

Problem statement

Data collection

Importing libraries

```
In [1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Importing dataset

```
In [2]: data=pd.read_csv(r"C:\Users\user\Downloads\uber - uber.csv")
data
```

```
Out[2]:
```

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_latitude
0	24238194	2015-05-07 19:52:06	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.738354	-73.999817
1	27835199	2009-07-17 20:04:56	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.728225	-73.994355
2	44984355	2009-08-24 21:45:00	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.740770	-74.005043
3	25894730	2009-06-26 08:22:21	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.790844	-73.976124
4	17610152	2014-08-28 17:47:00	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.744085	-73.925023
...
199995	42598914	2012-10-28 10:49:00	3.0	2012-10-28 10:49:00 UTC	-73.987042	40.739367	-73.987042
199996	16382965	2014-03-14 01:09:00	7.5	2014-03-14 01:09:00 UTC	-73.984722	40.736837	-73.984722
199997	27804658	2009-06-29 00:42:00	30.9	2009-06-29 00:42:00 UTC	-73.986017	40.756487	-73.986017

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude
199998	20259894	2015-05-20 14:56:25	14.5	2015-05-20 14:56:25 UTC	-73.997124	40.725452	-73.997124
199999	11951496	2010-05-15 04:08:00	14.1	2010-05-15 04:08:00 UTC	-73.984395	40.720077	-73.984395

200000 rows × 9 columns

head

In [3]:

```
# to display first 8 dataset values
da=data.head(8)
da
```

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude
0	24238194	2015-05-07 19:52:06	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.738354	-73.999817
1	27835199	2009-07-17 20:04:56	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.728225	-73.994355
2	44984355	2009-08-24 21:45:00	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.740770	-73.962124
3	25894730	2009-06-26 08:22:21	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.790844	-73.965124
4	17610152	2014-08-28 17:47:00	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.744085	-73.973124
5	44470845	2011-02-12 02:27:09	4.9	2011-02-12 02:27:09 UTC	-73.969019	40.755910	-73.969019
6	48725865	2014-10-12 07:04:00	24.5	2014-10-12 07:04:00 UTC	-73.961447	40.693965	-73.871124
7	44195482	2012-12-11 13:52:00	2.5	2012-12-11 13:52:00 UTC	0.000000	0.000000	0.000000



info

```
In [4]: # to identify missing values
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
```

describe

```
In [5]: # to display summary of the dataset
data.describe()
```

```
Out[5]:
```

	Unnamed: 0	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude
count	2.000000e+05	200000.000000	200000.000000	200000.000000	199999.000000	199999.000000
mean	2.771250e+07	11.359955	-72.527638	39.935885	-72.525292	39.923800
std	1.601382e+07	9.901776	11.437787	7.720539	13.117408	6.794800
min	1.000000e+00	-52.000000	-1340.648410	-74.015515	-3356.666300	-881.985500
25%	1.382535e+07	6.000000	-73.992065	40.734796	-73.991407	40.733800
50%	2.774550e+07	8.500000	-73.981823	40.752592	-73.980093	40.753000
75%	4.155530e+07	12.500000	-73.967153	40.767158	-73.963659	40.768000
max	5.542357e+07	499.000000	57.418457	1644.421482	1153.572603	872.697600

columns

```
In [6]: # to display headings of the dataset
data.columns
```

```
Out[6]: Index(['Unnamed: 0', 'key', 'fare_amount', 'pickup_datetime',
               'pickup_longitude', 'pickup_latitude', 'dropoff_longitude',
               'dropoff_latitude', 'passenger_count'],
              dtype='object')
```

In [7]:

