# e-rides-using-regression-analysis

October 16, 2024

**To Predict the fare amount of future rides using regression analysis**

**Importing Important Libraries**

[3]:
```
import pandas as pd
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 r2_score
import warnings
warnings.filterwarnings("ignore")
import matplotlib.pyplot as plt
import numpy as np
```

**Reading Dataset using pandas function read_csv**

[5]:
```
z = pd.read_csv(r"C:\Users\skj_h\OneDrive\Desktop\uber.csv")
z
```

[5]:
```
        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]
```

[6]:
```
z.head()
```

[6]:
```
   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

           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

   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
```

[7]:
```
z.tail()
```

[7]:
```
        Unnamed: 0                            key  fare_amount  \
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  \
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
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
```

### Count number of null values present in dataset

[9]:
```
z.isnull().sum()
```

[9]:
```
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
```

### Dropping unnamed column from dataset

[11]:
```
z.drop(["Unnamed: 0"], axis = 1, inplace = True)
```

[12]:
```
for i in z:
    z = z[z[i].notna()]
z
```

[12]:
```
                                key  fare_amount          pickup_datetime  \
0         2015-05-07 19:52:06.0000003          7.5  2015-05-07 19:52:06 UTC
1         2009-07-17 20:04:56.0000002          7.7  2009-07-17 20:04:56 UTC
2        2009-08-24 21:45:00.00000061         12.9  2009-08-24 21:45:00 UTC
```

```
3          2009-06-26 08:22:21.0000001          5.3  2009-06-26 08:22:21 UTC
4        2014-08-28 17:47:00.000000188         16.0  2014-08-28 17:47:00 UTC
…                                  …          …                    …
199995    2012-10-28 10:49:00.00000053          3.0  2012-10-28 10:49:00 UTC
199996     2014-03-14 01:09:00.0000008          7.5  2014-03-14 01:09:00 UTC
199997    2009-06-29 00:42:00.00000078         30.9  2009-06-29 00:42:00 UTC
199998     2015-05-20 14:56:25.0000004         14.5  2015-05-20 14:56:25 UTC
199999    2010-05-15 04:08:00.00000076         14.1  2010-05-15 04:08:00 UTC

        pickup_longitude  pickup_latitude  dropoff_longitude  \
0             -73.999817        40.738354         -73.999512
1             -73.994355        40.728225         -73.994710
2             -74.005043        40.740770         -73.962565
3             -73.976124        40.790844         -73.965316
4             -73.925023        40.744085         -73.973082
…                    …                …                  …
199995        -73.987042        40.739367         -73.986525
199996        -73.984722        40.736837         -74.006672
199997        -73.986017        40.756487         -73.858957
199998        -73.997124        40.725452         -73.983215
199999        -73.984395        40.720077         -73.985508

        dropoff_latitude  passenger_count
0              40.723217                1
1              40.750325                1
2              40.772647                1
3              40.803349                3
4              40.761247                5
…                    …              …
199995         40.740297                1
199996         40.739620                1
199997         40.692588                2
199998         40.695415                1
199999         40.768793                1

[199999 rows x 8 columns]
```

