vpunczt6a

July 28, 2023

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

[2]:
```
df=pd.read_csv("/content/7_uber.csv")[0:500]
df
```

[2]:

	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	
..	…	…	…	
495	1204312	2012-06-03 12:18:02.0000001	25.7	
496	2511529	2014-12-24 05:54:45.0000001	8.0	
497	24116460	2010-01-18 02:18:16.0000001	10.5	
498	42607669	2015-03-30 10:58:37.0000001	5.5	
499	36533403	2015-03-09 16:16:21.0000006	10.0	

	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	
..	…	…	…	
495	2012-06-03 12:18:02 UTC	-73.862765	40.770908	
496	2014-12-24 05:54:45 UTC	-73.918530	40.743330	
497	2010-01-18 02:18:16 UTC	-74.005734	40.743641	
498	2015-03-30 10:58:37 UTC	-74.001648	40.740940	
499	2015-03-09 16:16:21 UTC	-73.960037	40.780624	

	dropoff_longitude	dropoff_latitude	passenger_count
0	-73.999512	40.723217	1.0
1	-73.994710	40.750325	1.0

```
2           -73.962565         40.772647              1.0
3           -73.965316         40.803349              3.0
4           -73.973082         40.761247              5.0
..                 …                 …              …
495         -73.989013         40.688776              1.0
496         -73.946696         40.749438              1.0
497         -74.006287         40.708330              2.0
498         -74.005730         40.750175              1.0
499         -73.971756         40.765934              1.0

[500 rows x 9 columns]
```

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

[3]:
```
   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              1.0
1         -73.994710         40.750325              1.0
2         -73.962565         40.772647              1.0
3         -73.965316         40.803349              3.0
4         -73.973082         40.761247              5.0
```

1 DATA CLEANING AND DATA PREPROCESSING

[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    float64
dtypes: float64(6), int64(1), object(2)
memory usage: 35.3+ KB
```

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

[5]:
```
         Unnamed: 0  fare_amount  pickup_longitude  pickup_latitude  \
count  5.000000e+02   500.000000        500.000000       500.000000
mean   2.737940e+07    10.708720        -72.053865        39.692497
std    1.607155e+07     8.334145         11.784239         6.491541
min    1.862090e+05     2.500000        -74.030417         0.000000
25%    1.250293e+07     6.000000        -73.992804        40.735994
50%    2.749836e+07     8.100000        -73.982352        40.752445
75%    4.157492e+07    12.500000        -73.968724        40.765865
max    5.519870e+07    57.330000          0.001782        40.850558

       dropoff_longitude  dropoff_latitude  passenger_count
count         500.000000        500.000000       500.000000
mean          -72.201155         39.772818         1.664000
std            11.333432          6.243123         1.267405
min           -74.027813          0.000000         0.000000
25%           -73.991571         40.730869         1.000000
50%           -73.980784         40.750428         1.000000
75%           -73.965878         40.767497         2.000000
max             0.000875         40.901391         6.000000
```

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

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

[7]:
```
df1=df.dropna(axis=1)
df1
```

[7]:
```
     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
```

```
..          …                              …          …
495     1204312    2012-06-03 12:18:02.0000001         25.7
496     2511529    2014-12-24 05:54:45.0000001          8.0
497    24116460    2010-01-18 02:18:16.0000001         10.5
498    42607669    2015-03-30 10:58:37.0000001          5.5
499    36533403    2015-03-09 16:16:21.0000006         10.0

              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
..                       …               …               …
495  2012-06-03 12:18:02 UTC        -73.862765        40.770908
496  2014-12-24 05:54:45 UTC        -73.918530        40.743330
497  2010-01-18 02:18:16 UTC        -74.005734        40.743641
498  2015-03-30 10:58:37 UTC        -74.001648        40.740940
499  2015-03-09 16:16:21 UTC        -73.960037        40.780624

     dropoff_longitude  dropoff_latitude  passenger_count
0           -73.999512         40.723217              1.0
1           -73.994710         40.750325              1.0
2           -73.962565         40.772647              1.0
3           -73.965316         40.803349              3.0
4           -73.973082         40.761247              5.0
..                 …                …              …
495         -73.989013         40.688776              1.0
496         -73.946696         40.749438              1.0
497         -74.006287         40.708330              2.0
498         -74.005730         40.750175              1.0
499         -73.971756         40.765934              1.0

