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

In [2]:
```
df=pd.read_csv("uber.csv")[0:500]
df
```

Out[2]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dr
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	
...	...	...	...	...	...	...	...
495	1204312	2012-06-03 12:18:02.0000001	25.7	2012-06-03 12:18:02 UTC	-73.862765	40.770908	
496	2511529	2014-12-24 05:54:45.0000001	8.0	2014-12-24 05:54:45 UTC	-73.918530	40.743330	
497	24116460	2010-01-18 02:18:16.0000001	10.5	2010-01-18 02:18:16 UTC	-74.005734	40.743641	
498	42607669	2015-03-30 10:58:37.0000001	5.5	2015-03-30 10:58:37 UTC	-74.001648	40.740940	
499	36533403	2015-03-09 16:16:21.0000006	10.0	2015-03-09 16:16:21 UTC	-73.960037	40.780624	

500 rows × 9 columns

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

Out[3]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	drop
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	

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	drop
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	

DATA CLEANING AND DATA PREPROCESSING

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Unnamed: 0         500 non-null    int64  
 1   key                500 non-null    object 
 2   fare_amount        500 non-null    float64
 3   pickup_datetime    500 non-null    object 
 4   pickup_longitude   500 non-null    float64
 5   pickup_latitude    500 non-null    float64
 6   dropoff_longitude  500 non-null    float64
 7   dropoff_latitude   500 non-null    float64
 8   passenger_count    500 non-null    int64  
dtypes: float64(5), int64(2), object(2)
memory usage: 35.3+ KB
```

In [5]:
```
df.describe()
```

Out[5]:

	Unnamed: 0	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude
count	5.000000e+02	500.000000	500.000000	500.000000	500.000000	500.000000
mean	2.737940e+07	10.708720	-72.053865	39.692497	-72.201155	39.772818
std	1.607155e+07	8.334145	11.784239	6.491541	11.333432	6.243123
min	1.862090e+05	2.500000	-74.030417	0.000000	-74.027813	0.000000
25%	1.250293e+07	6.000000	-73.992804	40.735994	-73.991571	40.730869
50%	2.749836e+07	8.100000	-73.982352	40.752445	-73.980784	40.750428
75%	4.157492e+07	12.500000	-73.968724	40.765865	-73.965878	40.767497
max	5.519870e+07	57.330000	0.001782	40.850558	0.000875	40.901391

In [6]:
```
df.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]:
```
df1=df.dropna(axis=1)
df1
```

Out[7]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dr
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	
...	...	...	...	...	...	...	...
495	1204312	2012-06-03 12:18:02.0000001	25.7	2012-06-03 12:18:02 UTC	-73.862765	40.770908	
496	2511529	2014-12-24 05:54:45.0000001	8.0	2014-12-24 05:54:45 UTC	-73.918530	40.743330	
497	24116460	2010-01-18 02:18:16.0000001	10.5	2010-01-18 02:18:16 UTC	-74.005734	40.743641	
498	42607669	2015-03-30 10:58:37.0000001	5.5	2015-03-30 10:58:37 UTC	-74.001648	40.740940	
499	36533403	2015-03-09 16:16:21.0000006	10.0	2015-03-09 16:16:21 UTC	-73.960037	40.780624	