### Shape of dataset

[14]:
```
z.shape
```

[14]:
```
(199999, 8)
```

### Size of dataset

[16]:
```
z.size
```

[16]:
```
1599992
```

[17]:
```
z.info()
```

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

### Number of Dimension of dataset

[19]:
```
z.ndim
```

[19]:
```
2
```

### Datatype of respective columns

[21]:
```
z.dtypes
```

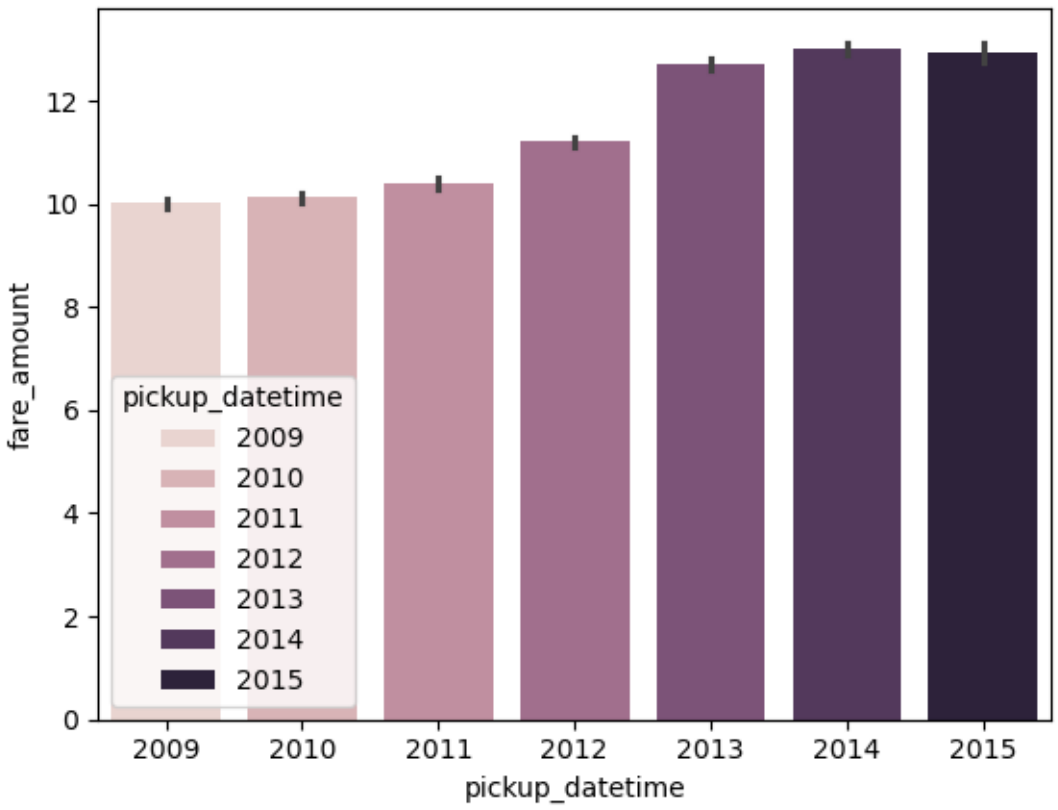
[21]:
```
key                 object
fare_amount        float64
pickup_datetime     object
pickup_longitude   float64
pickup_latitude    float64
dropoff_longitude  float64
dropoff_latitude   float64
passenger_count      int64
dtype: object
```

### Bivariate analysis

### Plotting barplot between pickup_datetime and fare_amount

[24]:
```
sns.barplot(x = pd.DatetimeIndex(z["pickup_datetime"]).year, y =␣
↪z["fare_amount"], data = z, hue = pd.DatetimeIndex(z["pickup_datetime"]).
↪year)
```

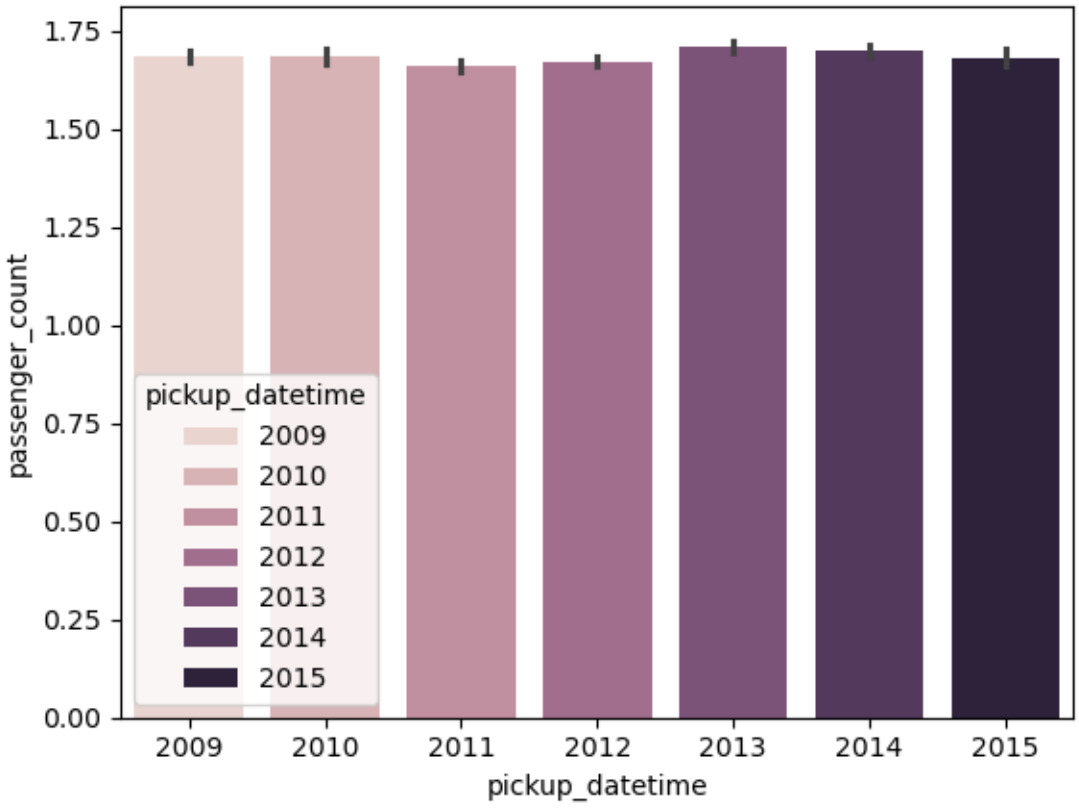
[24]:
```
<Axes: xlabel='pickup_datetime', ylabel='fare_amount'>
```