[500 rows x 9 columns]
```

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

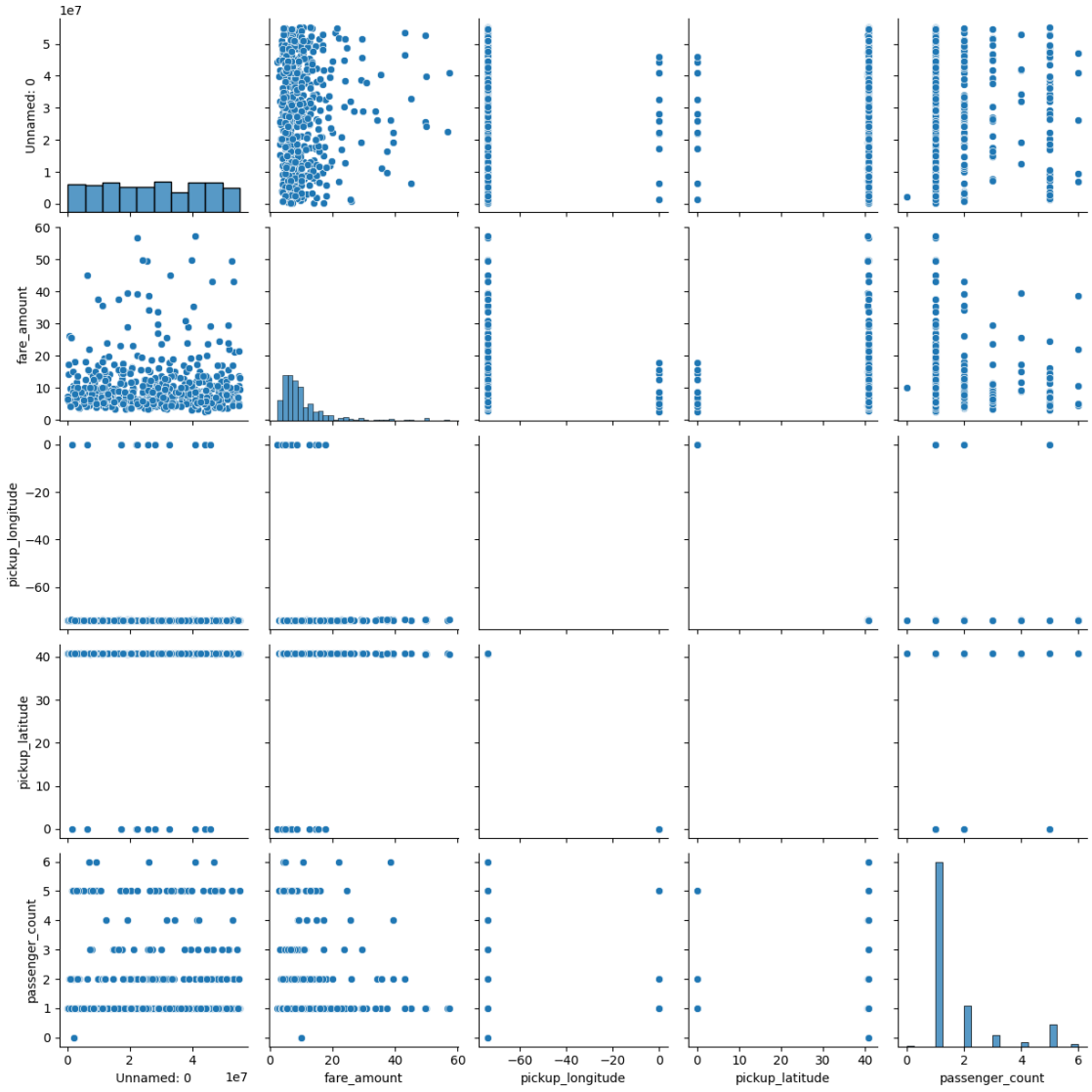
[8]:
```
Index(['Unnamed: 0', 'key', 'fare_amount', 'pickup_datetime',
       'pickup_longitude', 'pickup_latitude', 'dropoff_longitude',
       'dropoff_latitude', 'passenger_count'],
      dtype='object')
```

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

2 EDA AND VISUALIZATION

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

[10]:
```
<seaborn.axisgrid.PairGrid at 0x78a032cc6a70>
```

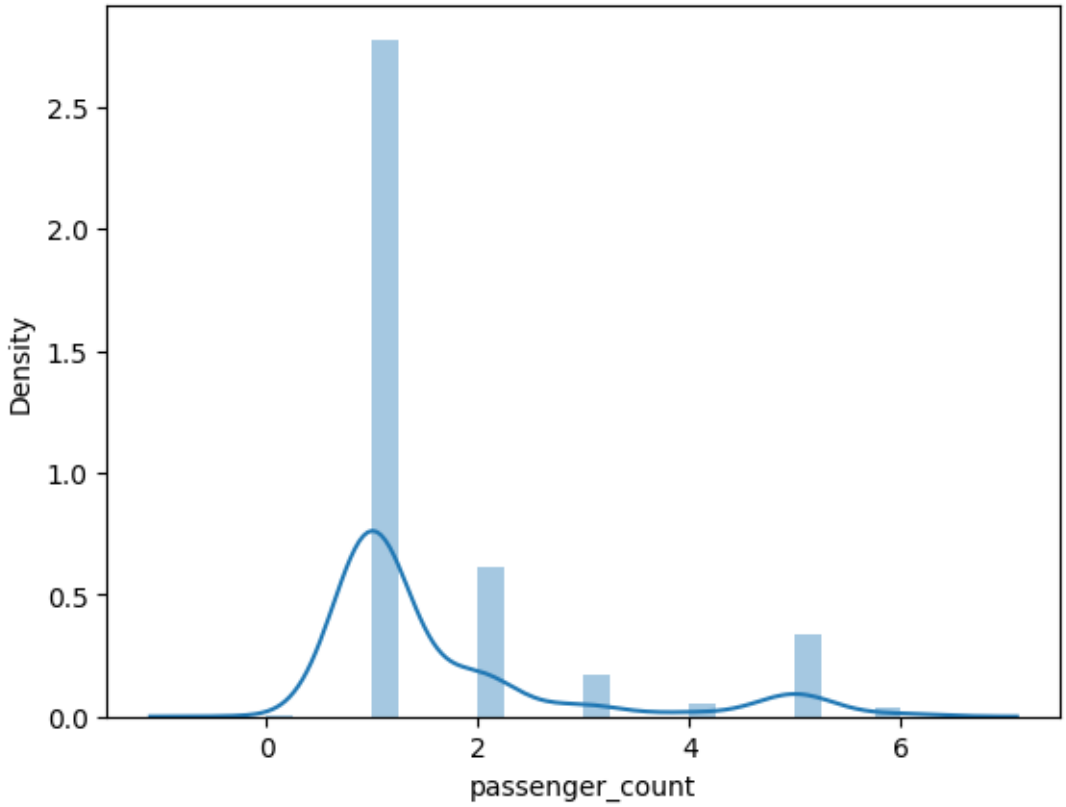


[11]:
```
sns.distplot(df1['passenger_count'])
```