```
a=data.dropna(axis=1)
a
b=a.head(8)
b
```

Out[7]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	passenger_count
0	24238194	2015-05-07 19:52:06	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.738354	
1	27835199	2009-07-17 20:04:56	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.728225	
2	44984355	2009-08-24 21:45:00	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.740770	
3	25894730	2009-06-26 08:22:21	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.790844	
4	17610152	2014-08-28 17:47:00	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.744085	
5	44470845	2011-02-12 02:27:09	4.9	2011-02-12 02:27:09 UTC	-73.969019	40.755910	
6	48725865	2014-10-12 07:04:00	24.5	2014-10-12 07:04:00 UTC	-73.961447	40.693965	
7	44195482	2012-12-11 13:52:00	2.5	2012-12-11 13:52:00 UTC	0.000000	0.000000	

In [8]:

```
a.columns
```

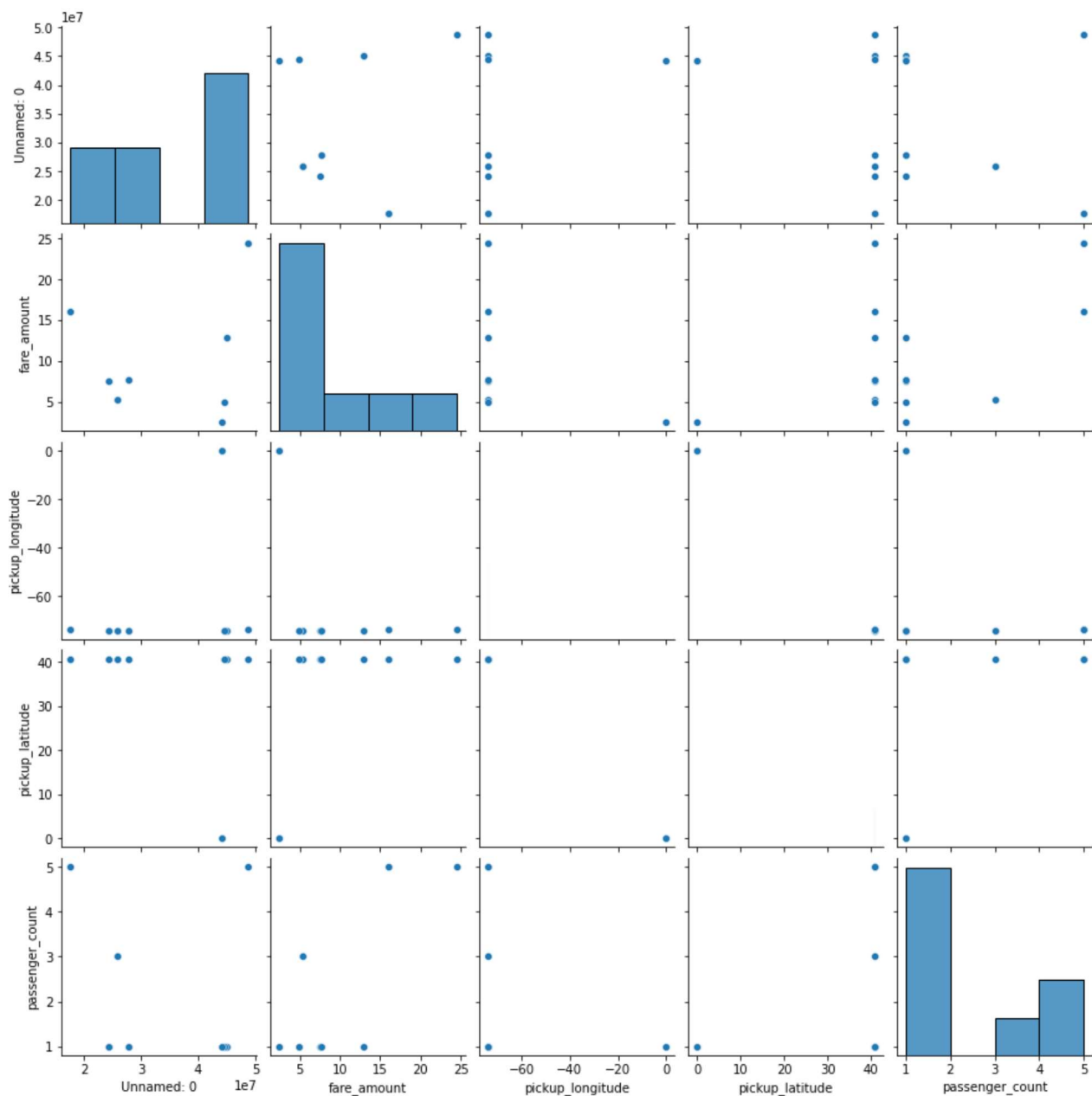
```
Out[8]: Index(['Unnamed: 0', 'key', 'fare_amount', 'pickup_datetime',
              'pickup_longitude', 'pickup_latitude', 'passenger_count'],
              dtype='object')
```

EDA and Visualization

In [9]:

