In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [98]:
```
a=pd.read_csv(r"C:\Users\user\Downloads\6_Salesworkload1.csv")
a
```

0	10.2016	1.0	United Kingdom	88253.0	London (I)	1.0	Dry	3184.764	0.0	39
1	10.2016	1.0	United Kingdom	88253.0	London (I)	2.0	Frozen	1582.941	0.0	8
2	10.2016	1.0	United Kingdom	88253.0	London (I)	3.0	other	47.205	0.0	43
3	10.2016	1.0	United Kingdom	88253.0	London (I)	4.0	Fish	1623.852	0.0	30
4	10.2016	1.0	United Kingdom	88253.0	London (I)	5.0	Fruits & Vegetables	1759.173	0.0	16
...	...	...	...	...	...	...	...	...	...	
7653	06.2017	9.0	Sweden	29650.0	Gothenburg	12.0	Checkout	6322.323	0.0	388
7654	06.2017	9.0	Sweden	29650.0	Gothenburg	16.0	Customer Services	4270.479	0.0	
7655	06.2017	9.0	Sweden	29650.0	Gothenburg	11.0	Delivery	0	0.0	

In [99]:
```
a=a.head(10)
a
```

Out[99]:

	MonthYear	Time index	Country	StoreID	City	Dept_ID	Dept. Name	HoursOwn	HoursLease	Sales units	T
0	10.2016	1.0	United Kingdom	88253.0	London (I)	1.0	Dry	3184.764	0.0	398560.0	12
1	10.2016	1.0	United Kingdom	88253.0	London (I)	2.0	Frozen	1582.941	0.0	82725.0	3
2	10.2016	1.0	United Kingdom	88253.0	London (I)	3.0	other	47.205	0.0	438400.0	6
3	10.2016	1.0	United Kingdom	88253.0	London (I)	4.0	Fish	1623.852	0.0	309425.0	4
4	10.2016	1.0	United Kingdom	88253.0	London (I)	5.0	Fruits & Vegetables	1759.173	0.0	165515.0	3
5	10.2016	1.0	United Kingdom	88253.0	London (I)	6.0	Meat	8270.316	0.0	1713310.0	56
6	10.2016	1.0	United Kingdom	88253.0	London (I)	13.0	Food	16468.251	0.0	3107935.0	87
7	10.2016	1.0	United Kingdom	88253.0	London (I)	7.0	Clothing	4698.471	0.0	213680.0	16
8	10.2016	1.0	United Kingdom	88253.0	London (I)	8.0	Household	1183.272	0.0	54915.0	2
9	10.2016	1.0	United Kingdom	88253.0	London (I)	9.0	Hardware	2029.815	0.0	59260.0	4

In [100]:
```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 14 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   MonthYear      10 non-null     object 
 1   Time index     10 non-null     float64
 2   Country        10 non-null     object 
 3   StoreID        10 non-null     float64
 4   City           10 non-null     object 
 5   Dept_ID        10 non-null     float64
 6   Dept. Name     10 non-null     object 
 7   HoursOwn       10 non-null     object 
 8   HoursLease     10 non-null     float64
 9   Sales units    10 non-null     float64
 10  Turnover       10 non-null     float64
 11  Customer       0 non-null      float64
 12  Area (m2)      10 non-null     object 
 13  Opening hours  10 non-null     object 
dtypes: float64(7), object(7)
memory usage: 1.2+ KB
```

