KAMAU LYNN MWENDE

658884

APT3010A FALL SEMESTER 2021

DR. LAWRENCE NDERU

JUPYTER DATASET ASSIGNMENT

The dataset used is ***dccrdcoachpking.cvs*** Transport and Infrastructure Parking meters for Dublin City. 'Includes location, code, No of spaces per street(PD-Pay and Display D Disc Parking), exact locatioin, data install, tariff (cost per hour), nearest location of pay and display, clarway, if clearway conditions in operation(No parking or stopping during the hours indicated on the street sign), Coach Bay locations, further information, finished, x coordinate, y coordinate, tariff zone and Parking Voucher outlets and locations Spatial project. I edited the dataset to get rid of all 'nan' values and the new dataset ***newparking.cvs.***

The link to the data set is-> https://data.gov.ie/dataset/parking-meters-location-tariffs-and-zones-in-dublin-city

Reasons for using the dataset

The dataset has many numeric data

The data is easy to translate

It is easy to plot graphs with the data set

No nan values

Interpreting the data using python in Jupyter

a) Importing functions and extracting data from the dataset newparking.

jupyter Untitled4 Last Checkpoint: 18 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

```
In [1]: import matplotlib.pyplot as plt

In [2]: %matplotlib inline

In [3]: import pandas as pd

In [4]: parking = pd.read_csv('newparking.csv')
        print(parking.head)
```

```
21  235406.00
22  234488.00
23  235247.00
24  233715.00
25  234394.00
26  238755.00
27  233164.00
28  238810.00
29  238843.00
30  233733.00
31  238609.00
32  237666.00
33  234918.00
34  234860.00
35  233330.58
36  234338.96
37  234279.00
38  235094.00
39  234523.00  >
```

b) Displaying column names, changing 'Road_Markings' column name to 'Markings' and 'Time_Restrictions' to 'Restrictions'.

jupyter Untitled4 Last Checkpoint: 19 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

```
In [5]: print(parking.columns)

Index(['Road_Markings', 'X_end', 'Stat', 'ZONE', 'x_centre', 'y_centre',
       'Prohibition', 'X_start', 'Space', 'Time_Restriction', 'Coach_Parking',
       'Y_start', 'STREET', 'Operational_Hours', 'Y_end'],
      dtype='object')

In [6]: parking.columns=['Markings', 'X_end', 'Stat', 'ZONE', 'x_centre', 'y_centre', 'Prohibition', 'X_start', 'Space', 'Restriction',
        print(parking.head)
```

```
<bound method NDFrame.head of          Markings      X_end  Stat    ZONE  x_centre  y_centre  \
0    Loading Bay  315901.00     1  Yellow    315876    234506
1     Coach Bays  315255.00     1  Yellow    315222    233625
2     Coach Bays  316830.00     1  Yellow    316823    232852
3     Coach Bays  315392.00     1  Yellow    315380    233954
4     Coach Bays  315829.00     1  Yellow    315838    233762
5     Coach Bays  315043.00     1  Yellow    314977    234015
6     Coach Bays  314959.00     1  Yellow    314956    233721
7     Coach Bays  315742.00     1  Yellow    315741    232985
8     Coach Bays  315812.00     1  Yellow    315801    232921
9     Coach Bays  316329.00     1  Yellow    316331    234278
10    Coach Bays  316033.00     1  Yellow    316027    234779
11    Coach Bays  316607.83     1  Yellow    316608    233506
12    Coach Bays  316757.00     1  Yellow    316729    232833
13    Coach Bays  315867.00     0  Yellow    315860    234871
14    Coach Bays  315629.00     1  Yellow    315625    235149
15    Coach Bays  316817.00     0  Yellow    316812    234062
16    Coach Bays  316688.00     1  Yellow    316702    232463
17    Coach Bays  316593.00     1  Yellow    316605    232810
```

c) Perfoming integer division of the parking dataframe, since X_end values are constant.

jupyter Untitled4 Last Checkpoint: 20 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help | Trusted | Python 3

```
In [7]: parking.X_end = parking.X_end.floordiv(100)
        print(parking.head)
```

