t7ncupo3s

July 31, 2023

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

[ ]:
```
df=pd.read_csv("/content/3_Fitness-1.csv")
df
```

[ ]:

	Row Labels	Sum of Jan	Sum of Feb	Sum of Mar	Sum of Total Sales
0	A	5.62%	7.73%	6.16%	75
1	B	4.21%	17.27%	19.21%	160
2	C	9.83%	11.60%	5.17%	101
3	D	2.81%	21.91%	7.88%	127
4	E	25.28%	10.57%	11.82%	179
5	F	8.15%	16.24%	18.47%	167
6	G	18.54%	8.76%	17.49%	171
7	H	25.56%	5.93%	13.79%	170
8	Grand Total	100.00%	100.00%	100.00%	1150

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

[ ]:

	Row Labels	Sum of Jan	Sum of Feb	Sum of Mar	Sum of Total Sales
0	A	5.62%	7.73%	6.16%	75
1	B	4.21%	17.27%	19.21%	160
2	C	9.83%	11.60%	5.17%	101
3	D	2.81%	21.91%	7.88%	127
4	E	25.28%	10.57%	11.82%	179

1 DATA CLEANING AND DATA PREPROCESSING

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9 entries, 0 to 8
Data columns (total 5 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
```

```
 0   Row Labels          9 non-null      object
 1   Sum of Jan          9 non-null      object
 2   Sum of Feb          9 non-null      object
 3   Sum of Mar          9 non-null      object
 4   Sum of Total Sales  9 non-null      int64
dtypes: int64(1), object(4)
memory usage: 488.0+ bytes
```

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

[ ]:
```
       Sum of Total Sales
count            9.000000
mean           255.555556
std            337.332963
min             75.000000
25%            127.000000
50%            167.000000
75%            171.000000
max           1150.000000
```

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

[ ]:
```
Index(['Row Labels', 'Sum of Jan', 'Sum of Feb', 'Sum of Mar',
       'Sum of Total Sales'],
      dtype='object')
```

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

[ ]:

	Row Labels	Sum of Jan	Sum of Feb	Sum of Mar	Sum of Total Sales
0	A	5.62%	7.73%	6.16%	75
1	B	4.21%	17.27%	19.21%	160
2	C	9.83%	11.60%	5.17%	101
3	D	2.81%	21.91%	7.88%	127
4	E	25.28%	10.57%	11.82%	179
5	F	8.15%	16.24%	18.47%	167
6	G	18.54%	8.76%	17.49%	171
7	H	25.56%	5.93%	13.79%	170
8	Grand Total	100.00%	100.00%	100.00%	1150

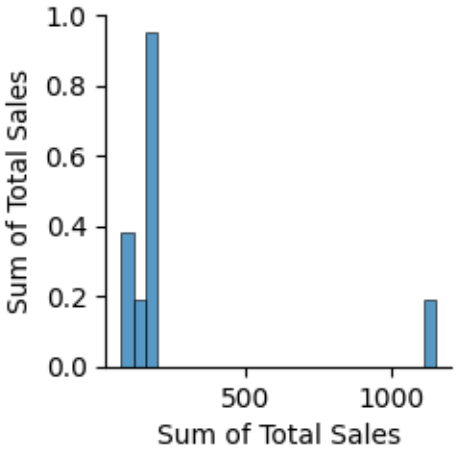
[ ]:
```
df1.columns
```

[ ]:
```
Index(['Row Labels', 'Sum of Jan', 'Sum of Feb', 'Sum of Mar',
       'Sum of Total Sales'],
      dtype='object')
```

2 EDA AND VISUALIZATION

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

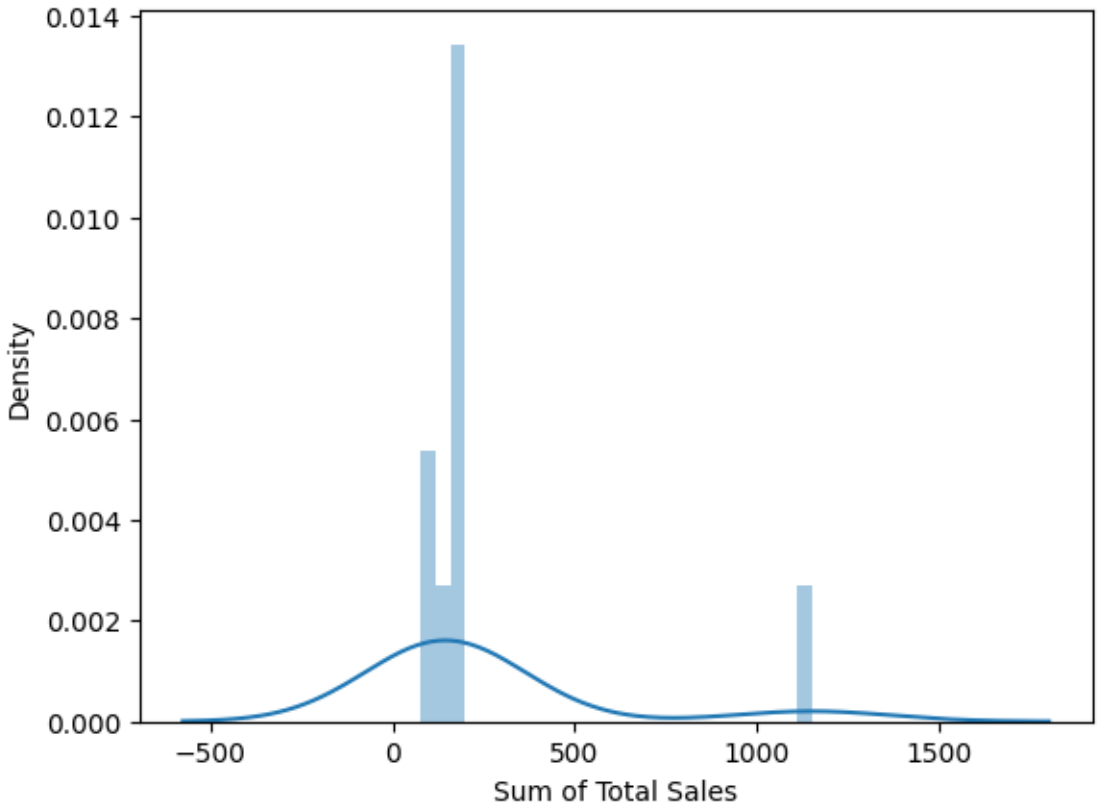
[ ]: <seaborn.axisgrid.PairGrid at 0x7cf7ccfca980>



[ ]:
```
sns.distplot(df1['Sum of Total Sales'])
```

