DBSCAN

▼ DBSCAN vs K-Means

DBSCAN and Clustering Examples

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

✓ 10.5s

```python
blobs = pd.read_csv('cluster_blobs.csv')
```

✓ 0.9s

```python
blobs.head()
```

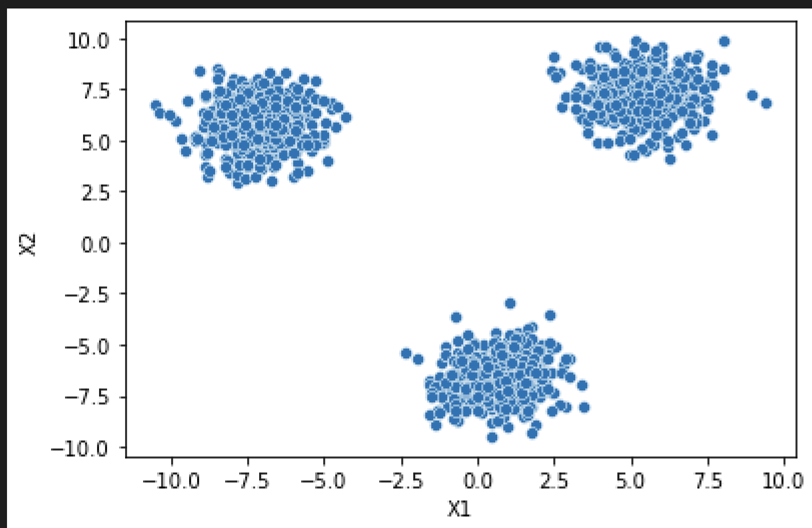
✓ 0.2s

	X1	X2
0	4.645333	6.822294
1	4.784032	6.422883
2	-5.851786	5.774331
3	-7.459592	6.456415
4	4.918911	6.961479

```python
1  sns.scatterplot(data=blobs, x="X1", y="X2")
```

✓ 0.7s

```
<AxesSubplot:xlabel='X1', ylabel='X2'>
```



```python
moons = pd.read_csv('cluster_moons.csv')
```

✓ 0.9s

```python
moons.tail()
```

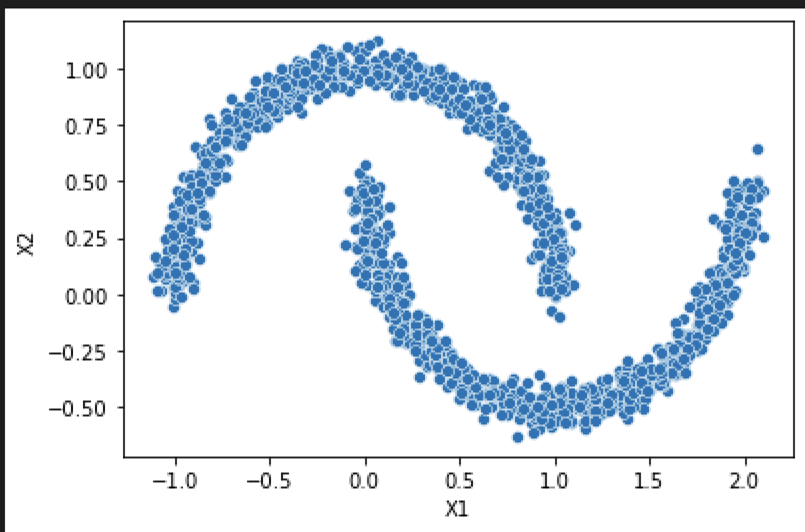
✓ 0.1s

	X1	X2
1495	1.957344	0.187184
1496	0.962394	0.384304
1497	-0.761893	0.581666
1498	1.803858	-0.154705
1499	0.203305	0.079049

```python
sns.scatterplot(data=moons, x="X1", y="X2")
```

✓ 0.6s

<AxesSubplot:xlabel='X1', ylabel='X2'>



```python
circles = pd.read_csv('cluster_circles.csv')
```

✓ 0.9s

```python
circles.head()
```

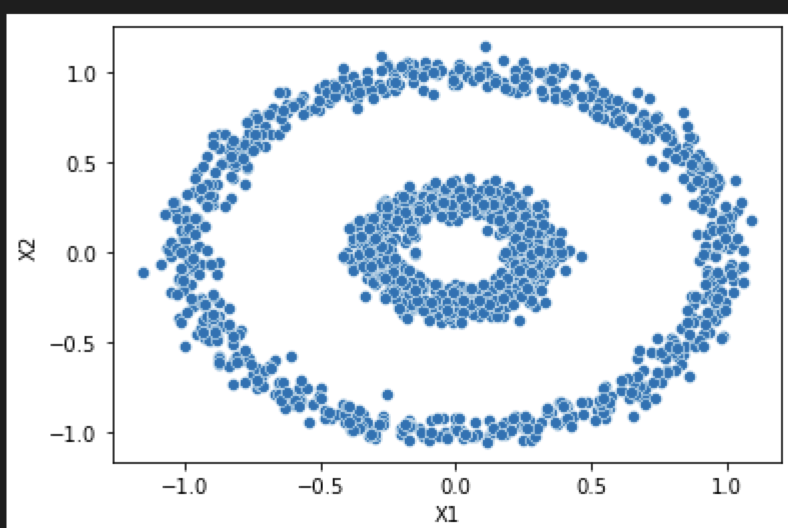
✓ 0.1s

	X1	X2
0	-0.348677	0.010157
1	-0.176587	-0.954283
2	0.301703	-0.113045
3	-0.782889	-0.719468
4	-0.733280	-0.757354

```python
sns.scatterplot(data=circles, x="X1", y="X2")
```

✓ 0.6s

```
<AxesSubplot:xlabel='X1', ylabel='X2'>
```

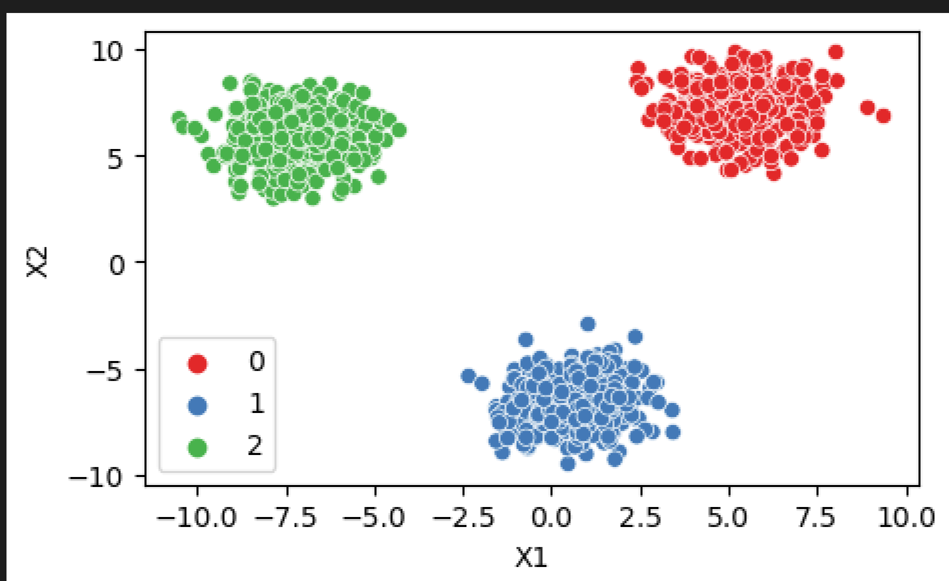


```python
def display_categories(model,data):

    labels = model.fit_predict(data)
    sns.scatterplot(data=data, x='X1', y='X2', hue=labels, palette="Set1")
```
✓ 0.1s