In [101]:
```
a.columns
```

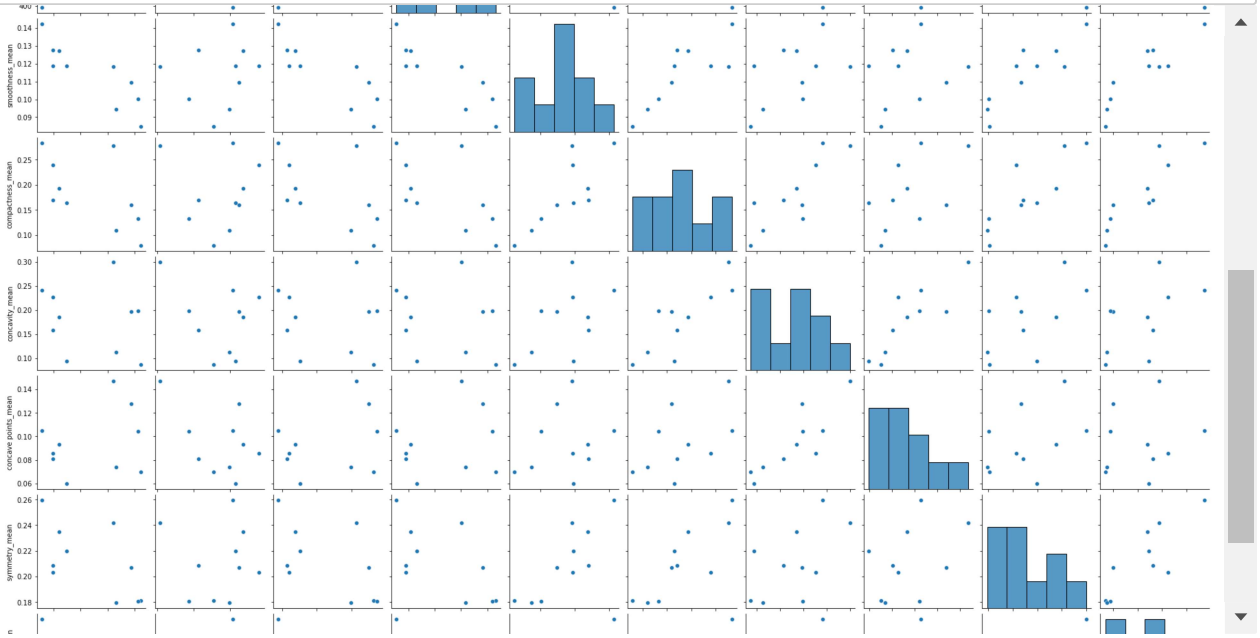
Out[101]:
```
Index(['MonthYear', 'Time index', 'Country', 'StoreID', 'City', 'Dept_ID',
       'Dept. Name', 'HoursOwn', 'HoursLease', 'Sales units', 'Turnover',
       'Customer', 'Area (m2)', 'Opening hours'],
      dtype='object')
```

In [102]:
```
a.describe()
```

Out[102]:

	Time index	StoreID	Dept_ID	HoursLease	Sales units	Turnover	Customer
count	10.0	10.0	10.000000	10.0	1.000000e+01	1.000000e+01	0.0
mean	1.0	88253.0	5.800000	0.0	6.543725e+05	1.978511e+06	NaN
std	0.0	0.0	3.614784	0.0	9.914003e+05	2.861420e+06	NaN
min	1.0	88253.0	1.000000	0.0	5.491500e+04	2.904000e+05	NaN
25%	1.0	88253.0	3.250000	0.0	1.034225e+05	4.033612e+05	NaN
50%	1.0	88253.0	5.500000	0.0	2.615525e+05	5.770455e+05	NaN
75%	1.0	88253.0	7.750000	0.0	4.284400e+05	1.518067e+06	NaN
max	1.0	88253.0	13.000000	0.0	3.107935e+06	8.714679e+06	NaN