**Plotting barplot between pickup_datetime and passenger_count**

[26]:
```
sns.barplot(x = pd.DatetimeIndex(z["pickup_datetime"]).year, y =␣
↪z["passenger_count"], data = z, hue = pd.DatetimeIndex(z["pickup_datetime"]).
↪year)
```

[26]:
```
<Axes: xlabel='pickup_datetime', ylabel='passenger_count'>
```




### Correlation Coffiecient

[28]:
```
b = z.copy()
for i in b:
    if(b[i].dtype == "object"):
        b.drop([i], axis = 1, inplace = True)
b
```

[28]:

| | fare_amount | pickup_longitude | pickup_latitude | dropoff_longitude | \ |
|---|---|---|---|---|---|
| 0 | 7.5 | -73.999817 | 40.738354 | -73.999512 | |
| 1 | 7.7 | -73.994355 | 40.728225 | -73.994710 | |
| 2 | 12.9 | -74.005043 | 40.740770 | -73.962565 | |
| 3 | 5.3 | -73.976124 | 40.790844 | -73.965316 | |
| 4 | 16.0 | -73.925023 | 40.744085 | -73.973082 | |
| … | … | … | … | … | |
| 199995 | 3.0 | -73.987042 | 40.739367 | -73.986525 | |
| 199996 | 7.5 | -73.984722 | 40.736837 | -74.006672 | |
| 199997 | 30.9 | -73.986017 | 40.756487 | -73.858957 | |
| 199998 | 14.5 | -73.997124 | 40.725452 | -73.983215 | |
| 199999 | 14.1 | -73.984395 | 40.720077 | -73.985508 | |

```
        dropoff_latitude  passenger_count
0              40.723217                1
1              40.750325                1
2              40.772647                1
3              40.803349                3
4              40.761247                5
…                    …              …
199995         40.740297                1
199996         40.739620                1
199997         40.692588                2
199998         40.695415                1
199999         40.768793                1

[199999 rows x 6 columns]
```

[29]:
```
b.corr()
```

[29]:
```
                   fare_amount  pickup_longitude  pickup_latitude  \
fare_amount           1.000000          0.010458        -0.008482
pickup_longitude      0.010458          1.000000        -0.816461
pickup_latitude      -0.008482         -0.816461         1.000000
dropoff_longitude     0.008986          0.833026        -0.774787
dropoff_latitude     -0.011014         -0.846324         0.702367
passenger_count       0.010158         -0.000415        -0.001559

                   dropoff_longitude  dropoff_latitude  passenger_count
fare_amount                 0.008986         -0.011014         0.010158
pickup_longitude            0.833026         -0.846324        -0.000415
pickup_latitude            -0.774787          0.702367        -0.001559
dropoff_longitude           1.000000         -0.917010         0.000033
dropoff_latitude           -0.917010          1.000000        -0.000659
passenger_count             0.000033         -0.000659         1.000000
```

### Separating independent and dependent varaible from dataset

[31]:
```
X = b.copy()
X.drop(["fare_amount"], axis = 1, inplace = True)
Y = b["fare_amount"]
```

### Principle Component Analysis

[33]:
```
from sklearn.decomposition import PCA
```

[34]:
```
a = PCA()
x = a.fit_transform(X)
x.shape
```

[34]:
```
(199999, 5)
```

### Model selection

[36]:
```
x_train, x_test, y_train, y_test = train_test_split(x, Y, train_size = 0.7,␣
↪test_size = 0.3, random_state = 100)
```

[37]:
```
x_train.shape
```

[37]: (139999, 5)

[38]:
```
x_test.shape
```

[38]: (60000, 5)

[39]:
```
y_train = np.array(y_train).reshape(-1, 1)
y_test = np.array(y_test).reshape(-1, 1)
```

[40]:
```
y_train.shape
```

[40]: (139999, 1)

[41]:
```
y_test.shape
```

[41]: (60000, 1)

*Training model with training dataset*

[43]:
```
n = RandomForestRegressor()
n.fit(x_train, y_train)
```

[43]: RandomForestRegressor()

### Evaluating Training dataset

[45]:
```
y_predict_train = n.predict(x_train)
r2_train = r2_score(y_true = y_train, y_pred = y_predict_train)
```

[46]:
```
round(r2_train, 2)*100
```

[46]: 94.0

### Testing model with testing dataset

[48]:
```
n = RandomForestRegressor()
n.fit(x_test, y_test)
```

[48]: RandomForestRegressor()

### Evaluating Testing dataset

[50]:
```
y_predict_test = n.predict(x_test)
r2_test = r2_score(y_true = y_test, y_pred = y_predict_test)
```

[51]:
```
round(r2_test, 2)*100
```

[51]: 93.0

[ ]:
```

```