```
<ipython-input-10-269cd82fce18>: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['Sum of Total Sales'])
```

[ ]: <Axes: xlabel='Sum of Total Sales', ylabel='Density'>



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

```
<ipython-input-11-3ed1a1a51dc0>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  sns.heatmap(df1.corr())
```

[ ]: <Axes: >

3 TO TRAIN THE MODEL AND MODEL BULDING

[ ]:
```
x=df[['Sum of Total Sales','Sum of Total Sales' ]]
y=df['Sum of Total Sales']
```

[ ]:
```
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)
```

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

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

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

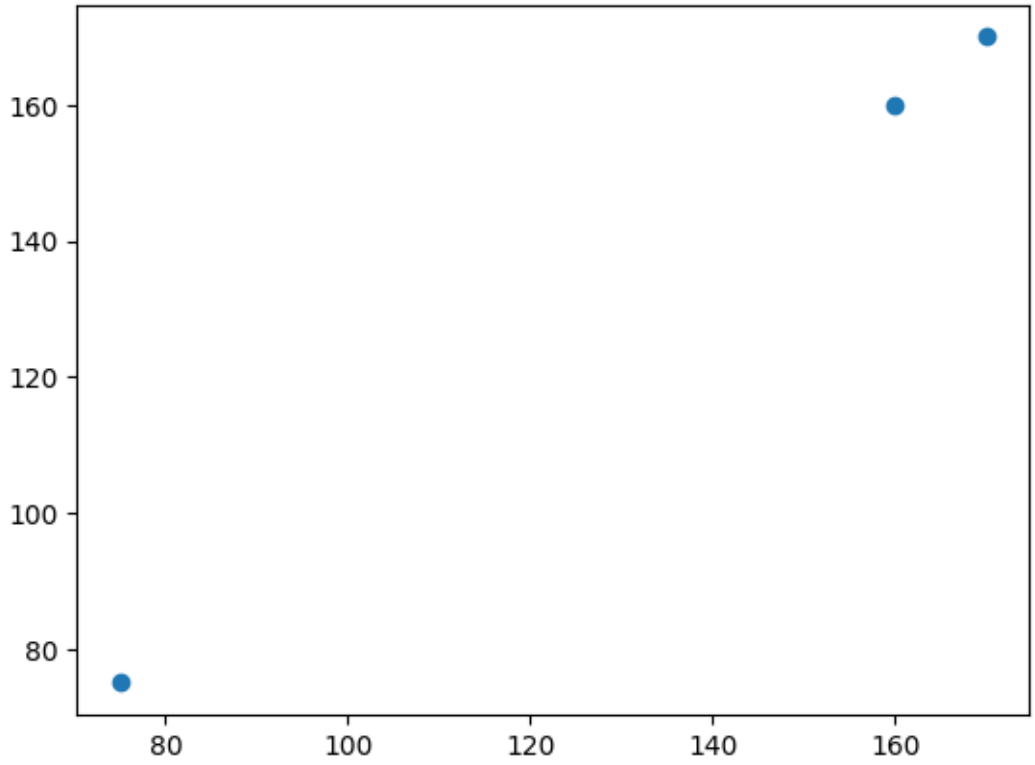
[ ]:
```
0.0
```

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

[ ]:
```
                    Co-efficient
Sum of Total Sales           0.5
Sum of Total Sales           0.5
```

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

[ ]:
```
<matplotlib.collections.PathCollection at 0x7cf7c7e1e290>
```



4 ACCURACY

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

[ ]:
```
1.0
```

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

[ ]:
```
1.0
```

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

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

[ ]: Ridge(alpha=10)

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

[ ]: 0.9999999993260014

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

[ ]: 0.9999999999645272

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

[ ]: Lasso(alpha=10)

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

[ ]: 0.9999999948918413

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

[ ]: 0.9999999029430332

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

[ ]: ElasticNet()

```
[ ]: print(en.coef_)
print(en.intercept_)
```

```
[9.99989279e-01 7.14703617e-06]
0.0011286603286180252
```

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

[ ]: array([170.00052115, 75.00086064, 160.00055689])

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

[ ]: 0.9999999997573541

[ ]:
```
from sklearn import metrics
print("Mean Absolute Error: ", metrics.mean_absolute_error(y_test,prediction))
print("Mean Squared Error: ", metrics.mean_squared_error(y_test,prediction))
print("Root Mean Squared Error: ", np.sqrt(metrics.
  ↪mean_squared_error(y_test,prediction)))
```

```
Mean Absolute Error:  0.0006462250430179969
Mean Squared Error:  4.40806691657591e-07
Root Mean Squared Error:  0.0006639327463362468
```