```
<ipython-input-11-dd1ad478bc93>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
```

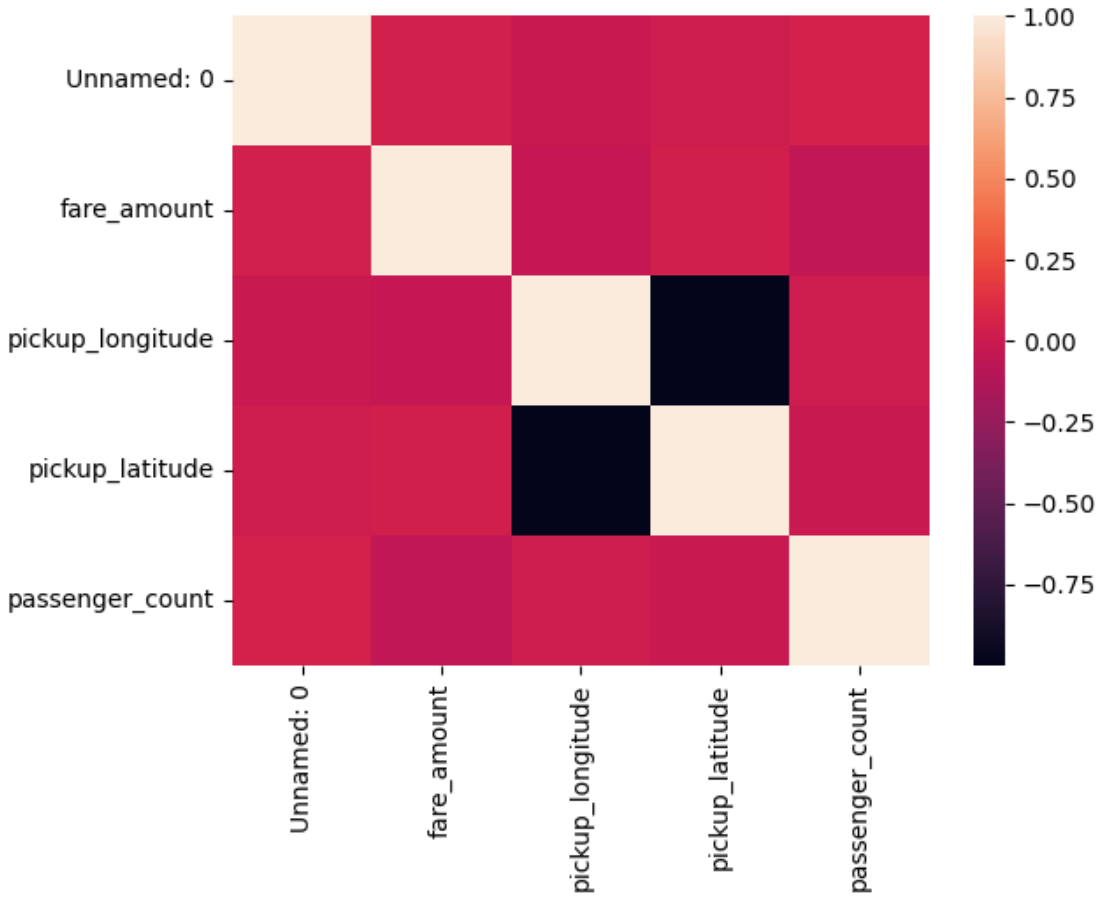
```
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df1['passenger_count'])
```

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



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

[12]: <Axes: >



3 TO TRAIN THE MODEL AND MODEL BULDING

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

[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)
```