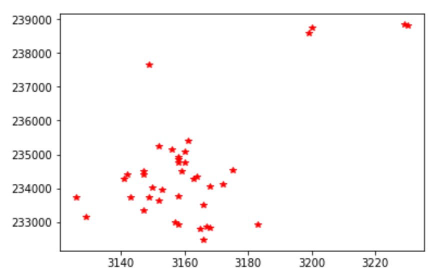
```
<bound method NDFrame.head of        Markings  X_end  Stat    ZONE  x_centre  y_centre  \
0   Loading Bay  3159.0     1  Yellow    315876    234506
1    Coach Bays  3152.0     1  Yellow    315222    233625
2    Coach Bays  3168.0     1  Yellow    316823    232852
3    Coach Bays  3153.0     1  Yellow    315380    233954
4    Coach Bays  3158.0     1  Yellow    315838    233762
5    Coach Bays  3150.0     1  Yellow    314977    234015
6    Coach Bays  3149.0     1  Yellow    314956    233721
7    Coach Bays  3157.0     1  Yellow    315741    232985
8    Coach Bays  3158.0     1  Yellow    315801    232921
9    Coach Bays  3163.0     1  Yellow    316331    234278
10   Coach Bays  3160.0     1  Yellow    316027    234779
11   Coach Bays  3166.0     1  Yellow    316608    233506
12   Coach Bays  3167.0     1  Yellow    316729    232833
13   Coach Bays  3158.0     0  Yellow    315860    234871
14   Coach Bays  3156.0     1  Yellow    315625    235149
15   Coach Bays  3168.0     0  Yellow    316812    234062
16   Coach Bays  3166.0     1  Yellow    316702    232463
17   Coach Bays  3165.0     1  Yellow    316605    232810
```

d) Plotting X_end and Y_end values as lines with X_end values on x-axis and Y_end values on the y-axis. Identifying the type of data on X_end values nad Y_end values

jupyter Untitled4 Last Checkpoint: 21 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help | Trusted | Python 3

```
In [8]: plt.plot(parking['X_end'],parking['Y_end'], 'r*')
Out[8]: [<matplotlib.lines.Line2D at 0x18e5e714c18>]
```




```
In [9]: type(parking['X_end'])
Out[9]: pandas.core.series.Series
```

```
In [10]: type(parking['Y_end'])
Out[10]: pandas.core.series.Series
```

e) Describing the parking dataframe. '.shape' property gets the current shape of an X_end and Y_end array.

jupyter Untitled4 Last Checkpoint: 25 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted | Python 3

```
In [11]: parking.describe()
Out[11]:
```

	X_end	Stat	x_centre	y_centre	X_start	Coach_Parking	Y_start	Y_end
count	40.000000	40.000000	40.00000	40.000000	40.000000	40.000000	40.000000	40.000000
mean	3162.125000	0.825000	316241.67500	234567.175000	316229.628250	0.950000	234570.026000	234562.923500
std	21.460981	0.384808	2150.36918	1702.744043	2155.806282	0.220721	1702.422842	1702.398039
min	3126.000000	0.000000	312622.00000	232463.000000	312602.000000	0.000000	232448.000000	232474.000000
25%	3149.750000	1.000000	314971.75000	233595.250000	314937.750000	1.000000	233596.100000	233595.350000
50%	3158.000000	1.000000	315861.00000	234279.000000	315852.000000	1.000000	234289.000000	234269.500000
75%	3166.250000	1.000000	316708.75000	234885.750000	316704.750000	1.000000	234896.500000	234874.500000
max	3230.000000	1.000000	323006.00000	238834.000000	323001.000000	1.000000	238828.000000	238843.000000

```
In [12]: parking['X_end'].shape
Out[12]: (40,)

In [13]: parking['X_end'].ndim
Out[13]: 1
```

f) Displaying X_end and Y_end values. Importing train_test_split from the sklearn.model_selection. Splitting the train_test_split dataframe, then reshapping the parking dataframe values of X_end and Y_end arrays with a random state of -11. Displaying X_test shape and X_test.shape.