K_Means

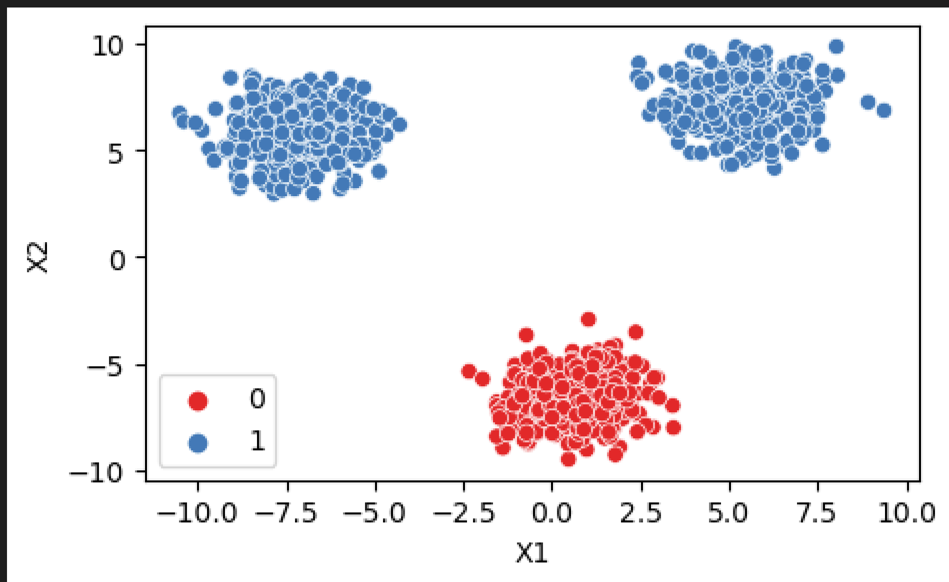
```python
from sklearn.cluster import KMeans
```
6] ✓ 0.5s

```python
model = KMeans(n_clusters=3)
plt.figure(figsize=(5,3),dpi=100)
display_categories(model,blobs)
```
4] ✓ 0.8s



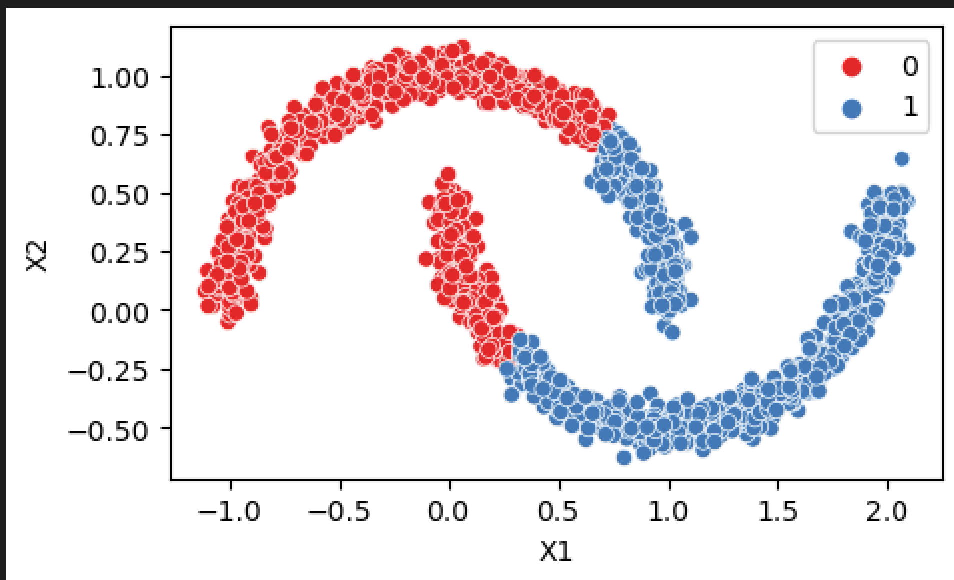
```python
model = KMeans(n_clusters=2)
plt.figure(figsize=(5,3),dpi=100)
display_categories(model,blobs)
```

✓ 0.6s



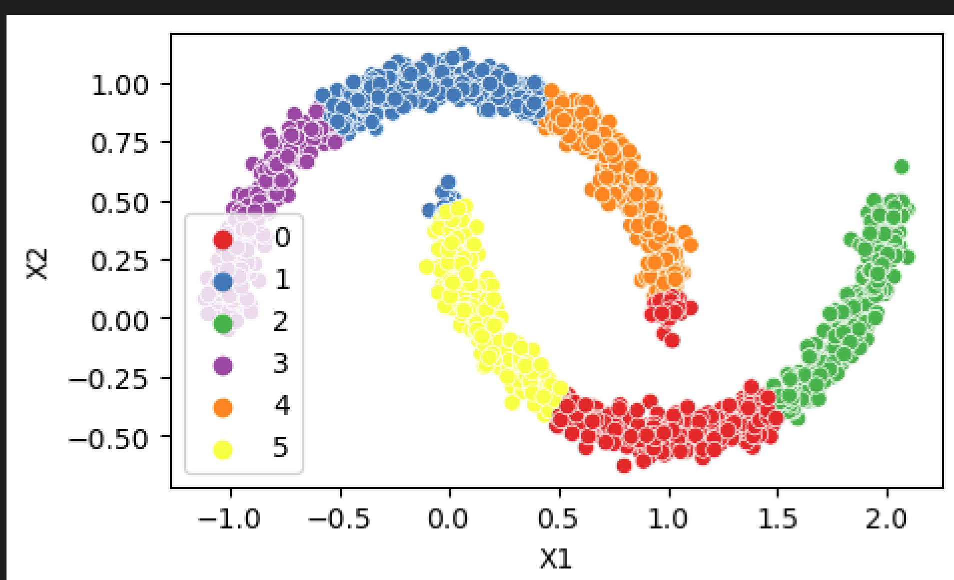
```python
model = KMeans(n_clusters=2)
plt.figure(figsize=(5,3),dpi=100)
display_categories(model,moons)
```

✓ 0.5s



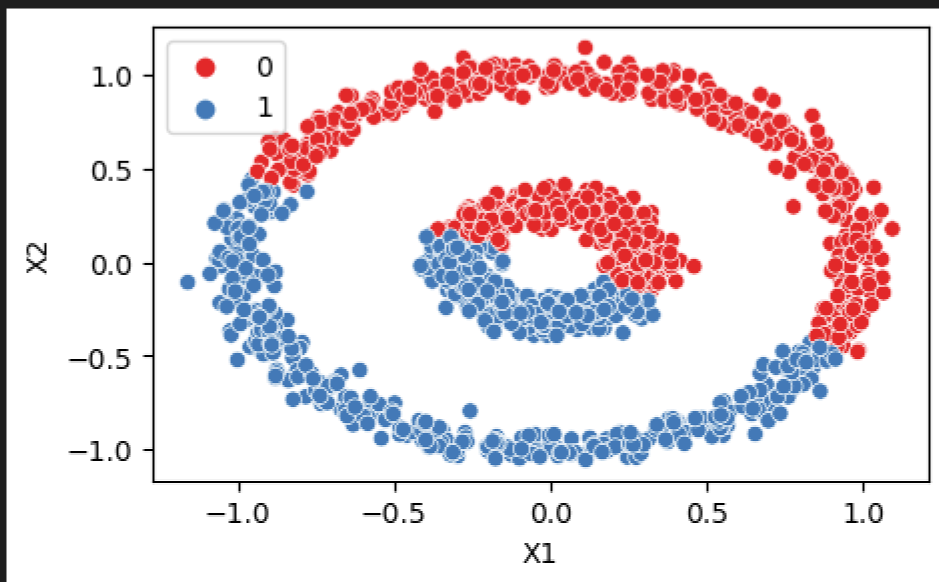

```python
model = KMeans(n_clusters=6)
plt.figure(figsize=(5,3),dpi=100)
display_categories(model,moons)
```

✓ 0.9s



```
model = KMeans(n_clusters=2)
plt.figure(figsize=(5,3),dpi=100)
display_categories(model,circles)
```

0.5s



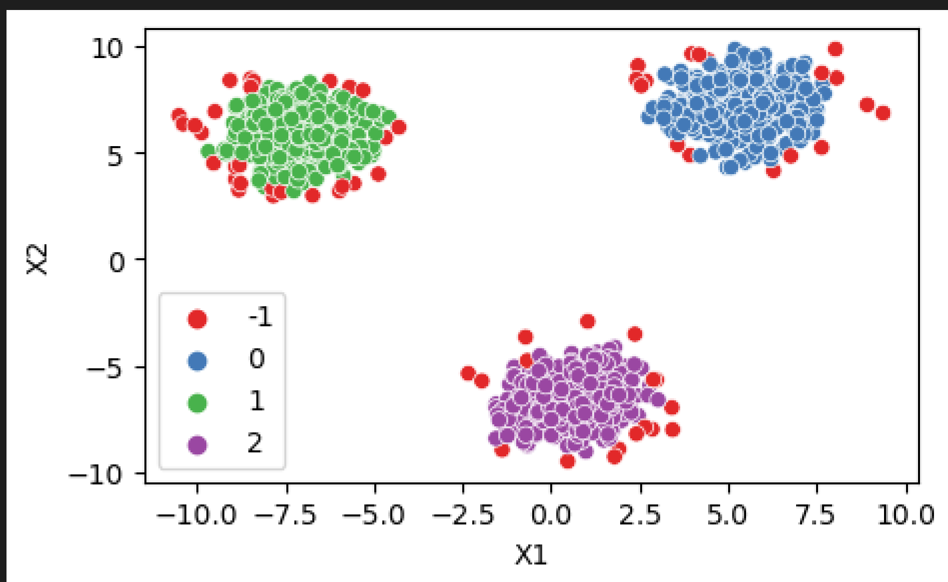
DBSCAN

```
from sklearn.cluster import DBSCAN
```

✓ 0.1s

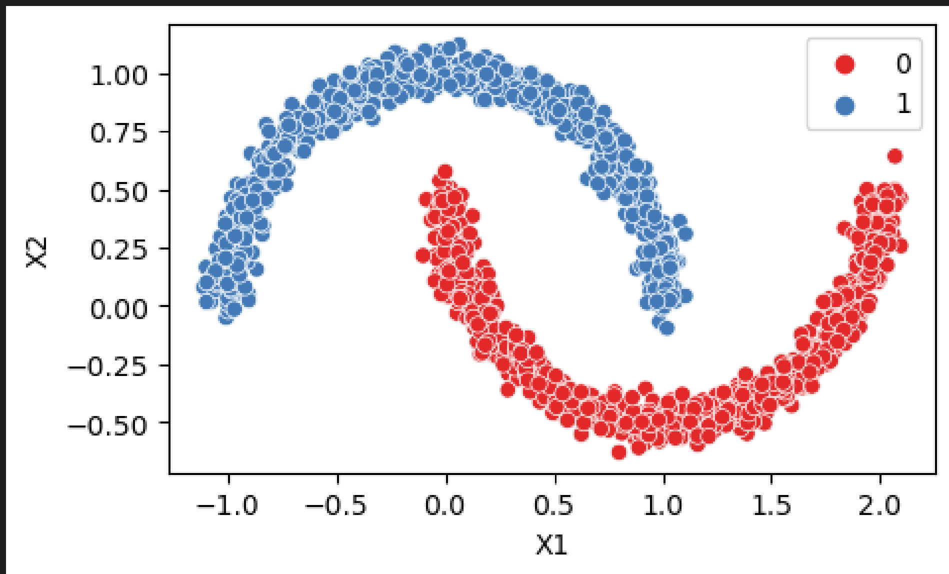
```
model = DBSCAN()
plt.figure(figsize=(5,3),dpi=100)
display_categories(model,blobs)
```

✓ 0.9s



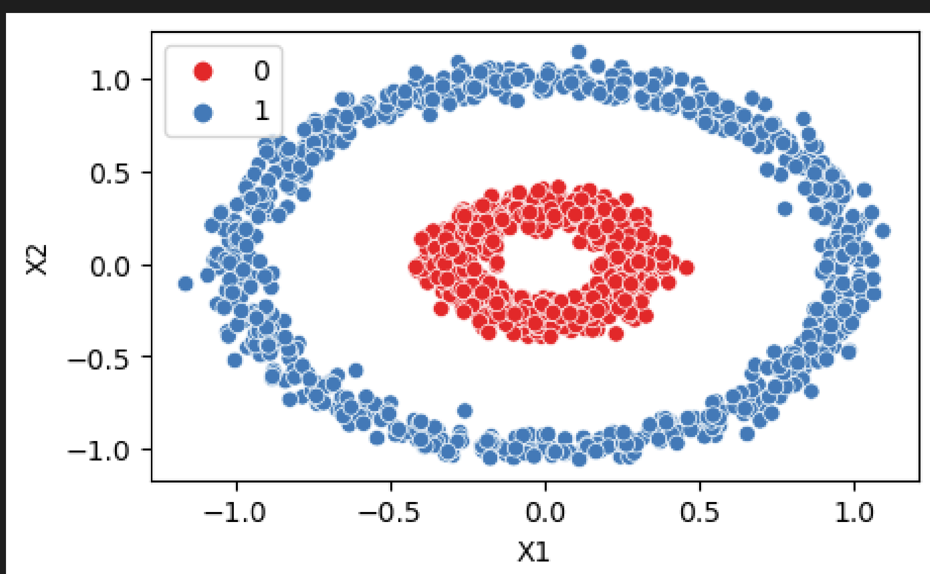
```
model = DBSCAN(eps=0.15)
plt.figure(figsize=(5,3),dpi=100)
display_categories(model,moons)
```

✓ 0.4s




```
model = DBSCAN(eps=0.15)
plt.figure(figsize=(5,3),dpi=100)
display_categories(model,circles)
```

✓ 0.6s



▼ Hyperparameters

DBSCAN Hyperparameters

Hyperparameter Examples

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

✓ 0.2s

```python
two_blobs = pd.read_csv('cluster_two_blobs.csv')
two_blobs_outliers = pd.read_csv('cluster_two_blobs_outliers.csv')
```

✓ 0.1s

```python
two_blobs
```

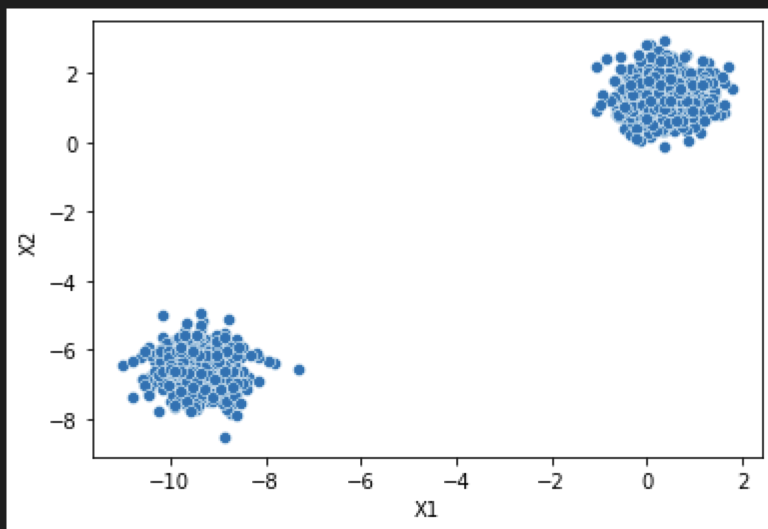
✓ 0.1s

	X1	X2
0	0.046733	1.765120
1	-8.994134	-6.508186
2	0.650539	1.264533

```python
sns.scatterplot(data=two_blobs, x="X1", y="X2")
```

✓ 0.4s

```
<AxesSubplot:xlabel='X1', ylabel='X2'>
```



```python
two_blobs_outliers.head()
```

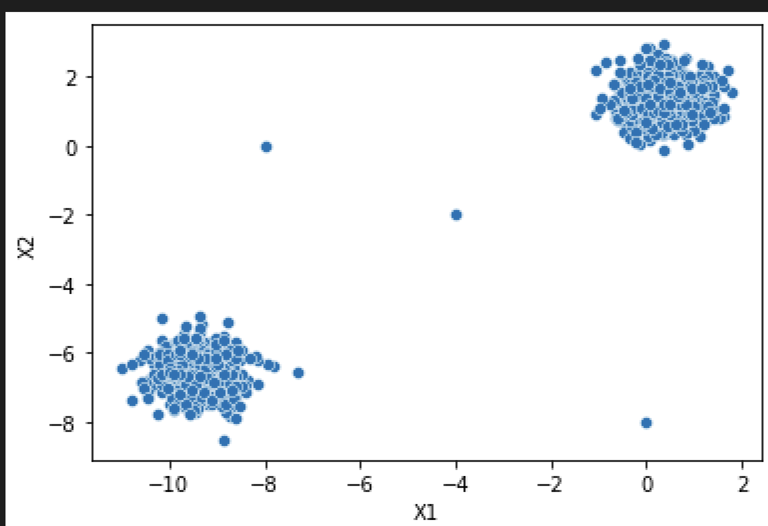
✓ 0.7s

	X1	X2
0	0.046733	1.765120
1	-8.994134	-6.508186
2	0.650539	1.264533
3	-9.501554	-6.736493
4	0.057050	0.188215

```python
sns.scatterplot(data=two_blobs_outliers, x="X1", y="X2")
```

✓ 0.4s

```
<AxesSubplot:xlabel='X1', ylabel='X2'>
```




```python
def display_categories(model,data):

    labels = model.fit_predict(data)
    sns.scatterplot(data=data, x="X1", y="X2", hue=labels, palette="Set1")
```

DBSCAN

```python
from sklearn.cluster import DBSCAN
```

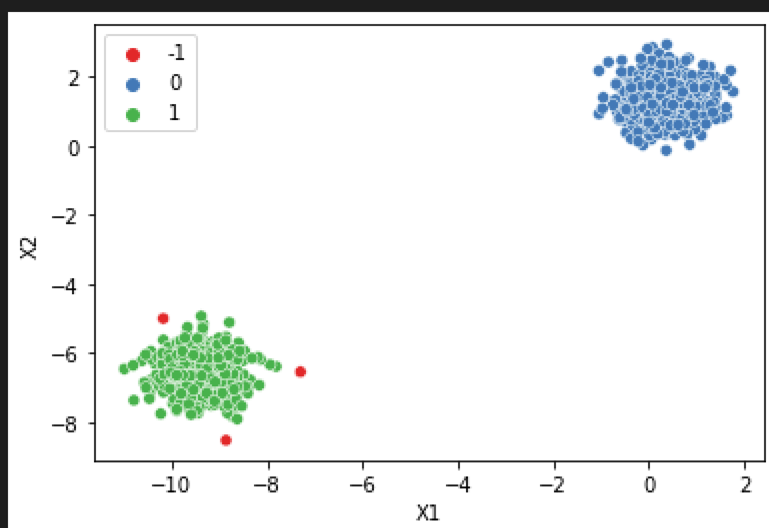
✓ 0.9s

```python
dbscan = DBSCAN()
```

✓ 0.8s

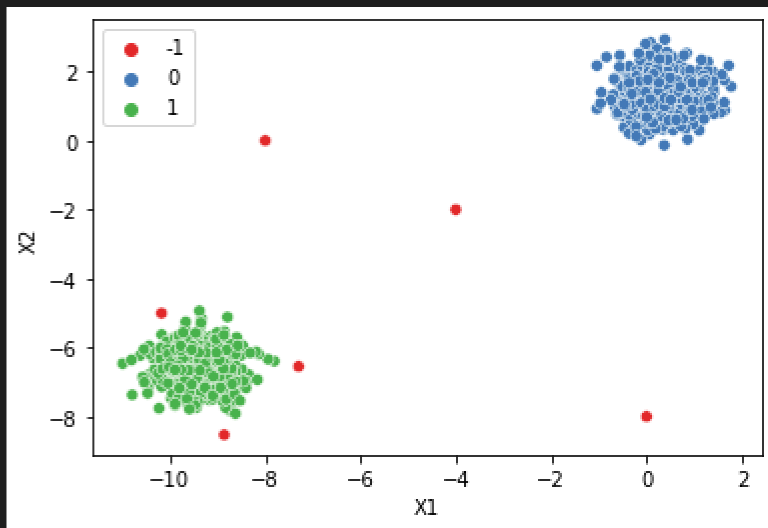
```python
display_categories(dbscan, two_blobs)
```

✓ 0.5s



```
display_categories(dbscan, two_blobs_outliers)
```

0.4s

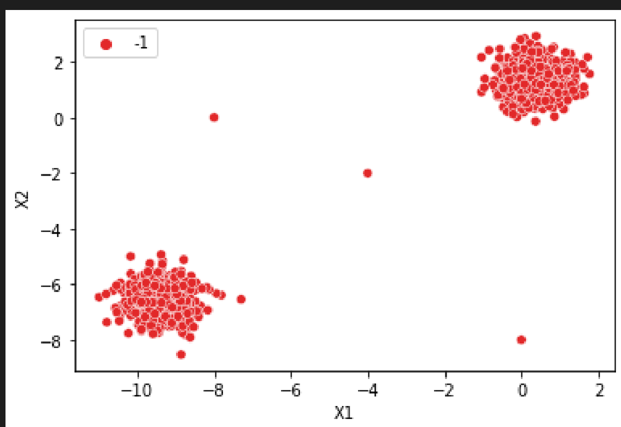


Epsilon

```
eps : float, default=0.5
 |      The maximum distance between two samples for one to be
considered
 |      as in the neighborhood of the other. This is not a maximum
bound
 |      on the distances of points within a cluster. This is the most
 |      important DBSCAN parameter to choose appropriately for your
data set
 |      and distance function.
```

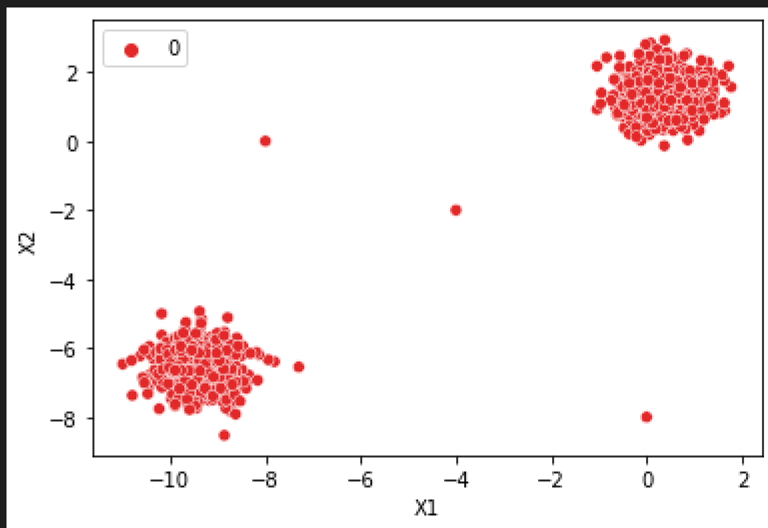
```python
dbscan = DBSCAN(eps=0.001)
display_categories(dbscan, two_blobs_outliers)
```

✓ 0.4s Python



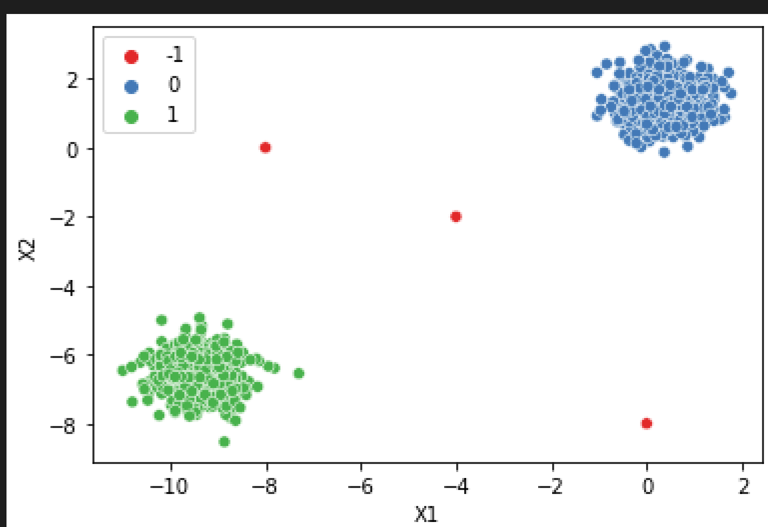
```python
dbscan = DBSCAN(eps=10)
display_categories(dbscan,two_blobs_outliers)
```

✓ 0.4s




```python
dbscan = DBSCAN(eps=1)
display_categories(dbscan,two_blobs_outliers)
```

✓ 0.5s



```python
np.sum(dbscan.labels_== -1)
```

0.7s

```
3
```

```python
100 * np.sum(dbscan.labels_ == -1) / len(dbscan.labels_)
```