500 rows × 9 columns

In [8]:
```
df1.columns
```

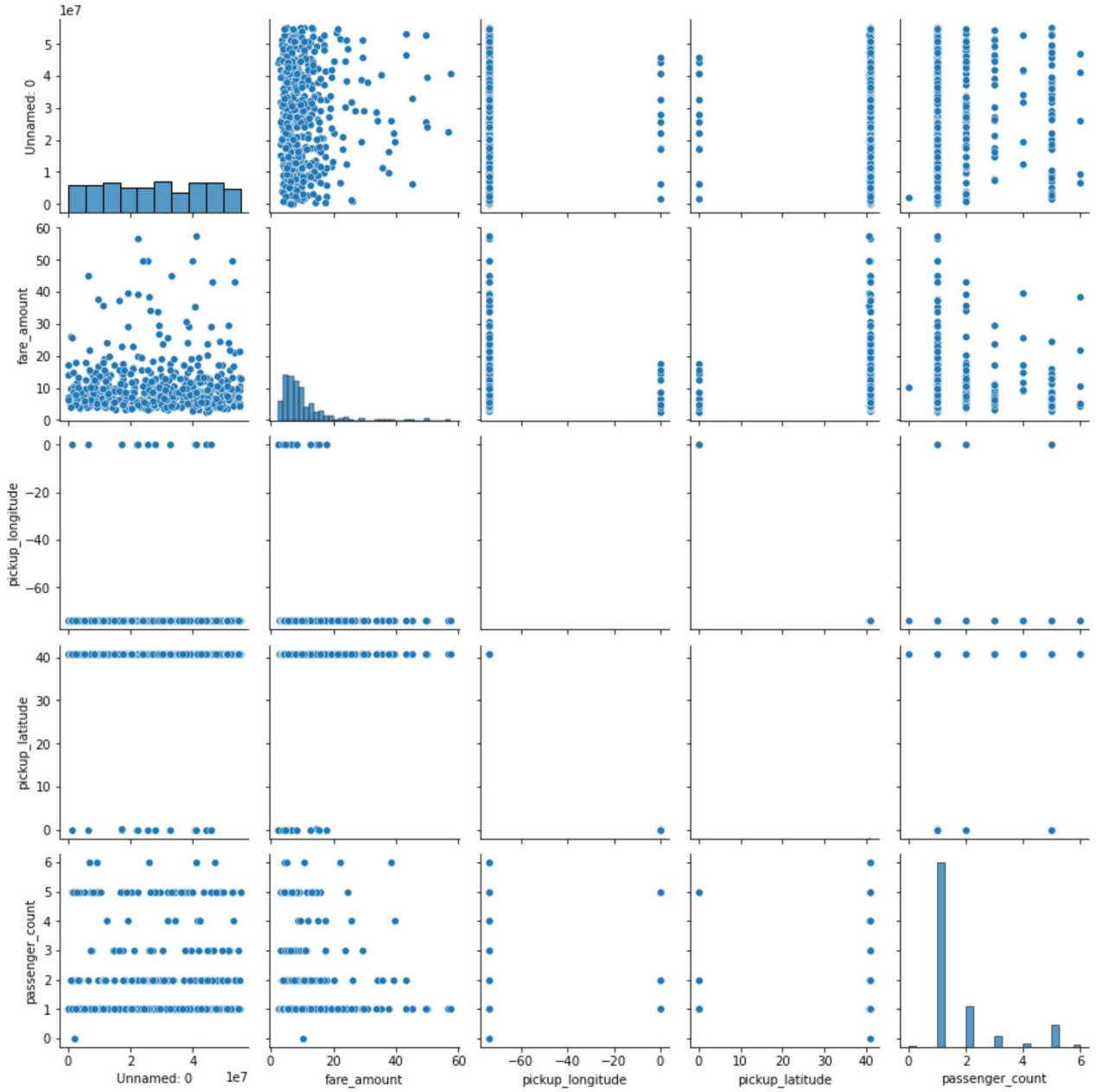
Out[8]:
```
Index(['Unnamed: 0', 'key', 'fare_amount', 'pickup_datetime',
       'pickup_longitude', 'pickup_latitude', 'dropoff_longitude',
       'dropoff_latitude', 'passenger_count'],
      dtype='object')
```

In [9]:
```
df1=df1[['Unnamed: 0', 'fare_amount',
       'pickup_longitude', 'pickup_latitude', 'passenger_count']]
```

EDA AND VISUALIZATION

In [10]:
```
sns.pairplot(df1)
```

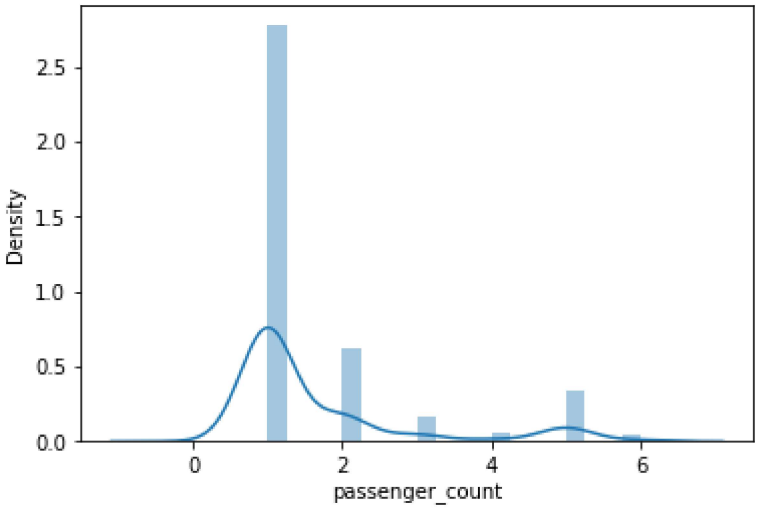
Out[10]: <seaborn.axisgrid.PairGrid at 0x226675c5c70>