jupyter Untitled4 Last Checkpoint: 26 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted | Python 3

```
In [14]: parking.X_end.values
Out[14]: array([3159., 3152., 3168., 3153., 3158., 3150., 3149., 3157., 3158.,
                3163., 3160., 3166., 3167., 3158., 3156., 3168., 3166., 3165.,
                3172., 3147., 3183., 3161., 3147., 3152., 3143., 3142., 3200.,
                3129., 3230., 3229., 3126., 3199., 3149., 3158., 3158., 3147.,
                3164., 3141., 3160., 3175.])

In [15]: parking.Y_end.values
Out[15]: array([234509.  , 233625.  , 232838.  , 233960.  , 233746.  , 234018.  ,
                233714.  , 232977.  , 232921.  , 234260.  , 234755.  , 233506.4 ,
                232848.  , 234753.  , 235153.  , 234050.  , 232474.  , 232802.  ,
                234123.  , 234403.  , 232908.  , 235406.  , 234488.  , 235247.  ,
                233715.  , 234394.  , 238755.  , 233164.  , 238810.  , 238843.  ,
                233733.  , 238609.  , 237666.  , 234918.  , 234860.  , 233330.58,
                234338.96, 234279.  , 235094.  , 234523.  ])

In [16]: from sklearn.model_selection import train_test_split

In [17]: X_train, X_test, y_train, y_test=train_test_split(parking.X_end.values.reshape(-1,1),parking.Y_end.values, random_state=11)

In [18]: X_test.shape
Out[18]: (10, 1)

In [19]: X_train.shape
Out[19]: (30, 1)
```

g) Importing LinearRegression from the sklearn.linear_model. Assigning liner_regression to the imported LinearRegression dataframe. Estimating the best representative function for the the data points. '.coef_' targets are the values to be predicted. intercept_ is the point where the function crosses the y-axis. Predicted values are the X_test and the expected values are y_test.

```
jupyter Untitled4 Last Checkpoint: 29 minutes ago (autosaved)                Logout
File Edit View Insert Cell Kernel Widgets Help                 Trusted | Python 3

In [20]: from sklearn.linear_model import LinearRegression

In [21]: liner_regression = LinearRegression()

In [22]: liner_regression.fit(X=X_train, y=y_train)

Out[22]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
                 normalize=False)

In [23]: liner_regression.coef_

Out[23]: array([52.40150991])

In [24]: liner_regression.intercept_

Out[24]: 68862.13888221621

In [25]: predicted = liner_regression.predict(X_test)

In [26]: expected = y_test

In [27]: #difference=[]
```

h) 'zip' prints both the predicted and expected value. Importing sns from seamodel and predicting the lamda of the liner_regression.coef_ (coefficient) and the liner_regression.intercept_. Displaying the predict.

```
jupyter Untitled4 Last Checkpoint: 29 minutes ago (autosaved)                Logout
File Edit View Insert Cell Kernel Widgets Help                 Trusted | Python 3

In [28]: for p,e in zip(predicted[::],expected[::]):
           #difference.append(p-e)
           print(f'predicted: {p:.2f}, expected {e:.2f}', )

         predicted: 236494.57, expected 238609.00
         predicted: 234031.70, expected 235247.00
         predicted: 233769.69, expected 234403.00
         predicted: 234346.11, expected 232921.00
         predicted: 234608.11, expected 234260.00
         predicted: 234503.31, expected 235406.00
         predicted: 233455.28, expected 234279.00
         predicted: 235656.14, expected 232908.00
         predicted: 234870.12, expected 234050.00
         predicted: 234660.52, expected 234338.96

In [29]: import seaborn as sns

In [30]: predict = (lambda x: liner_regression.coef_ * x + liner_regression.intercept_)

In [31]: predict(2022)

Out[31]: array([174817.99192194])
```

i) Plotting a scatter plot from the parking dataframe with x-axis being assigned X_end values and y-axis being assigned Y_end values. 'hue' parameter determines Y_end column in the parking data frame will be used for colour encoding. Palette uses winter column to set an interface, legend=false removes the legend. 'axes.set_ylim' set the y-limits of the current axes.

jupyter Untitled4 Last Checkpoint: 30 minutes ago (autosaved)

Logout

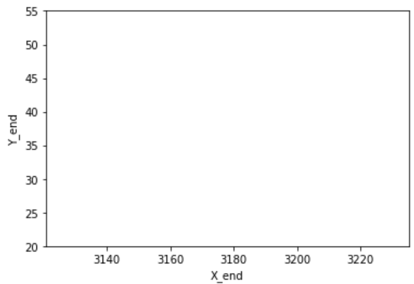
File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run | Code

In [33]:

```
axes= sns.scatterplot(data=parking,x='X_end',y='Y_end',hue='Y_end',palette='winter',legend=False)
axes.set_ylim(20,55)
import numpy as np
x=np.array([min(parking.X_end.values), max(parking.X_end.values)])
y=predict(x)
line = plt.plot(x,y)
```



In [ ]: