Problem Statement

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
d=pd.read_csv(r"C:\Users\user\Downloads\uber.csv")
d
```

Out[2]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	picku
0	24238194	2015-05-07 19:52:06.0000003	7.5	2015-05-07 19:52:06 UTC	-73.999817	
1	27835199	2009-07-17 20:04:56.0000002	7.7	2009-07-17 20:04:56 UTC	-73.994355	
2	44984355	2009-08-24 21:45:00.00000061	12.9	2009-08-24 21:45:00 UTC	-74.005043	
3	25894730	2009-06-26 08:22:21.0000001	5.3	2009-06-26 08:22:21 UTC	-73.976124	
4	17610152	2014-08-28 17:47:00.000000188	16.0	2014-08-28 17:47:00 UTC	-73.925023	
...	...	...	...	...	...	
199995	42598914	2012-10-28 10:49:00.00000053	3.0	2012-10-28 10:49:00 UTC	-73.987042	
199996	16382965	2014-03-14 01:09:00.0000008	7.5	2014-03-14 01:09:00 UTC	-73.984722	
199997	27804658	2009-06-29 00:42:00.00000078	30.9	2009-06-29 00:42:00 UTC	-73.986017	
199998	20259894	2015-05-20 14:56:25.0000004	14.5	2015-05-20 14:56:25 UTC	-73.997124	
199999	11951496	2010-05-15 04:08:00.00000076	14.1	2010-05-15 04:08:00 UTC	-73.984395	

200000 rows × 9 columns

In [3]:

```
d.columns
```

Out[3]:

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

In [4]:

```
d.fillna(value=1)
```

Out[4]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	picku
0	24238194	2015-05-07 19:52:06.0000003	7.5	2015-05-07 19:52:06 UTC	-73.999817	
1	27835199	2009-07-17 20:04:56.0000002	7.7	2009-07-17 20:04:56 UTC	-73.994355	
2	44984355	2009-08-24 21:45:00.00000061	12.9	2009-08-24 21:45:00 UTC	-74.005043	
3	25894730	2009-06-26 08:22:21.0000001	5.3	2009-06-26 08:22:21 UTC	-73.976124	
4	17610152	2014-08-28 17:47:00.000000188	16.0	2014-08-28 17:47:00 UTC	-73.925023	
...	...	...	...	...	...	
199995	42598914	2012-10-28 10:49:00.00000053	3.0	2012-10-28 10:49:00 UTC	-73.987042	
199996	16382965	2014-03-14 01:09:00.0000008	7.5	2014-03-14 01:09:00 UTC	-73.984722	
199997	27804658	2009-06-29 00:42:00.00000078	30.9	2009-06-29 00:42:00 UTC	-73.986017	
199998	20259894	2015-05-20 14:56:25.0000004	14.5	2015-05-20 14:56:25 UTC	-73.997124	
199999	11951496	2010-05-15 04:08:00.00000076	14.1	2010-05-15 04:08:00 UTC	-73.984395	

200000 rows × 9 columns

In [5]:

```
d.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 [6]:

```
x=d[['Unnamed: 0', 'fare_amount',
       'pickup_longitude', 'pickup_latitude']]
y=d['passenger_count']
```

In [7]:

```
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 [8]:

```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[8]:

```
LinearRegression()
```

In [9]:

```
print(lr.intercept_)
```

```
1.6782826635587302
```

In [10]:

```
print(lr.score(x_test,y_test))
```

```
3.061822822547633e-05
```

In [11]:

```
print(lr.score(x_train,y_train))
```

```
0.00014023174826049978
```

Ridge Regression

In [12]:

```
from sklearn.linear_model import Ridge,Lasso
```

In [13]:

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
```

Out[13]:

```
3.061819904248697e-05
```

In [14]:

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
```

Out[14]:

```
3.061819904248697e-05
```

Lasso Regression

In [15]:

```
la=Lasso(alpha=10)
```

In [16]:

```
la.fit(x_train,y_train)
```

Out[16]:

```
Lasso(alpha=10)
```

In [17]:

```
la.score(x_test,y_test)
```

Out[17]:

```
-2.4628875001653228e-05
```

Elastic Regression

In [18]:

```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[18]:

```
ElasticNet()
```

In [19]:

```
predict=(en.predict(x_test))
print(predict)
```

```
[1.68594447 1.67961547 1.68085247 ... 1.68064106 1.686006   1.68683586]
```

In [20]:

```
print(en.score(x_test,y_test))
```

```
-2.4628826478911847e-05
```

Evaluation Method

In [21]:

```
from sklearn import metrics
```

In [22]:

```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,predict))
```

```
Mean Absolute Error: 0.9626656974118661
```

In [23]:

```
print("Mean Square Error:",metrics.mean_squared_error(y_test,predict))
```

```
Mean Square Error: 2.419313769780454
```

In [24]:

```
print("Root Mean Square Error:",np.sqrt(metrics.mean_squared_error(y_test,predict)))
```

```
Root Mean Square Error: 1.5554143402259264
```

In [ ]: