Type *Markdown* and LaTeX: α^2

In [ ]:

In [ ]:

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [2]:
```
df = pd.read_csv(r"E:\154\uber - uber.csv")[0:600].dropna(axis=1)
df
```

Out[2]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropo
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	
...	...	...	...	...	...	...	...
595	3268252	2012-06-12 11:41:16	6.1	2012-06-12 11:41:16 UTC	-73.952088	40.786637	
596	5992726	2011-09-20 22:04:00	9.7	2011-09-20 22:04:00 UTC	-73.956445	40.775568	
597	42806767	2011-09-07 14:15:00	14.9	2011-09-07 14:15:00 UTC	-74.009533	40.705928	
598	8308940	2011-02-17 04:27:00	6.9	2011-02-17 04:27:00 UTC	-74.005672	40.725620	
599	41718495	2011-05-29 22:07:00	7.7	2011-05-29 22:07:00 UTC	-73.956430	40.813242	

600 rows × 9 columns

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

Out[3]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_
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	-

Data cleaning and pre processing

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 600 entries, 0 to 599
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Unnamed: 0         600 non-null    int64  
 1   key                600 non-null    object 
 2   fare_amount        600 non-null    float64
 3   pickup_datetime    600 non-null    object 
 4   pickup_longitude   600 non-null    float64
 5   pickup_latitude    600 non-null    float64
 6   dropoff_longitude  600 non-null    float64
 7   dropoff_latitude   600 non-null    float64
 8   passenger_count    600 non-null    int64  
dtypes: float64(5), int64(2), object(2)
memory usage: 42.3+ KB
```

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

Out[5]:

	Unnamed: 0	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_l
count	6.000000e+02	600.000000	600.000000	600.000000	600.000000	600.(
mean	2.754724e+07	10.797317	-72.128589	39.733052	-72.249515	39.8
std	1.603314e+07	8.299398	11.559512	6.367668	11.176725	6.
min	1.862090e+05	2.500000	-74.030417	0.000000	-74.027813	0.(
25%	1.294860e+07	6.000000	-73.992810	40.735292	-73.991901	40.
50%	2.791547e+07	8.100000	-73.982352	40.752495	-73.980722	40.
75%	4.171866e+07	12.500000	-73.968882	40.766560	-73.965445	40.
max	5.519870e+07	57.330000	0.001782	40.850558	0.000875	40.9

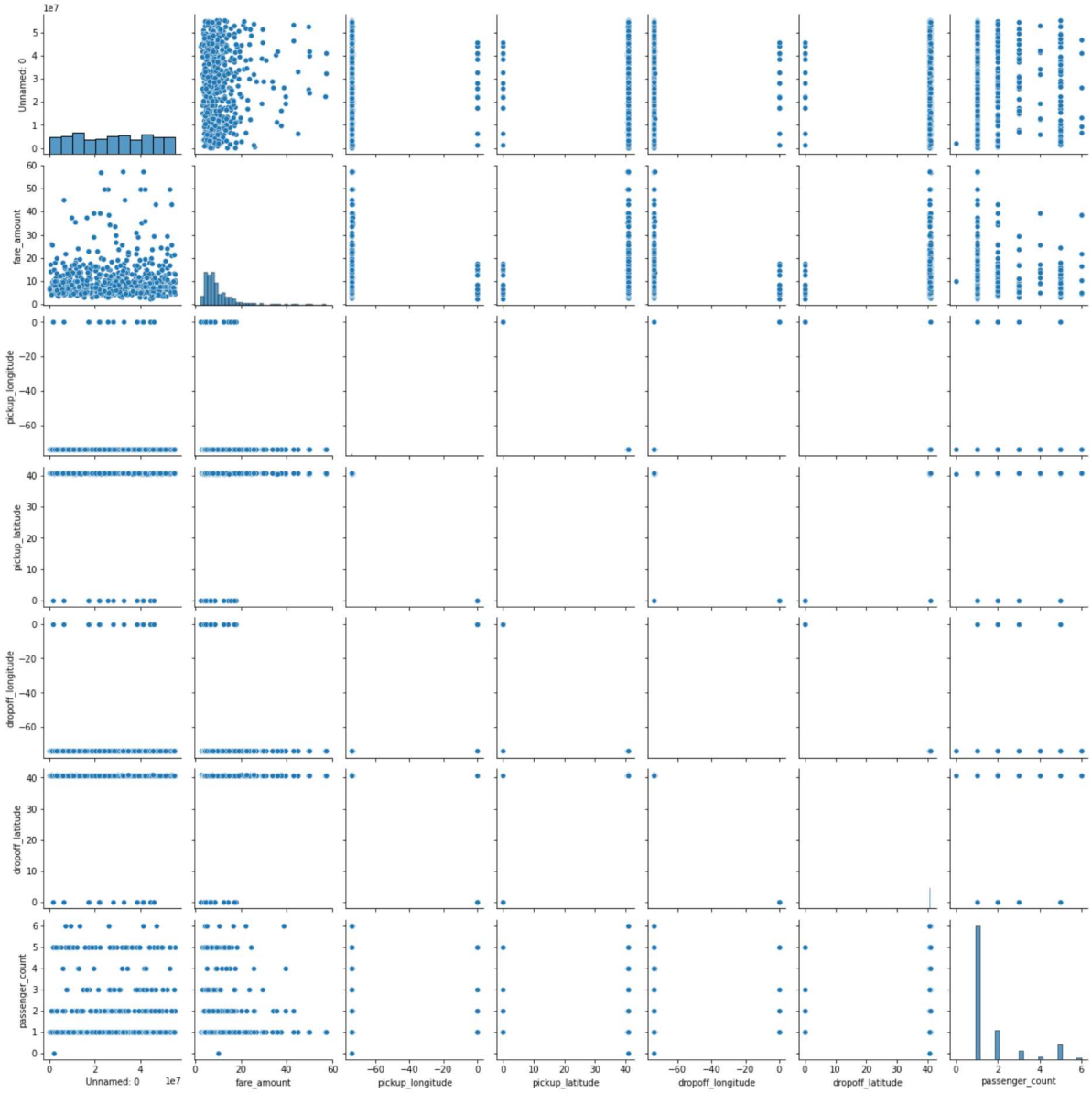
In [6]:
```
df.columns
```

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

EDA and VISUALIZATION

In [7]:
```
sns.pairplot(df)
```

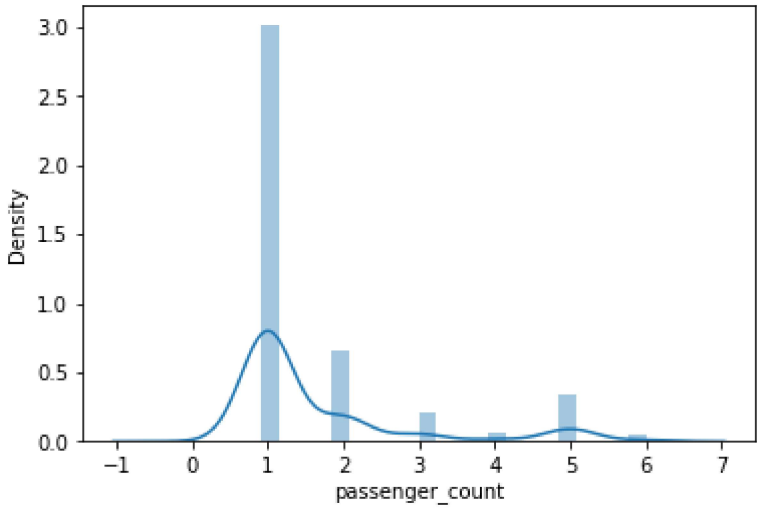
Out[7]:
```
<seaborn.axisgrid.PairGrid at 0x1f877777ee0>
```



In [8]:
```
sns.distplot(df["passenger_count"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function for hi
stograms).
  warnings.warn(msg, FutureWarning)
```

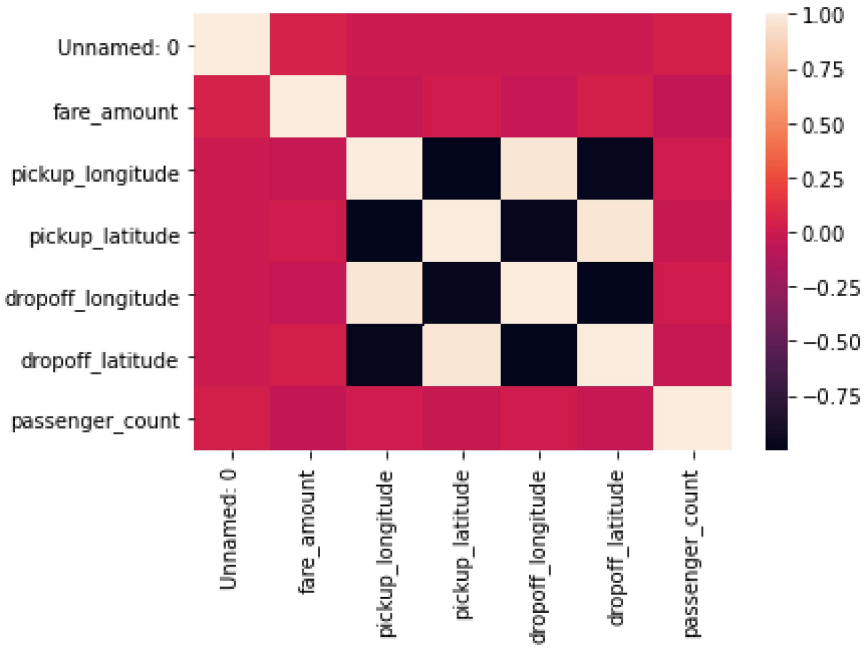
Out[8]: <AxesSubplot:xlabel='passenger_count', ylabel='Density'>



In [9]:
```
df1 = df[['Unnamed: 0', 'key', 'fare_amount', 'pickup_datetime',
       'pickup_longitude', 'pickup_latitude', 'dropoff_longitude',
       'dropoff_latitude', 'passenger_count']]
```

In [10]:
```
sns.heatmap(df1.corr())
```

Out[10]: <AxesSubplot:>



In [11]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 600 entries, 0 to 599
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Unnamed: 0         600 non-null    int64  
 1   key                600 non-null    object 
 2   fare_amount        600 non-null    float64
 3   pickup_datetime    600 non-null    object 
 4   pickup_longitude   600 non-null    float64
 5   pickup_latitude    600 non-null    float64
 6   dropoff_longitude  600 non-null    float64
 7   dropoff_latitude   600 non-null    float64
 8   passenger_count    600 non-null    int64  
dtypes: float64(5), int64(2), object(2)
memory usage: 42.3+ KB
```

In [12]:
```
x = df1[['Unnamed: 0',  'fare_amount',
       'pickup_longitude', 'pickup_latitude', 'dropoff_longitude',
       'dropoff_latitude']]
y = df1['passenger_count']
```

split the data into training and test data

In [13]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.3)
```

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

Out[14]:
```
LinearRegression()
```

In [15]:
```
lr.intercept_
```

Out[15]:
```
1.242288552748685
```

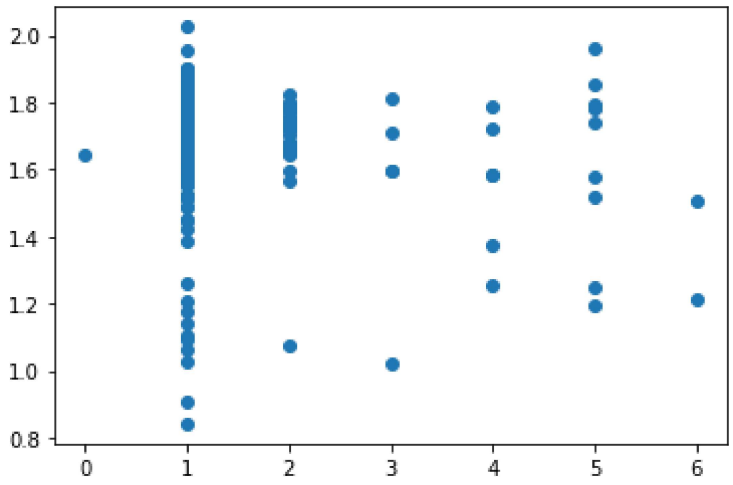
In [16]:
```
coeff = pd.DataFrame(lr.coef_, x.columns, columns =['Co-efficient'])
coeff
```

Out[16]:

	Co-efficient
Unnamed: 0	1.718375e-09
fare_amount	-1.711386e-02
pickup_longitude	-3.089407e-02
pickup_latitude	-3.964943e-02
dropoff_longitude	9.992542e-01
dropoff_latitude	1.812001e+00

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

Out[17]:
```
<matplotlib.collections.PathCollection at 0x1f8116902e0>
```



Accuracy

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

Out[18]:
```
-0.083471176183743
```

In [19]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [20]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_ridge.py:14
7: LinAlgWarning: Ill-conditioned matrix (rcond=7.79063e-17): result may not
be accurate.
  return linalg.solve(A, Xy, sym_pos=True,
```

