LinearRegression

In [4]:

```
import numpy as np
import pandas as pd
```

data collection

In [5]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as pp
import seaborn as sb
```

In [7]:

```
df = pd.read_csv(r"C:\Users\user\Desktop\fitness.csv")
df
```

Out[7]:

	Row Labels	Sum of Jan	Sum of Feb	Sum of Mar	Sum of Total Sales
0	A	0.06	0.08	0.06	75
1	B	0.04	0.17	0.19	160
2	C	0.10	0.12	0.05	101
3	D	0.03	0.22	0.08	127
4	E	0.25	0.11	0.12	179
5	F	0.08	0.16	0.18	167
6	G	0.19	0.09	0.17	171
7	H	0.26	0.06	0.14	170
8	Grand Total	1.00	1.00	1.00	1150

first 10 rows

In [8]:

```
df.head(10)
```

Out[8]:

	Row Labels	Sum of Jan	Sum of Feb	Sum of Mar	Sum of Total Sales
0	A	0.06	0.08	0.06	75
1	B	0.04	0.17	0.19	160
2	C	0.10	0.12	0.05	101
3	D	0.03	0.22	0.08	127
4	E	0.25	0.11	0.12	179
5	F	0.08	0.16	0.18	167
6	G	0.19	0.09	0.17	171
7	H	0.26	0.06	0.14	170
8	Grand Total	1.00	1.00	1.00	1150

data cleaning

In [9]:

```
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      float64
 2   Sum of Feb          9 non-null      float64
 3   Sum of Mar          9 non-null      float64
 4   Sum of Total Sales  9 non-null      int64  
dtypes: float64(3), int64(1), object(1)
memory usage: 488.0+ bytes
```

In [10]:

```
df.describe()
```

Out[10]:

	Sum of Jan	Sum of Feb	Sum of Mar	Sum of Total Sales
count	9.000000	9.000000	9.000000	9.000000
mean	0.223333	0.223333	0.221111	255.555556
std	0.304097	0.295508	0.296625	337.332963
min	0.030000	0.060000	0.050000	75.000000
25%	0.060000	0.090000	0.080000	127.000000
50%	0.100000	0.120000	0.140000	167.000000
75%	0.250000	0.170000	0.180000	171.000000
max	1.000000	1.000000	1.000000	1150.000000

In [11]:

```
df.columns
```

Out[11]:

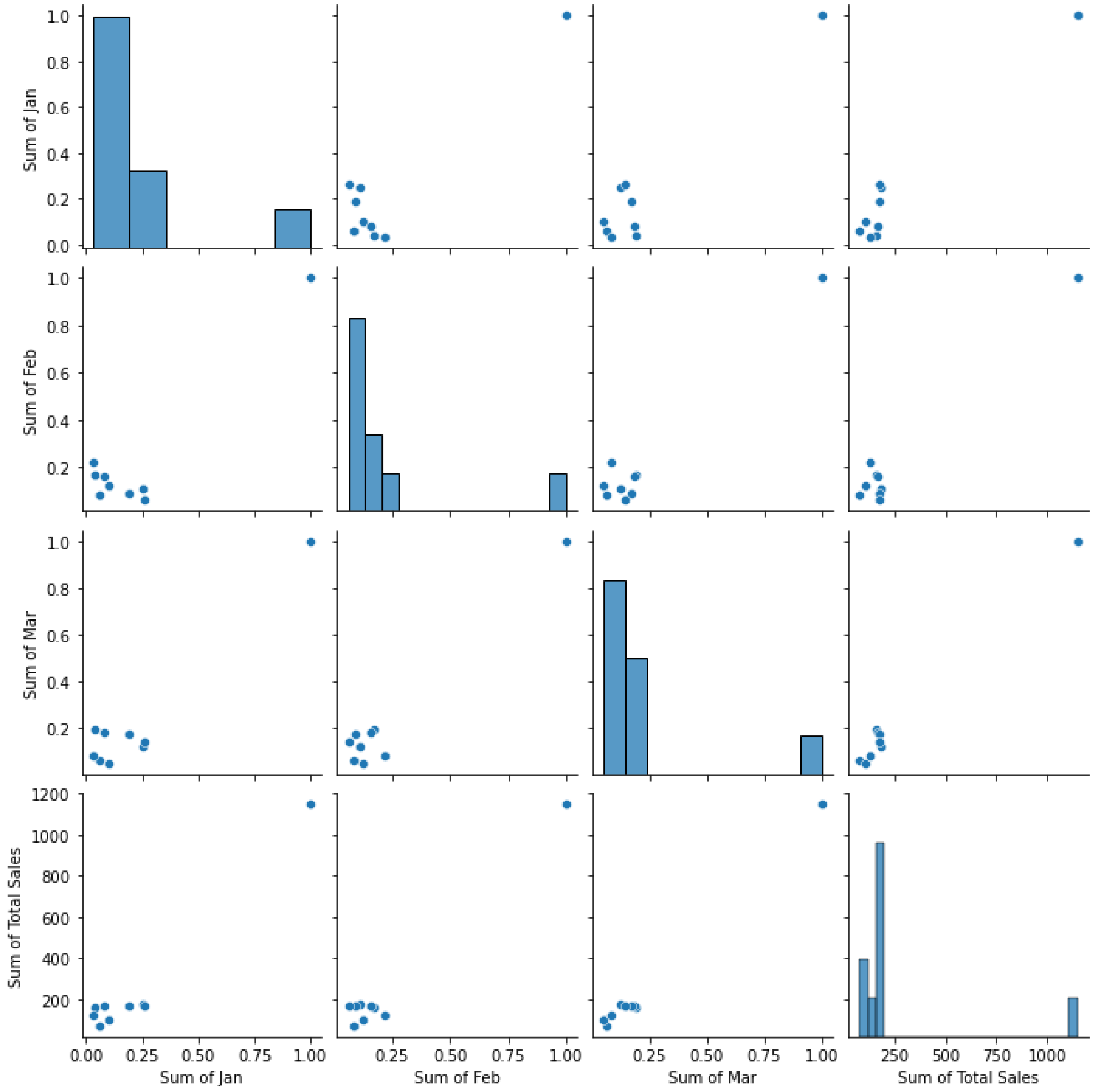
```
Index(['Row Labels', 'Sum of Jan', 'Sum of Feb', 'Sum of Mar',
       'Sum of Total Sales'],
      dtype='object')
```

In [12]:

```
sb.pairplot(df)
```

Out[12]:

```
<seaborn.axisgrid.PairGrid at 0x1da2cdb2b50>
```



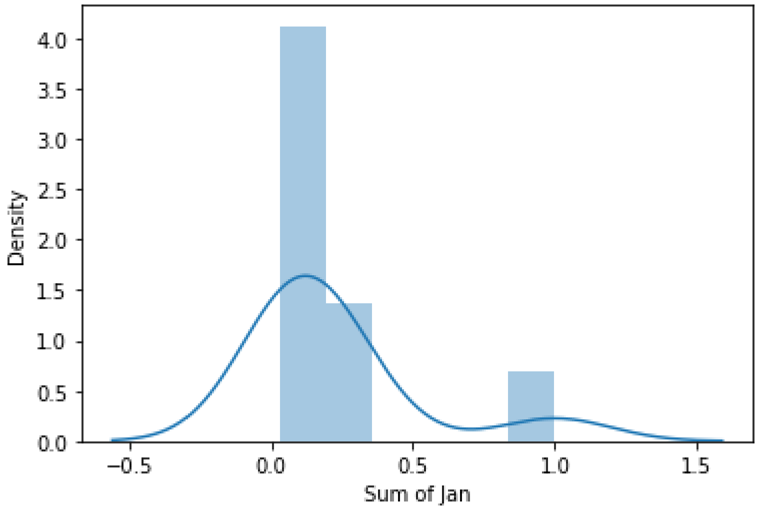
In [13]:

```
sb.distplot(df["Sum of Jan"])
```

```
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 adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `histplot` (an axes-level fun
ction for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[13]:

```
<AxesSubplot:xlabel='Sum of Jan', ylabel='Density'>
```