```
sns.pairplot(b)
```

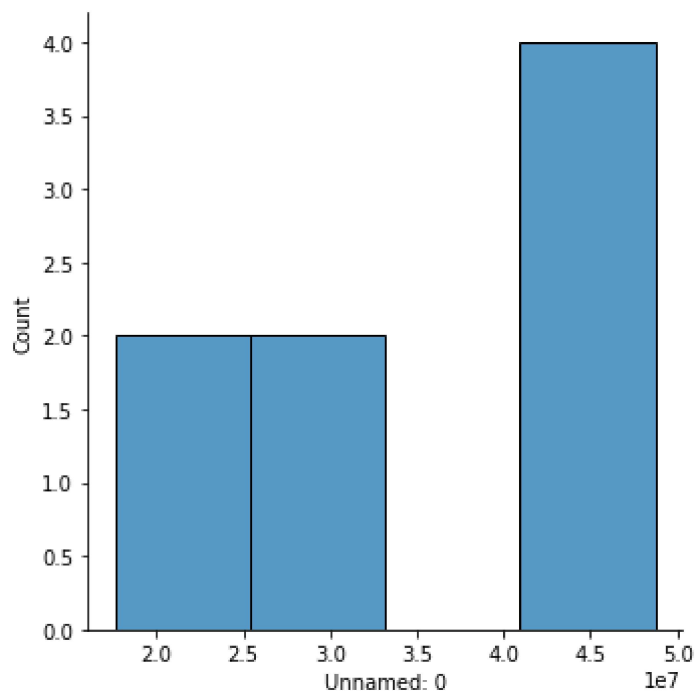
```
Out[9]: <seaborn.axisgrid.PairGrid at 0x2bda2900f70>
```



distribution plot

```
In [10]: sns.displot(b['Unnamed: 0'])
```

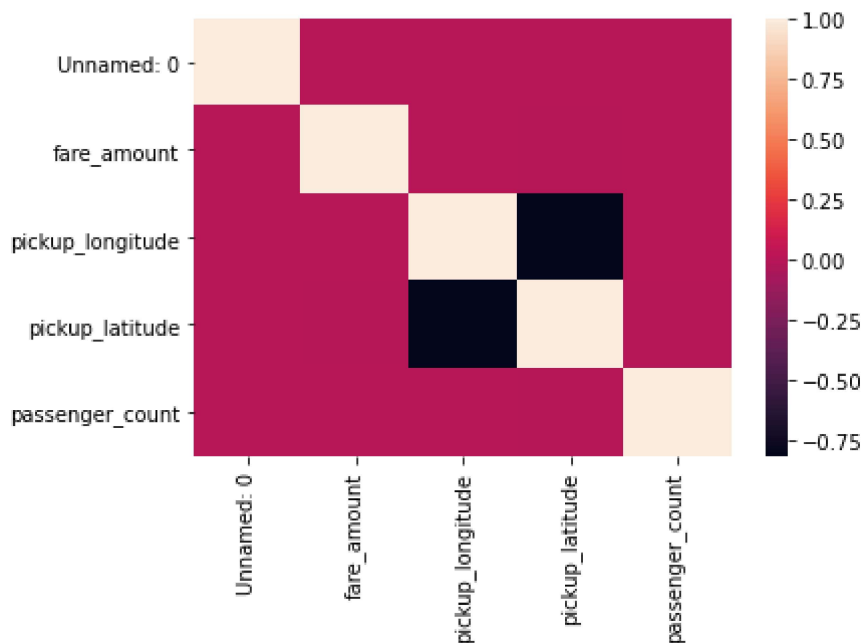
```
Out[10]: <seaborn.axisgrid.FacetGrid at 0x2bda4e302e0>
```



correlation