0.1s

```
0.29910269192422734
```

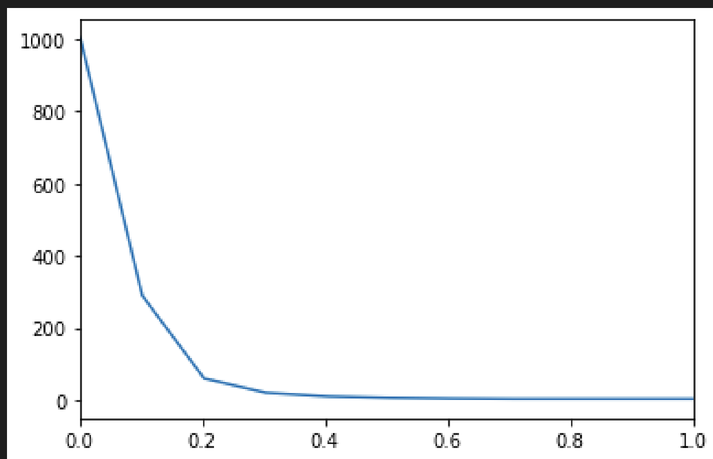
```python
outlier_percent = []
number_of_outliers = []

for eps in np.linspace(0.001,10,100):

    # Create Model
    dbscan = DBSCAN(eps=eps)
    dbscan.fit(two_blobs_outliers)

    # Log Number of Outliers
    number_of_outliers.append(np.sum(dbscan.labels_ == -1))

    # Log percentage of points that are outliers
    perc_outliers = 100 * np.sum(dbscan.labels_ == -1) / len(dbscan.labels_)

    outlier_percent.append(perc_outliers)

```

1.8s

```
sns.lineplot(x=np.linspace(0.001,10,100), y=number_of_outliers)
plt.xlim(0,1)
```

✓ 0.4s

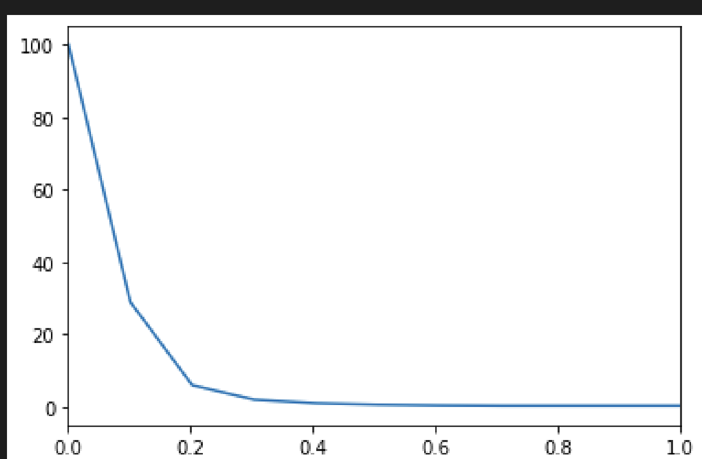
```
(0.0, 1.0)
```




```
sns.lineplot(x=np.linspace(0.001,10,100), y=outlier_percent)
plt.xlim(0,1)
```

✓ 0.3s

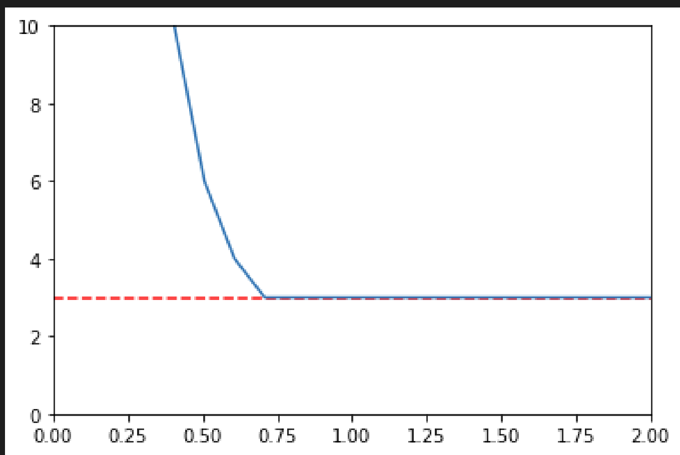
```
(0.0, 1.0)
```



```
sns.lineplot(x=np.linspace(0.001,10,100), y=number_of_outliers)
plt.xlim(0,1)
plt.ylim(0,10)
plt.xlim(0,2)
plt.hlines(y=3,xmin=0,xmax=2,colors='red',ls='--')
```

0.3s

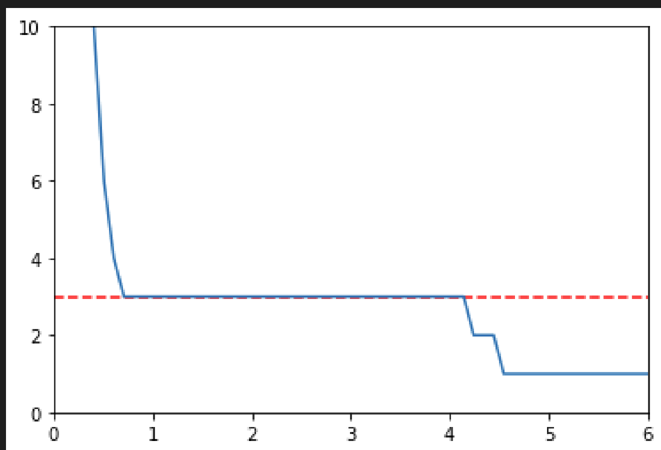
```
<matplotlib.collections.LineCollection at 0x251d8d62b50>
```



```python
sns.lineplot(x=np.linspace(0.001,10,100),y=number_of_outliers)
plt.ylim(0,10)
plt.xlim(0,6)
plt.hlines(y=3,xmin=0,xmax=10,colors='red',ls='--')
```

✓ 0.3s

```
&lt;matplotlib.collections.LineCollection at 0x251d8dfb070&gt;
```



Minimum Samples

```
|  min_samples : int, default=5
|      The number of samples (or total weight) in a neighborhood for a point
|      to be considered as a core point. This includes the point itself.
```

How to choose minimum number of points?

https://stats.stackexchange.com/questions/88872/a-routine-to-choose-eps-and-minpts-for-dbscan

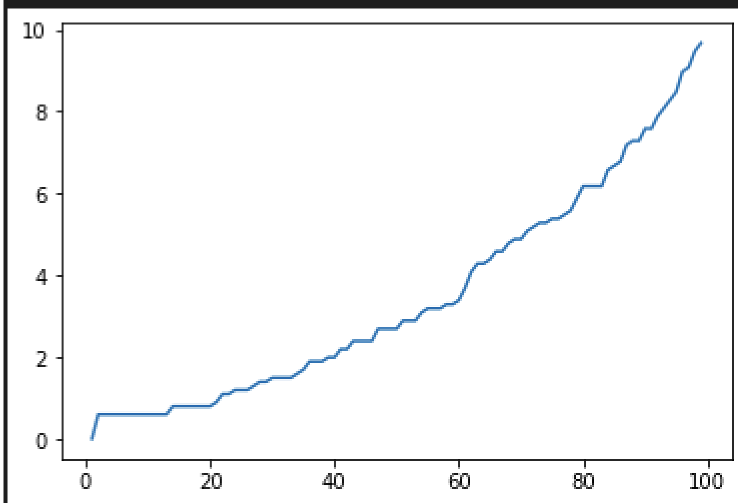
```python
outlier_percent = []

for n in np.arange(1,100):

    # Create Model
    dbscan = DBSCAN(min_samples=n)
    dbscan.fit(two_blobs_outliers)

    # Log percentage of points that are outliers
    perc_outliers = 100 * np.sum(dbscan.labels_ == -1) / len(dbscan.labels_)

    outlier_percent.append(perc_outliers)