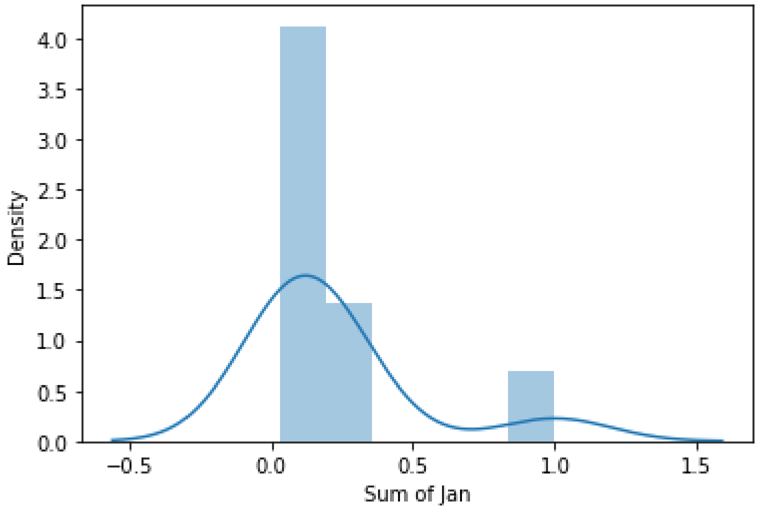
In [14]:

```
sb.distplot(df["Sum of Jan"])
```

```
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 adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `histplot` (an axes-level fun
ction for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[14]:

```
<AxesSubplot:xlabel='Sum of Jan', ylabel='Density'>
```



In [16]:

```
df1=df[['Sum of Jan', 'Sum of Feb', 'Sum of Mar',
       'Sum of Total Sales']]
df1
```

Out[16]:

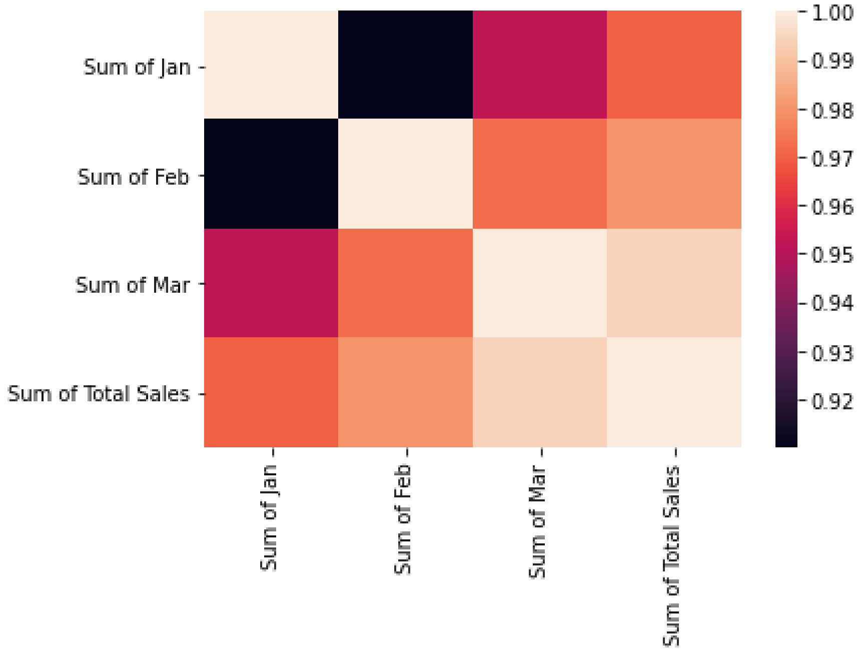
	Sum of Jan	Sum of Feb	Sum of Mar	Sum of Total Sales
0	0.06	0.08	0.06	75
1	0.04	0.17	0.19	160
2	0.10	0.12	0.05	101
3	0.03	0.22	0.08	127
4	0.25	0.11	0.12	179
5	0.08	0.16	0.18	167
6	0.19	0.09	0.17	171
7	0.26	0.06	0.14	170
8	1.00	1.00	1.00	1150

In [17]:

```
sb.heatmap(df1.corr())
```

Out[17]:

```
<AxesSubplot:>
```



model building

In [19]:

```
x = df1[['Sum of Jan', 'Sum of Feb', 'Sum of Mar',
       'Sum of Total Sales']]
y = df1['Sum of Jan']
```

In [20]:

```
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)
```

linear regression

In [21]:

```
from sklearn.linear_model import LinearRegression

lr = LinearRegression()
lr.fit(x_train,y_train)
```

Out[21]:

```
LinearRegression()
```

In [22]:

```
print(lr.intercept_)
```

```
0.0
```

In [23]:

```
coef = pd.DataFrame(lr.coef_,x.columns,columns=['Co_efficient'])
coef
```

Out[23]:

	Co_efficient
Sum of Jan	1.000000e+00
Sum of Feb	9.273251e-15
Sum of Mar	1.194422e-14
Sum of Total Sales	-2.583192e-17

In [24]:

```
print(lr.score(x_test,y_test))
```

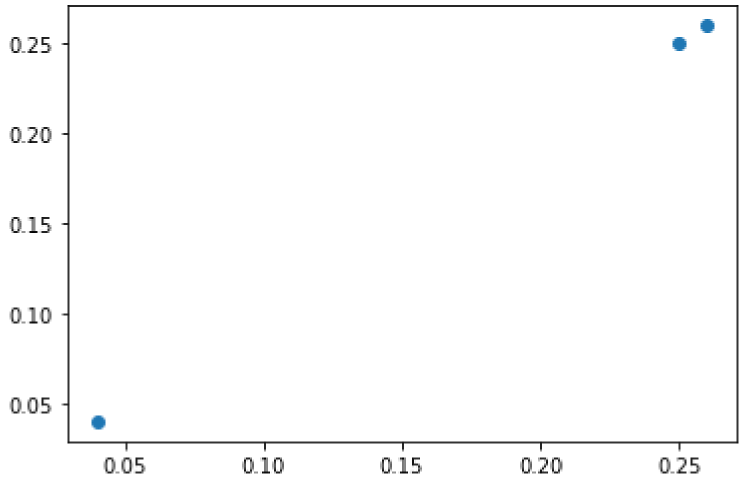
```
1.0
```

In [25]:

```
prediction = lr.predict(x_test)
pp.scatter(y_test,prediction)
```

Out[25]:

```
<matplotlib.collections.PathCollection at 0x1da2f98fcd0>
```



lasso and ridge regression

In [26]:

```
lr.score(x_test,y_test)
```

Out[26]:

```
1.0
```

In [27]:

```
lr.score(x_train,y_train)
```

Out[27]:

```
1.0
```

In [28]:

```
from sklearn.linear_model import Ridge,Lasso
```

In [29]:

```
r = Ridge(alpha=10)
r.fit(x_train,y_train)
r.score(x_test,y_test)
r.score(x_train,y_train)
```

Out[29]:

```
0.9847590101289931
```

In [30]:

```
l = Lasso(alpha=10)
l.fit(x_train,y_train)
l.score(x_test,y_test)
l.score(x_train,y_train)
```

Out[30]:

```
0.9788499697747168
```

elasticnet

In [31]:

```
from sklearn.linear_model import ElasticNet
e = ElasticNet()
e.fit(x_train,y_train)
```

Out[31]:

```
ElasticNet()
```

In [32]:

```
print(e.coef_)
```

```
[ 0.         -0.          0.          0.00088423]
```

In [33]:

```
print(e.intercept_)
```

```
-0.020610287604333356
```

In [34]:

```
predictions = e.predict(x_test)
predictions
```

Out[34]:

```
array([0.13766746, 0.12970936, 0.12086703])
```

In [35]:

```
print(e.score(x_test,y_test))
```

```
-0.17063906992605493
```

In [36]:

```
from sklearn import metrics
```

mean absolute error

```
In [37]:
```

```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,predictions))
```

```
Mean Absolute Error: 0.10783006866853657
```

mean squared error

```
In [38]:
```

```
print("Mean Squared Error:", metrics.mean_squared_error(y_test,predictions))
```

```
Mean Squared Error: 0.012044575319461409
```

root mean squared error

```
In [39]:
```

```
print("Root Mean Squared Error",np.sqrt(metrics.mean_squared_error(y_test,predictions)))
```

```
Root Mean Squared Error 0.10974778047624202
```

```
In [ ]:
```