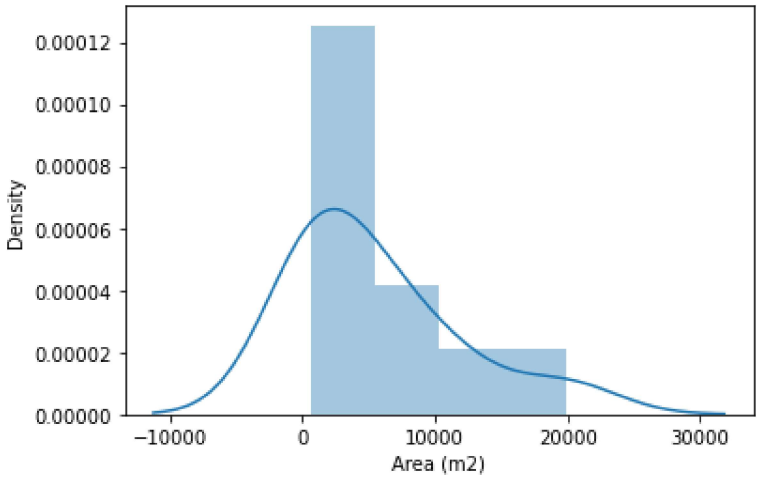
In [103]:
```
sns.pairplot(d)
```



In [116]:
```
sns.distplot(a['Area (m2)'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarnin
g: `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 flexibil
ity) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

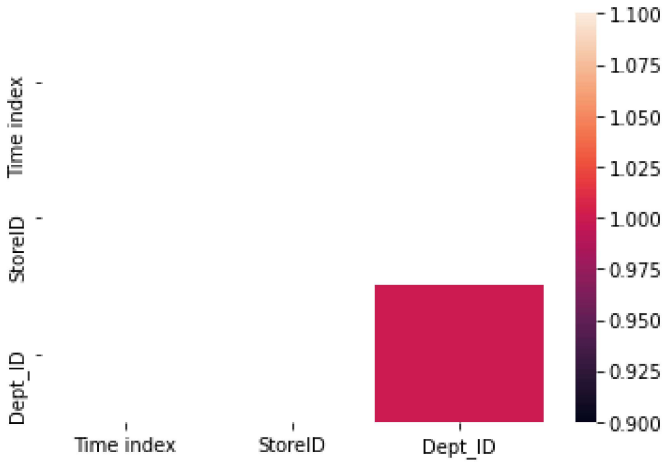
Out[116]: <AxesSubplot:xlabel='Area (m2)', ylabel='Density'>



In [117]:
```
x1=a[['Time index', 'StoreID', 'Dept_ID, 'HoursOwn']]
```

In [118]:
```
sns.heatmap(x1.corr())
```

Out[118]: <AxesSubplot:>



In [125]:
```
x=a[['Time index',  'StoreID', 'Dept_ID', 'HoursOwn']]
y=a['Area (m2)']
```

In [126]:
```
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)
```

In [127]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[127]:
```
LinearRegression()
```

In [128]:
```
print(lr.intercept_)
```

```
-1317.5818747019039
```

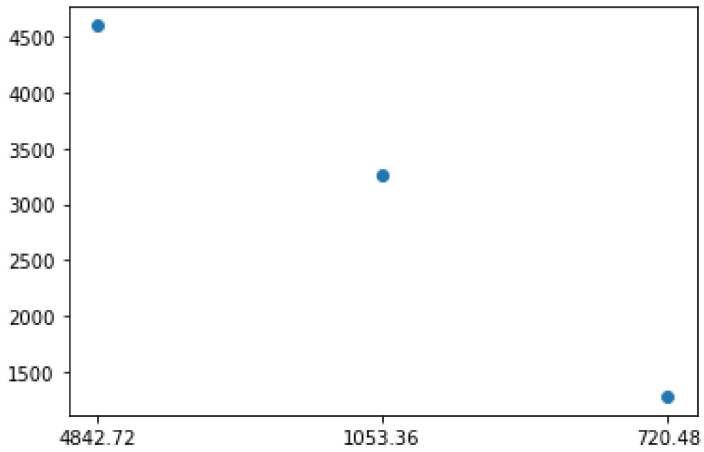
In [129]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[129]:

	Co-efficient
Time index	0.000000e+00
StoreID	6.282708e-11
Dept_ID	6.108624e+02
HoursOwn	8.714180e-01

In [130]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[130]:
```
<matplotlib.collections.PathCollection at 0x190b0d05e80>
```



In [131]:
```
print(lr.score(x_test,y_test))
```

```
0.49579438418782285
```

In [132]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [133]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[133]:
```
Ridge(alpha=10)
```

In [134]:
```
rr.score(x_test,y_test)
```

Out[134]:
```
0.4530736632860114
```

In [135]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[135]:
```
Lasso(alpha=10)
```

In [136]:
```
la.score(x_test,y_test)
```

Out[136]:
```
0.4955793902461354
```

In [137]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[137]:
```
ElasticNet()
```

In [138]:
```
print(en.coef_)
```

```
[  0.           0.         564.45308368   0.89621245]
```

In [139]:
```
print(en.intercept_)
```

```
-1161.153619593002
```

In [140]:
```
print(en.predict(x_test))
```

```
[4414.93415066 3237.70454711 1386.40398319]
```

In [141]:
```
en.score(x_test,y_test)
```

Out[141]:
```
0.48531638894255613
```

In [142]:
```
from sklearn import metrics
```

In [143]:
```
print("Mean Absolute Error",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error 1007.2298471066747
```

In [144]:
```
print("Mean Squared Error",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error 1762642.379836129
```

In [145]:
```
print(" Root Mean Squared Error",np.sqrt(metrics.mean_squared_error(y_test,prediction))
```

```
 Root Mean Squared Error 1327.645427000797
```