```
In [11]: dat=data[['Unnamed: 0', 'key', 'fare_amount', 'pickup_datetime',
                  'pickup_longitude', 'pickup_latitude', 'passenger_count']]
          sns.heatmap(dat.corr())
```

Out[11]: <AxesSubplot:>



To train the model-Model Building

```
In [12]: x=b[['passenger_count']]
         y=b['Unnamed: 0']
```

```
In [13]: # to split my dataset into training and test data
         from sklearn.model_selection import train_test_split
         x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

```
In [14]: from sklearn.linear_model import LinearRegression
         lr= LinearRegression()
         lr.fit(x_train,y_train)
```

Out[14]: LinearRegression()

```
In [15]: print(lr.intercept_)
```

42741640.46875001

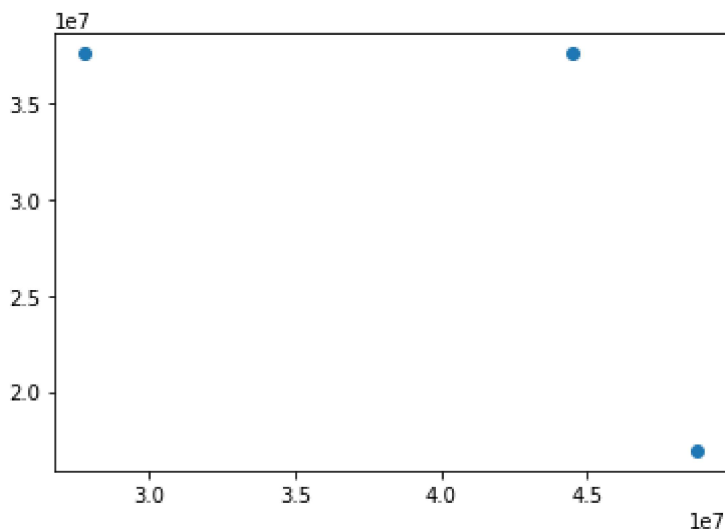
```
In [16]: coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
         coeff
```

Out[16]:

	Co-efficient
passenger_count	-5.162299e+06

```
In [17]: prediction=lr.predict(x_test)
         plt.scatter(y_test,prediction)
```

Out[17]: <matplotlib.collections.PathCollection at 0x2bdac61ccd0>



```
In [18]: print(lr.score(x_test,y_test))
```

-3.7318056708533227

```
In [20]: lr.score(x_test,y_test)
```

```
Out[20]: -3.7318056708533227
```

Ridge regression

```
In [23]: from sklearn.linear_model import Ridge,Lasso
```

```
In [24]: rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
```

```
Out[24]: -2.2397675501269685
```

```
In [25]: rr.score(x_train,y_train)
```

```
Out[25]: 0.4443310237946857
```

Lasso regression

```
In [26]: la=Lasso(alpha=10)
la.fit(x_train,y_train)
la.score(x_train,y_train)
```

```
Out[26]: 0.5501644421905896
```

```
In [27]: la.score(x_test,y_test)
```

```
Out[27]: -3.7318027077554774
```

```
In [ ]:
```