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

In [2]:
```
df = pd.read_csv("./datasets/uber.csv")
```

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

Out[3]:

| | Unnamed: 0 | key | fare_amount | pickup_datetime | pickup_longitude | pickup_latitude |
|---|---|---|---|---|---|---|
| **0** | 24238194 | 2015-05-07 19:52:06.0000003 | 7.5 | 2015-05-07 19:52:06 UTC | -73.999817 | 40.738354 |
| **1** | 27835199 | 2009-07-17 20:04:56.0000002 | 7.7 | 2009-07-17 20:04:56 UTC | -73.994355 | 40.728225 |
| **2** | 44984355 | 2009-08-24 21:45:00.00000061 | 12.9 | 2009-08-24 21:45:00 UTC | -74.005043 | 40.740770 |
| **3** | 25894730 | 2009-06-26 08:22:21.0000001 | 5.3 | 2009-06-26 08:22:21 UTC | -73.976124 | 40.790844 |
| **4** | 17610152 | 2014-08-28 17:47:00.000000188 | 16.0 | 2014-08-28 17:47:00 UTC | -73.925023 | 40.744085 |

In [4]:
```
df.columns
```

Out[4]:
```
Index(['Unnamed: 0', 'key', 'fare_amount', 'pickup_datetime',
       'pickup_longitude', 'pickup_latitude', 'dropoff_longitude',
       'dropoff_latitude', 'passenger_count'],
      dtype='object')
```

In [5]:
```
df = df.drop(['Unnamed: 0', 'key'], axis=1)
df.head()
```

Out[5]:

| | fare_amount | pickup_datetime | pickup_longitude | pickup_latitude | dropoff_longitude | dropoff_latit |
|---|---|---|---|---|---|---|
| **0** | 7.5 | 2015-05-07 19:52:06 UTC | -73.999817 | 40.738354 | -73.999512 | 40.723 |
| **1** | 7.7 | 2009-07-17 20:04:56 UTC | -73.994355 | 40.728225 | -73.994710 | 40.750 |
| **2** | 12.9 | 2009-08-24 21:45:00 UTC | -74.005043 | 40.740770 | -73.962565 | 40.772 |
| **3** | 5.3 | 2009-06-26 08:22:21 UTC | -73.976124 | 40.790844 | -73.965316 | 40.803 |
| **4** | 16.0 | 2014-08-28 17:47:00 UTC | -73.925023 | 40.744085 | -73.973082 | 40.761 |

In [6]:
```
df.isna().sum()
```

Out[6]:
```
fare_amount          0
pickup_datetime      0
pickup_longitude     0
pickup_latitude      0
dropoff_longitude    1
dropoff_latitude     1
passenger_count      0
dtype: int64
```

In [7]:
```
df = df.dropna(axis=0)
```

In [8]:
```
df.isna().sum()
```

Out[8]:
```
fare_amount          0
pickup_datetime      0
pickup_longitude     0
pickup_latitude      0
dropoff_longitude    0
dropoff_latitude     0
passenger_count      0
dtype: int64
```

In [9]:
```
df.shape
```

Out[9]:
```
(199999, 7)
```

In [10]:
```
df.dtypes
```

Out[10]:
```
fare_amount          float64
pickup_datetime       object
pickup_longitude     float64
pickup_latitude      float64
dropoff_longitude    float64
dropoff_latitude     float64
passenger_count        int64
dtype: object
```

In [11]:
```
df['pickup_datetime'] = pd.to_datetime(df['pickup_datetime'])
```

In [12]:
```
df.dtypes
```

Out[12]:
```
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 [13]:
```
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 [14]:
```
df = df.drop("pickup_datetime", axis=1)
```

In [15]:
```
df.shape
```

Out[15]: (199999, 11)

In [16]:
```
x = df.drop("fare_amount", axis=1)
y = df["fare_amount"]
```

In [17]:
```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_stat
```

# Linear Regression

In [18]:
```
model = LinearRegression()
```

In [19]:
```
model.fit(X_train, y_train)
```

Out[19]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)

In [20]:
```
y_pred = model.predict(X_test)
```

## MAE

In [21]:
```
mean_absolute_error(y_test, y_pred)
```

Out[21]: 5.988779835612282

## MSE

In [22]:
```
mean_squared_error(y_test, y_pred)
```

Out[22]: 96.73861203803375

## RMSE

In [23]:
```
np.sqrt(mean_squared_error(y_test, y_pred))
```

Out[23]: 9.83557888677803

# Random Forest

In [24]:
```
model = RandomForestRegressor(n_estimators=100)
```

In [25]:
```
model.fit(X_train, y_train)
```

Out[25]:
```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=None, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=100, n_jobs=None, oob_score=False,
                      random_state=None, verbose=0, warm_start=False)
```

In [26]:
```
y_pred = model.predict(X_test)
```

## MAE

In [27]:
```
mean_absolute_error(y_test, y_pred)
```

Out[27]:
```
1.9980047450595237
```

## MSE

In [28]:
```
mean_squared_error(y_test, y_pred)
```

Out[28]:
```
19.19960462828065
```

## RMSE

In [29]:
```
np.sqrt(mean_squared_error(y_test, y_pred))
```

Out[29]:
```
4.381735344390467
```