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

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

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

[16]:
```
1.9313179041815287
```

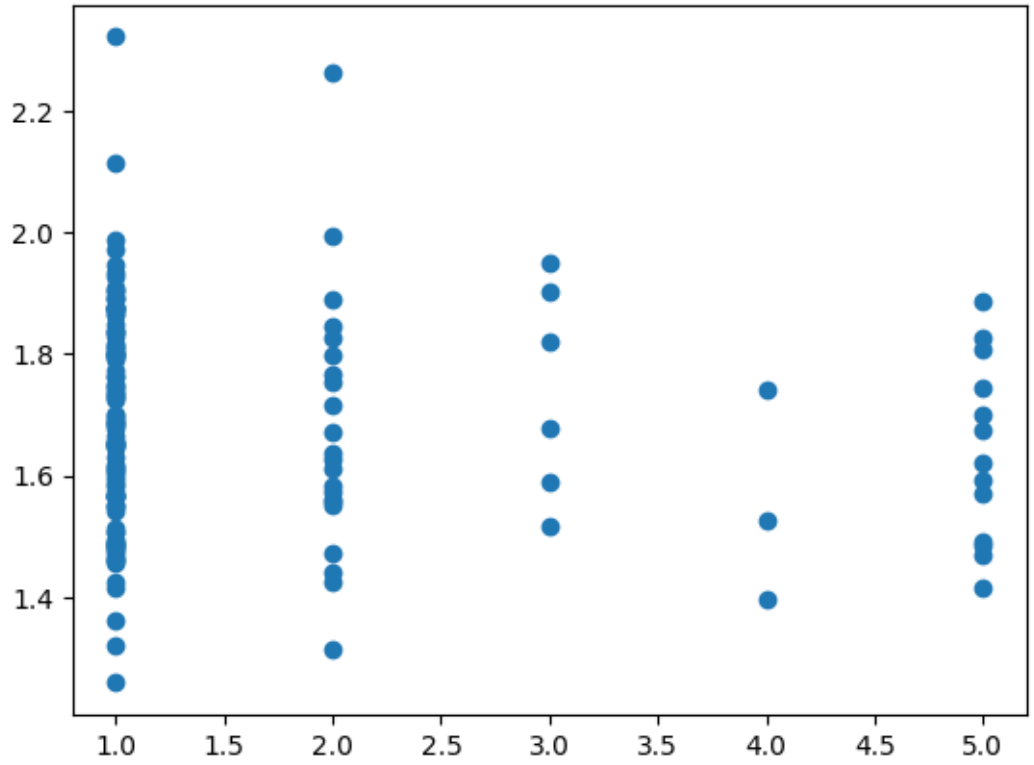
[17]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

[17]:

	Co-efficient
Unnamed: 0	9.190708e-09
fare_amount	-6.596362e-03
pickup_longitude	6.537954e-01
pickup_latitude	1.175916e+00

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

[18]: <matplotlib.collections.PathCollection at 0x78a0216fa4d0>



4 ACCURACY

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

[19]: -0.05046942195480253

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

[20]: 0.018052598493757954

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

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_ridge.py:216:
LinAlgWarning: Ill-conditioned matrix (rcond=9.70036e-17): result may not be
accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
```

[21]: Ridge(alpha=10)

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

[22]: 0.01721004477929977

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

[23]: -0.052562072025860385

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

[24]: Lasso(alpha=10)

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

[25]: 0.012883021447241183

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

[26]: -0.04334617663723028