```

✓ 1.8s

```python
sns.lineplot(x=np.arange(1,100),y=outlier_percent)
```

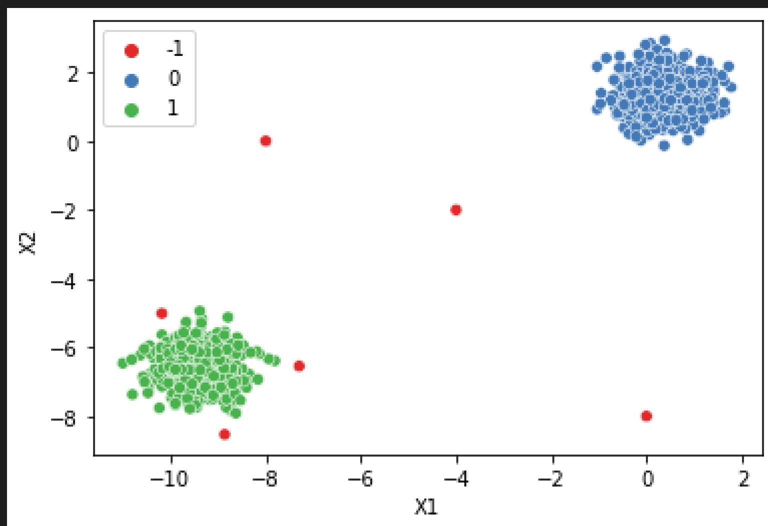
✓ 0.3s

```
<AxesSubplot:>
```



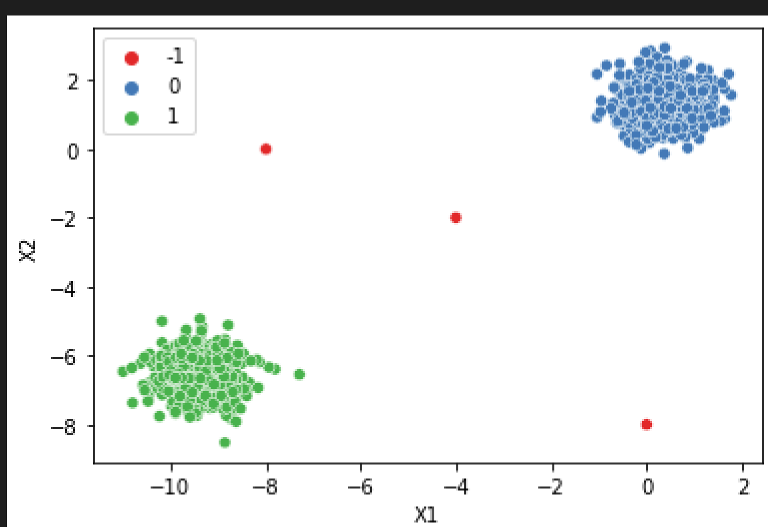
```python
num_dim = two_blobs_outliers.shape[1]

dbscan = DBSCAN(min_samples=2*num_dim)
display_categories(dbscan,two_blobs_outliers)
```

✓ 0.5s



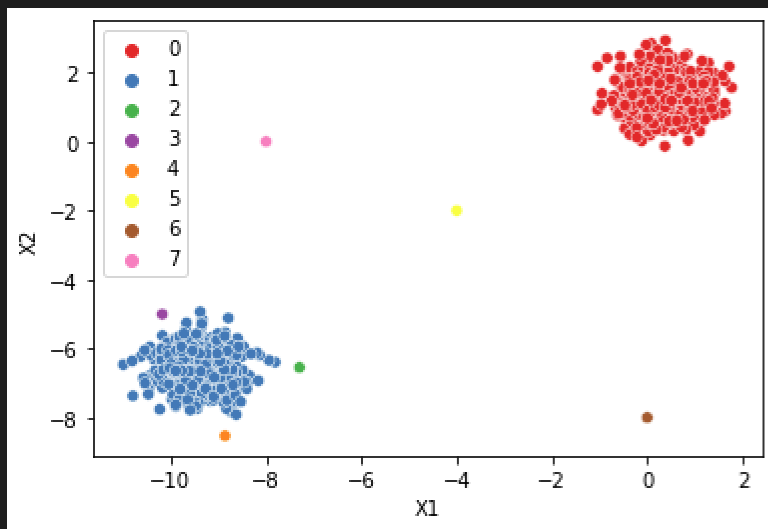

```python
num_dim = two_blobs_outliers.shape[1]

dbscan = DBSCAN(eps=0.75,min_samples=2*num_dim)
display_categories(dbscan,two_blobs_outliers)
```

✓ 0.5s



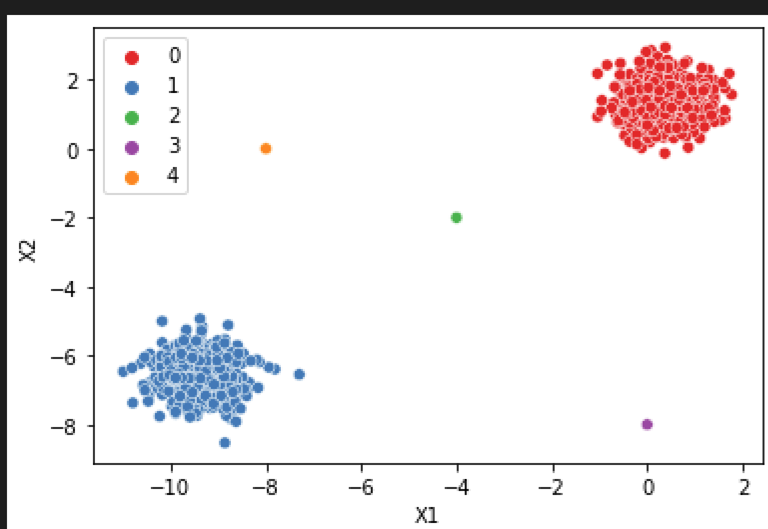
```python
dbscan = DBSCAN(min_samples=1)
display_categories(dbscan,two_blobs_outliers)
```

✓ 0.9s




```python
dbscan = DBSCAN(eps=0.75,min_samples=1)
display_categories(dbscan,two_blobs_outliers)
```

✓ 0.6s



▼ Customer Data Project

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

✓ 2.7s

```python
df = pd.read_csv('wholesome_customers_data.csv')
```

✓ 0.7s

```python
df
```

✓ 0.4s

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
0	2	3	12669	9656	7561	214	2674	1338
1	2	3	7057	9810	9568	1762	3293	1776
2	2	3	6353	8808	7684	2405	3516	7844
3	1	3	13265	1196	4221	6404	507	1788
4	2	3	22615	5410	7198	3915	1777	5185
...	...	...	...	...	...	...	...	...

```
1  df.info()
```

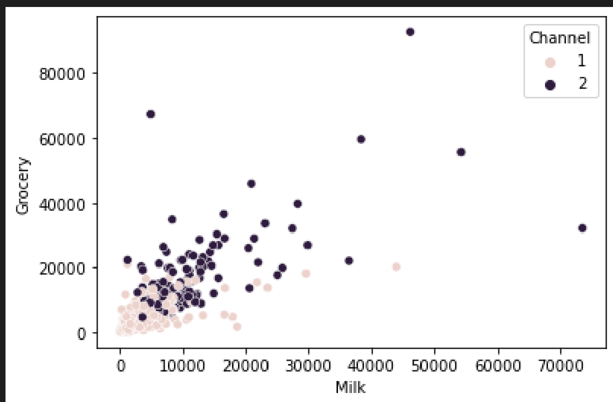
✓ 0.1s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 440 entries, 0 to 439
Data columns (total 8 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Channel           440 non-null    int64
 1   Region            440 non-null    int64
 2   Fresh             440 non-null    int64
 3   Milk              440 non-null    int64
 4   Grocery           440 non-null    int64
 5   Frozen            440 non-null    int64
 6   Detergents_Paper  440 non-null    int64
 7   Delicassen        440 non-null    int64
dtypes: int64(8)
memory usage: 27.6 KB
```

EDA

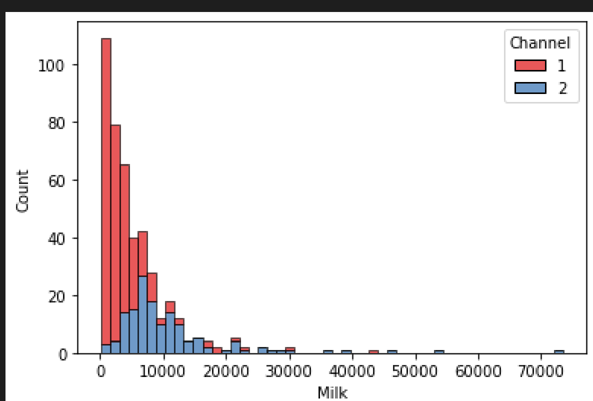
```
1  sns.scatterplot(data=df, x="Milk", y="Grocery", hue="Channel")
✓ 0.5s
```

