

```
In [1]: import numpy as np
import pandas as pd

In [2]: data = pd.read_csv("C:/Users/Admin/Desktop/Aditya K/uber.csv")

In [3]: data.head

Out[3]: <bound method NDFrame.head of          Unnamed: 0          key  fare
_amount  \
0      24238194    2015-05-07 19:52:06.0000003          7.5
1      27835199    2009-07-17 20:04:56.0000002          7.7
2      44984355    2009-08-24 21:45:00.00000061         12.9
3      25894730    2009-06-26 08:22:21.0000001          5.3
4      17610152    2014-08-28 17:47:00.000000188         16.0
...      ...      ...      ...      ...
199995  42598914    2012-10-28 10:49:00.00000053          3.0
199996  16382965    2014-03-14 01:09:00.0000008          7.5
199997  27804658    2009-06-29 00:42:00.00000078         30.9
199998  20259894    2015-05-20 14:56:25.0000004          14.5
199999  11951496    2010-05-15 04:08:00.00000076          14.1

          pickup_datetime  pickup_longitude  pickup_latitude  \
0      2015-05-07 19:52:06 UTC          -73.999817          40.738354
1      2009-07-17 20:04:56 UTC          -73.994355          40.728225
2      2009-08-24 21:45:00 UTC          -74.005043          40.740770
3      2009-06-26 08:22:21 UTC          -73.976124          40.790844
4      2014-08-28 17:47:00 UTC          -73.925023          40.744085
...      ...      ...      ...
199995  2012-10-28 10:49:00 UTC          -73.987042          40.739367
199996  2014-03-14 01:09:00 UTC          -73.984722          40.736837
199997  2009-06-29 00:42:00 UTC          -73.986017          40.756487
199998  2015-05-20 14:56:25 UTC          -73.997124          40.725452
199999  2010-05-15 04:08:00 UTC          -73.984395          40.720077

          dropoff_longitude  dropoff_latitude  passenger_count
0          -73.999512          40.723217              1
1          -73.994710          40.750325              1
2          -73.962565          40.772647              1
3          -73.965316          40.803349              3
4          -73.973082          40.761247              5
...      ...      ...      ...
199995          -73.986525          40.740297              1
199996          -74.006672          40.739620              1
199997          -73.858957          40.692588              2
199998          -73.983215          40.695415              1
199999          -73.985508          40.768793              1

[200000 rows x 9 columns]>

In [4]: data.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]: data["pickup_datetime"] = pd.to_datetime(data["pickup_datetime"])
```

```
In [6]: data.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  datetime64[ns, UTC]
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: datetime64[ns, UTC](1), float64(5), int64(2), object(1)
memory usage: 13.7+ MB
```

```
In [7]: #for finding missing values
data.isnull().sum()
```

```
Out[7]: Unnamed: 0            0
key                   0
fare_amount           0
pickup_datetime       0
pickup_longitude      0
pickup_latitude       0
dropoff_longitude     1
dropoff_latitude      1
passenger_count       0
dtype: int64
```

```
In [8]: # 0 means false & 1 means True
#if True means null or missing values in dataset or in row
#drop the row if it has missing values
data.dropna(inplace = True)
```

```
In [9]: data.isnull().sum()
```

```
Out[9]: 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  0
        dtype: int64
```

```
In [10]: # now create the machine Learning
```

```
In [11]: from sklearn.linear_model import LinearRegression
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import mean_squared_error
```

```
In [12]: # X IS PREDICTOR VARIABLE
        x = data.drop("fare_amount", axis = 1)

        #y is target variable
        y = data["fare_amount"]
```

```
In [13]: #to apply model
        x['pickup_datetime'] = pd.to_numeric(pd.to_datetime(x['pickup_datetime']))
        x = x.loc[:, x.columns.str.contains('^Unnamed')]
```

```
In [14]: x_train , x_test ,y_train ,y_test =train_test_split(x,y,test_size =0.2,)
        #testing dataset is 20%
        #traing dataset is 80% ,allocated to model
```

```
In [15]: # creating linear regression model
        lrmodel =LinearRegression()
        lrmodel.fit(x_train, y_train)
```

```
Out[15]: ▾ LinearRegression
        LinearRegression()
```

```
In [16]: #model is created
        pred = lrmodel.predict(x_test)
```

```
In [17]: #calculating RMSE root mean squared error
        lrmodelrmse = np.sqrt(mean_squared_error(pred ,y_test))
        print("RMSE error is : ",lrmodelrmse)

        RMSE error is : 9.678125467724962
```

```
In [18]: #Random forest Regression
        from sklearn.ensemble import RandomForestRegressor
        # create RFR
        rfrmodel = RandomForestRegressor(n_estimators = 100 , random_state = 101)
```

```
In [26]: # fit the forest
```

```
rfrmodel.fit(x_train , y_train)
rfrmodel_pred = rfrmodel.predict(x_test)
```

```
In [25]: #calculate RMSE for RFR
rfrmodel_rmse = np.sqrt(mean_squared_error(rfrmodel_pred ,y_test))
print("RFR RMSE error is : ",rfrmodel_rmse)
```

RFR RMSE error is : 11.83010330673784

```
In [24]: #pridiction
pred = lrmodel.predict(x_test)
print("hh",pred)
lrmodel.predict(x_test)
```

hh [11.37276195 11.35858736 11.35280229 ... 11.37269616 11.38204769  
11.36220594]

```
Out[24]: array([11.37276195, 11.35858736, 11.35280229, ..., 11.37269616,  
11.38204769, 11.36220594])
```

```
In [28]: from sklearn import metrics

#R2 Score
metrics.r2_score(y_test, rfrmodel_pred)
```

```
Out[28]: -0.49417200840639075
```

```
In [29]: from sklearn import metrics
#R2 Score
#R2 score Linear Regression
metrics.r2_score(y_test , pred)
```

```
Out[29]: -1.3856878633866643e-05
```

```
In [30]: #R2 score Linear Regression is : 52%
#R2 score RF Model is : 52%
#Random Forest Model Best fit for Ths dataset, is perfect
```

```
In [ ]:
```