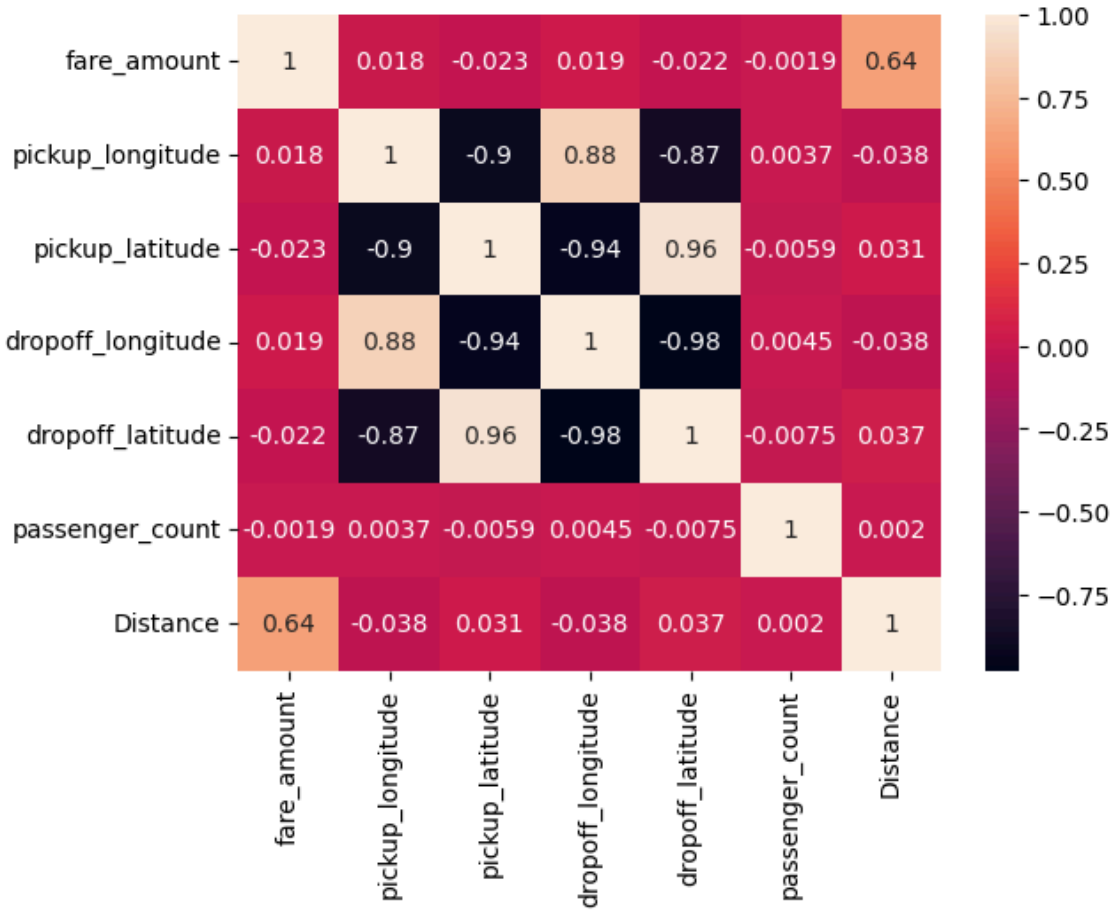
- θ_1, φ_1 – First point latitude and longitude coordinates;
- θ_2, φ_2 – Second point latitude and longitude coordinates;
- **R** – Earth's radius (**R = 6371 km**); and
- **d** – Distance between them along Earth's surface.

Keep in mind that the angles θ_1 and θ_2 are not the same as those used in the spherical coordinate system introduced above. They denote latitude in this formula.

In [426]:
```
import numpy as mp
R = 6371
df['Distance'] = 2 * R * np.arcsin( ( ( np.sin( (df['pickup_latitude']-df['
```

In [427]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.heatmap(df.corr(), annot=True);
```

```
<ipython-input-427-4bf01160af64>:3: FutureWarning: The default value of nu
meric_only in DataFrame.corr is deprecated. In a future version, it will d
efault to False. Select only valid columns or specify the value of numeric
_only to silence this warning.
  sns.heatmap(df.corr(), annot=True);
```



4. Implement linear regression and random forest regression models.

In [428]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

In [1473]:
```
train, test = train_test_split(df, test_size = 0.3, shuffle = True)
X_train, y_train = train.drop(['fare_amount','key','pickup_datetime','picku
X_test, y_test = test.drop(['fare_amount','key','pickup_datetime','pickup_l
```

In [1474]:
```
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [1475]:
```
lr = LinearRegression()
```

In [1476]:
```
lr.fit(X_train, y_train)
```

Out[1476]:
```
LinearRegression()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [1477]:
```
from sklearn.ensemble import RandomForestRegressor
rfr = RandomForestRegressor()
```

In [1478]:
```
rfr.fit(X_train, y_train)
```

Out[1478]:
```
RandomForestRegressor()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

5. Evaluate the models and compare their respective scores like R2, RMSE, etc.

In [1479]:
```
from sklearn.metrics import r2_score, mean_squared_error
```

In [1480]:
```
from sklearn.metrics import accuracy_score
```

In [1481]:
```
y_pred = lr.predict(X_test)
r2_score(y_test, y_pred)
```

Out[1481]:
```
0.45989286105013816
```

In [1482]:
```
mean_squared_error(y_test, y_pred) ** (1/2)
```

Out[1482]:
```
5.9432390670262745
```

In [1483]:
```
y_pred = rfr.predict(X_test)
r2_score(y_test, y_pred)
```

Out[1483]:
```
0.6917055451359275
```

In [1484]:
```
mean_squared_error(y_test, y_pred) ** (1/2)
```

Out[1484]:
```
4.4902042641254125
```