Out[20]:
```
Ridge(alpha=10)
```

In [21]:
```
rr.score(x_test,y_test)
```

Out[21]:
```
-0.07980063820715766
```

In [22]:
```
la =Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[22]:
```
Lasso(alpha=10)
```

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

Out[23]:
```
-0.007126674282251777
```

ElasticNet

In [24]:
```
from sklearn.linear_model import ElasticNet
en = ElasticNet()
en.fit(x_train,y_train)
```

Out[24]:
```
ElasticNet()
```

In [25]:
```
print(en.coef_)
```

```
[ 1.66689312e-09 -7.70060098e-03 -3.01160917e-03  0.00000000e+00
 -0.00000000e+00  0.00000000e+00]
```

In [26]:
```
print(en.intercept_)
```

```
1.5057100773220573
```

In [27]:
```
print(en.predict(x_test))
```

```
[1.7309761  1.73200697 1.67226399 1.64946095 1.67933655 1.49616811
 1.75585849 1.77656601 1.74668197 1.55058765 1.67265543 1.75650089
 1.67671612 1.74984523 1.75711888 1.56977398 1.58374753 1.59971548
 1.7310774  1.73649513 1.74427076 1.75605133 1.77225792 1.73190041
 1.65087653 1.75352171 1.50517954 1.71982156 1.69622632 1.65440934
 1.74813824 1.62065024 1.71949503 1.67767588 1.32850304 1.41083419
 1.72704531 1.68211386 1.69007347 1.75440198 1.66849698 1.64828562
 1.73216021 1.59891644 1.6674226  1.73412977 1.6460221  1.74339996
 1.72226495 1.38925722 1.7061822  1.70313834 1.71957011 1.78086683
 1.35466119 1.74856575 1.65963671 1.6870316  1.65815282 1.7293351
 1.68270247 1.62039059 1.71438866 1.75262806 1.62796252 1.72383588
 1.70351799 1.70806272 1.67090544 1.72966347 1.77874313 1.70409979
 1.70863996 1.67887115 1.74264877 1.73113736 1.76397145 1.39263
 1.70562959 1.73289866 1.74645797 1.75736718 1.73753983 1.67359167
 1.68590988 1.7114849  1.71695168 1.61309095 1.69146582 1.68614686
 1.69774064 1.74381313 1.78131952 1.67228388 1.46677129 1.70227788
 1.70580646 1.74355766 1.70361277 1.47487628 1.75447008 1.46116098
 1.75882507 1.65439867 1.63980528 1.73135037 1.69970868 1.73589006
 1.69139027 1.65061075 1.7498767  1.76604699 1.71601225 1.65556302
 1.68785963 1.70475838 1.71136593 1.76027875 1.689937   1.6860203
 1.70391738 1.710936   1.67606032 1.66827715 1.43961884 1.70683632
 1.61601414 1.53225787 1.64456702 1.74914765 1.45088816 1.69781231
 1.66749475 1.68541255 1.75027661 1.65306678 1.6905488  1.45611847
 1.69996473 1.77718959 1.62336689 1.66335894 1.66618304 1.69534831
 1.69582585 1.76672465 1.7298047  1.76406311 1.68325625 1.69619746
 1.62029134 1.76809261 1.73836071 1.7042307  1.64483981 1.72835037
 1.68845485 1.68535484 1.74375285 1.62337025 1.7186122  1.69388238
 1.73583523 1.70232301 1.56475346 1.45589113 1.7062034  1.7348692
 1.66479467 1.72113484 1.71541556 1.77632976 1.74260591 1.70608512
 1.72810007 1.68026542 1.70723539 1.67809987 1.6567483  1.67336637]
```

In [28]:
```
print(en.score(x_test,y_test))
```

```
-0.03217152375179744
```

Evaluation Metrics

In [29]:
```
from sklearn import metrics
```

In [30]:
```
print("Mean Absolute Error",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error 0.9016710019966401
```

In [31]:
```
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error: 1.5172274918096484
```

In [32]:
```
print("Root Mean Absolute Error:",np.sqrt(metrics.mean_squared_error(y_test,pr
```

```
Root Mean Absolute Error: 1.2317578868469439
```