```
<AxesSubplot:xlabel='Milk', ylabel='Grocery'>
```




```
1  sns.histplot(data=df, x="Milk", hue="Channel", palette="Set1", multiple="stack")
✓ 0.5s
```

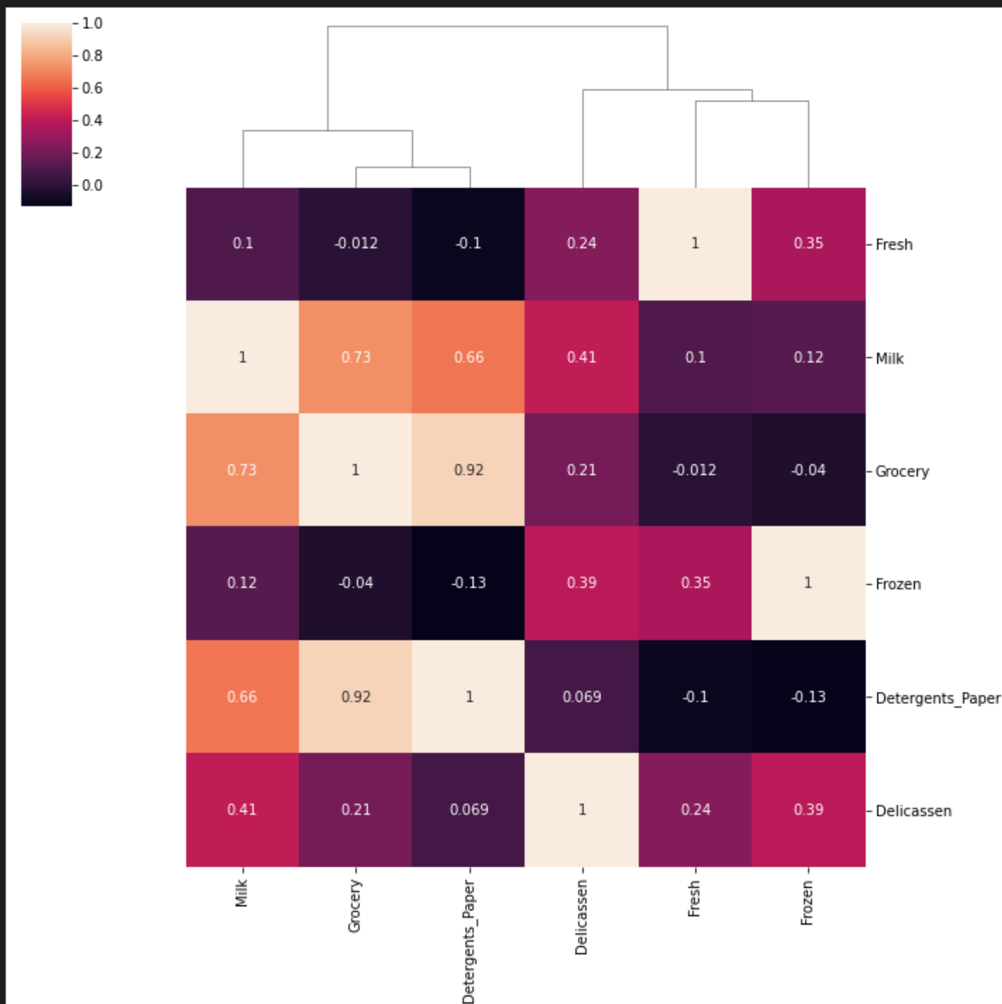
```
<AxesSubplot:xlabel='Milk', ylabel='Count'>
```



```python
clustermap(df.drop(["Region","Channel"], axis=1).corr(), annot=True, row_cluster=False)
```

✓ 1.5s

```
<seaborn.matrix.ClusterGrid at 0x1c5b326d3d0>
```

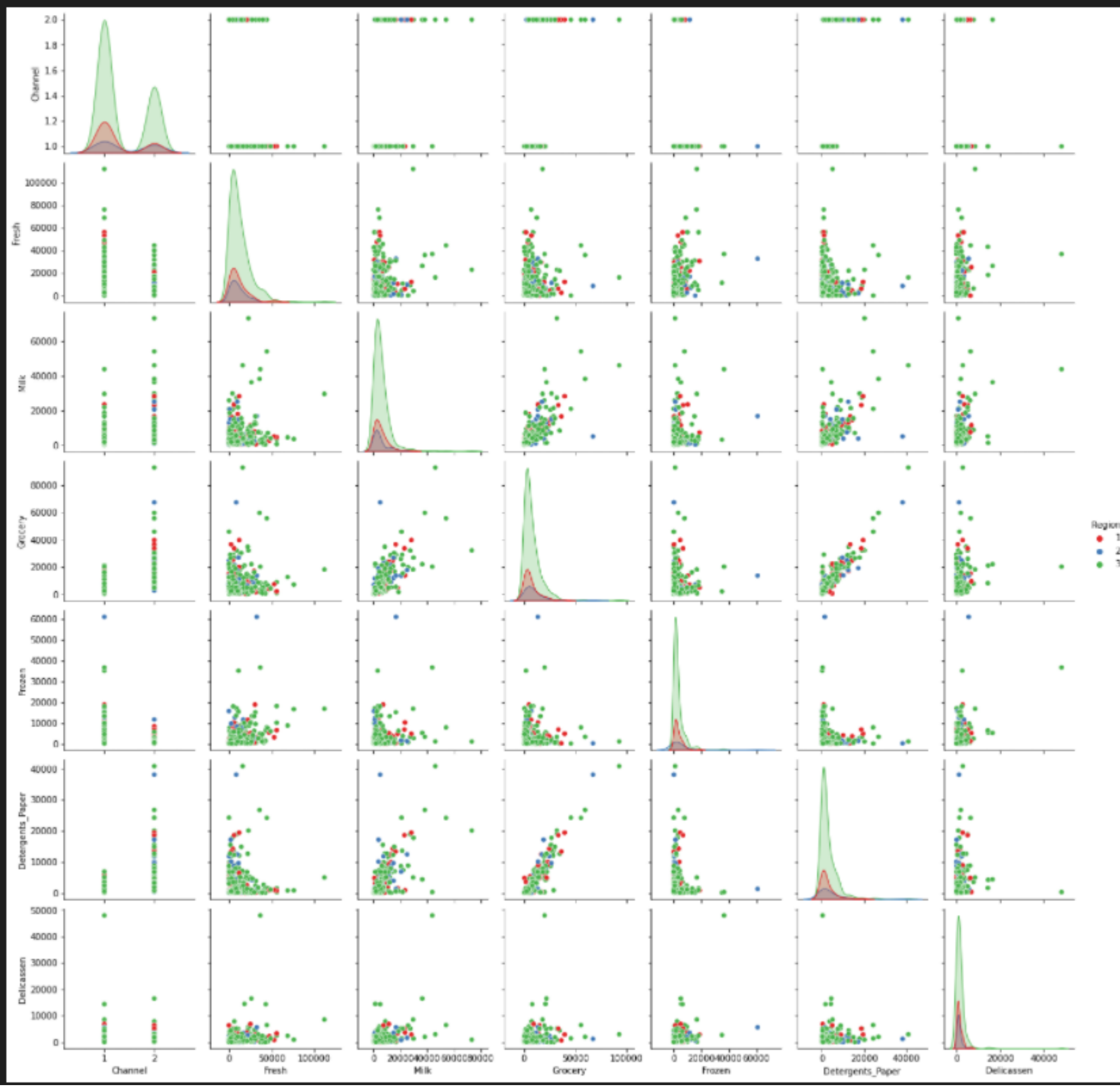