In [11]:
```
sns.distplot(df1['passenger_count'])
```

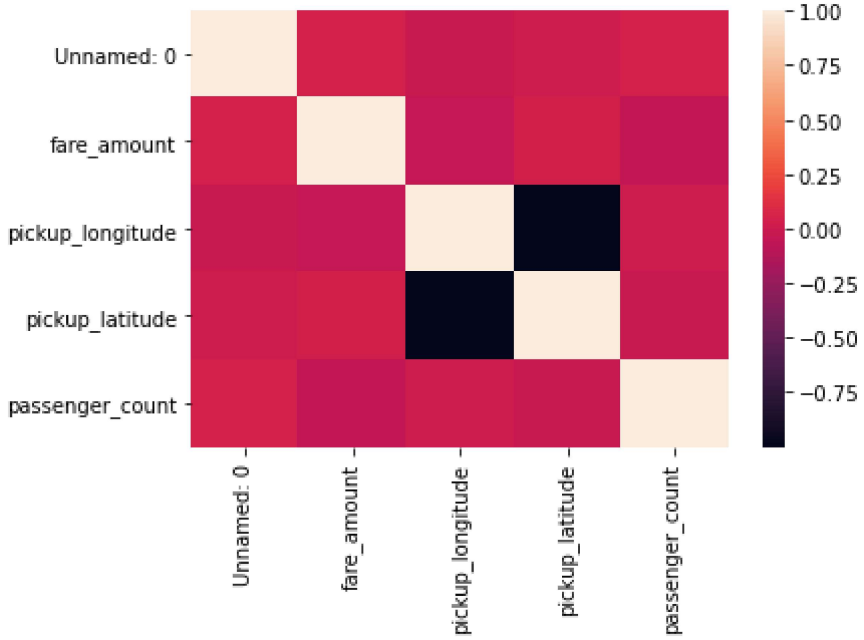
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[11]: <AxesSubplot:xlabel='passenger_count', ylabel='Density'>



In [12]:
```
sns.heatmap(df1.corr())
```

Out[12]: <AxesSubplot:>



TO TRAIN THE MODEL AND MODEL BULDING

In [13]:
```
x=df1[['Unnamed: 0', 'fare_amount',
       'pickup_longitude', 'pickup_latitude']]
y=df1['passenger_count']
```

In [14]:
```
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 [15]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

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

In [16]:
```
lr.intercept_
```

Out[16]:
```
1.4138476358059235
```

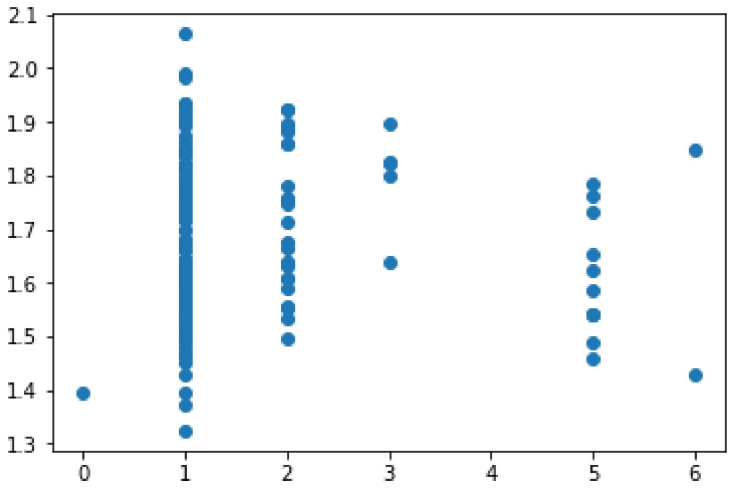
In [17]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[17]:

	Co-efficient
Unnamed: 0	7.636367e-09
fare_amount	-7.439512e-03
pickup_longitude	9.859199e-01
pickup_latitude	1.793545e+00

In [18]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[18]:
```
<matplotlib.collections.PathCollection at 0x22679debee0>
```



ACCURACY

In [19]:
```
lr.score(x_test,y_test)
```

Out[19]:
```
-0.03329128798279446
```

In [20]:
```
lr.score(x_train,y_train)
```

Out[20]:
```
0.012883954528980945
```

In [21]:
```
from sklearn.linear_model import Ridge,Lasso
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_ridge.py:147: LinAlgWar
ning: Ill-conditioned matrix (rcond=9.61016e-17): result may not be accurate.
  return linalg.solve(A, Xy, sym_pos=True,
```

Out[21]:
```
Ridge(alpha=10)
```

In [22]:
```
rr.score(x_train,y_train)
```

Out[22]:
```
0.011176396626520035
```

In [23]:
```
rr.score(x_test,y_test)
```

Out[23]:
```
-0.033054701138908626
```

In [24]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[24]:
```
Lasso(alpha=10)
```

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

Out[25]:
```
0.009237407439047773
```

In [26]:
```
la.score(x_test,y_test)
```

Out[26]:
```
-0.034144730018096814
```

In [27]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[27]:
```
ElasticNet()
```

In [28]:
```
en.coef_
```

Out[28]:
```
array([ 7.73004429e-09, -0.00000000e+00, -0.00000000e+00,  0.00000000e+00])
```

In [29]:
```
en.intercept_
```

Out[29]:
```
1.4860453516420387
```

In [30]:
```
prediction=en.predict(x_test)
```

In [31]:
```
en.score(x_test,y_test)
```

Out[31]:
```
-0.034144898124350176
```

In [32]:
```
from sklearn import metrics
print(metrics.mean_absolute_error(y_test,prediction))
print(metrics.mean_squared_error(y_test,prediction))
print(np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
0.8972005782520097
1.5204687722156278
1.2330728981757841
```