```python
1  sns.pairplot(df, hue="Region", palette="Set1")
```

✓ 18.7s

Python

```
<seaborn.axisgrid.PairGrid at 0x1c5b37bbac0>
```



ML Model

```python
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

✓ 0.2s

```python
scaled_X = scaler.fit_transform(df)
```

✓ 0.1s

```python
from sklearn.cluster import DBSCAN
```

✓ 0.7s

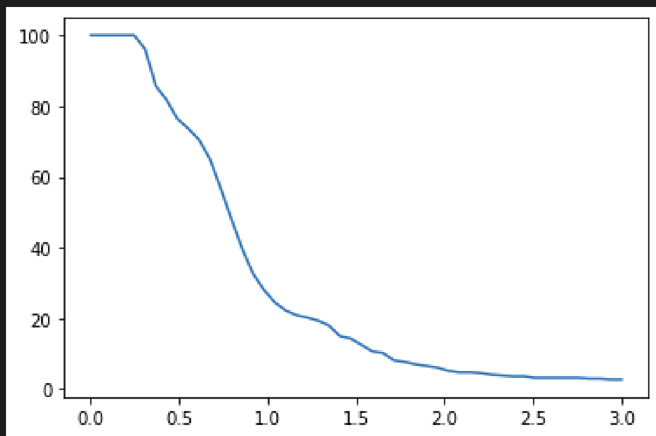
```python
outlier_percent = []

for eps in np.linspace(0.001,3,50):

    dbscan = DBSCAN(eps=eps, min_samples=2*scaled_X.shape[1])
    dbscan.fit(scaled_X)

    perc_outliers = 100*np.sum(dbscan.labels_== -1) / len(dbscan.labels_)

    outlier_percent.append(perc_outliers)
```

✓ 0.8s

```python
sns.lineplot(x=np.linspace(0.001,3,50), y=outlier_percent)
```

✓ 0.3s

```
<AxesSubplot:>
```




```python
dbscan = DBSCAN(eps=2, min_samples=scaled_X.shape[1])
```

✓ 0.1s

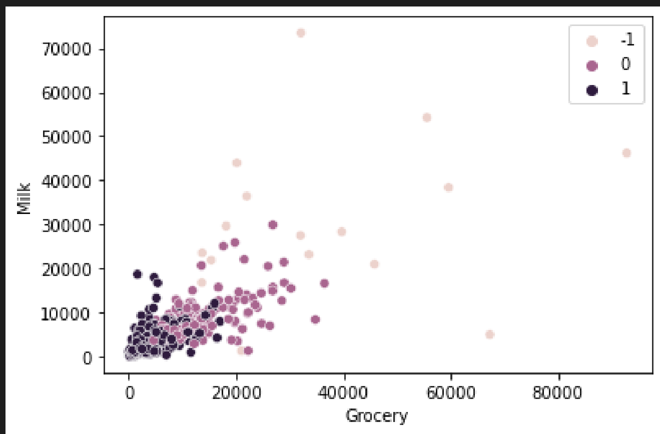
```python
dbscan.fit(scaled_X)
```

✓ 0.8s

```
DBSCAN(eps=2, min_samples=8)
```

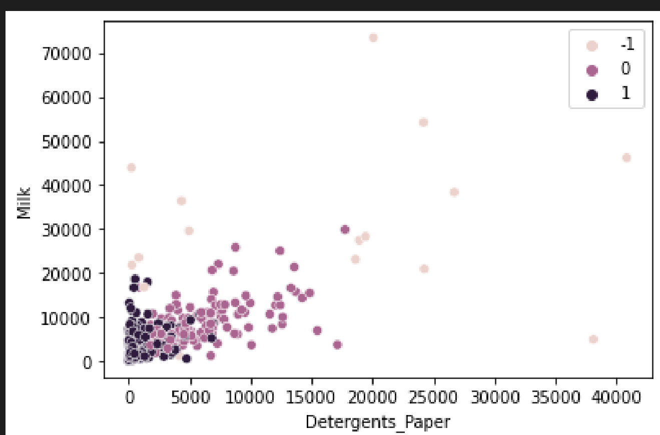
```
1  sns.scatterplot(data=df,x='Grocery',y='Milk',hue=dbscan.labels_)
```
✓ 0.7s

```
<AxesSubplot:xlabel='Grocery', ylabel='Milk'>
```




```
1  sns.scatterplot(data=df,x='Detergents_Paper',y='Milk',hue=dbscan.labels_)
```
✓ 0.8s

```
<AxesSubplot:xlabel='Detergents_Paper', ylabel='Milk'>
```



```python
1 df["Labels"] = dbscan.labels_
```
✓ 0.1s

```python
1 df
```
✓ 0.1s

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen	Labels
0	2	3	12669	9656	7561	214	2674	1338	0
1	2	3	7057	9810	9568	1762	3293	1776	0
2	2	3	6353	8808	7684	2405	3516	7844	0
3	1	3	13265	1196	4221	6404	507	1788	1
4	2	3	22615	5410	7198	3915	1777	5185	0
...	...	...	...	...	...	...	...	...	...
435	1	3	29703	12051	16027	13135	182	2204	1
436	1	3	39228	1431	764	4510	93	2346	1
437	2	3	14531	15488	30243	437	14841	1867	0
438	1	3	10290	1981	2232	1038	168	2125	1
439	1	3	2787	1698	2510	65	477	52	1

440 rows × 9 columns

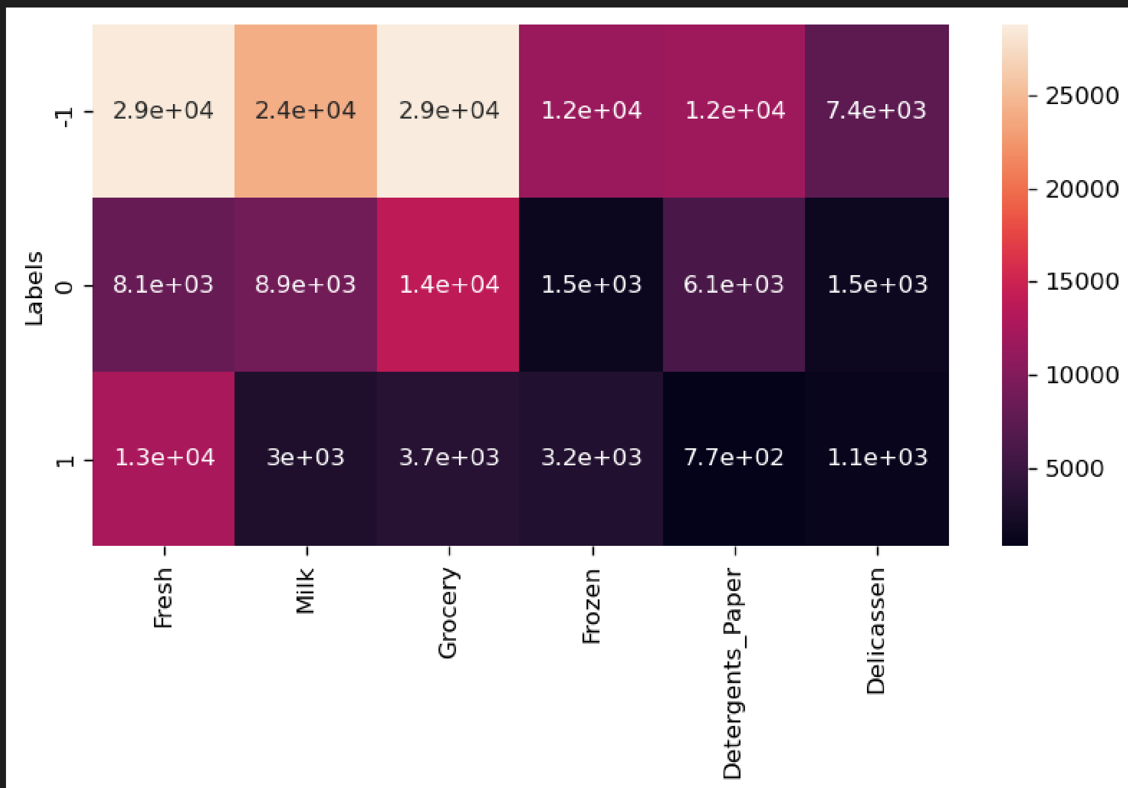
```python
1 cats = df.drop(["Channel","Region"], axis=1)
2 cat_mean = cats.groupby("Labels").mean()
3 cat_mean
```
✓ 0.6s

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
Labels						
-1	28678.285714	24176.523810	28797.857143	11535.000000	11932.523810	7367.380952
0	8134.862595	8909.916031	14004.427481	1450.595420	6080.832061	1533.519084
1	12542.430556	3039.760417	3677.871528	3192.315972	766.267361	1094.920139

```python
plt.figure(figsize=(8,4), dpi=120)
sns.heatmap(cat_mean, annot=True)
```

✓ 0.7s

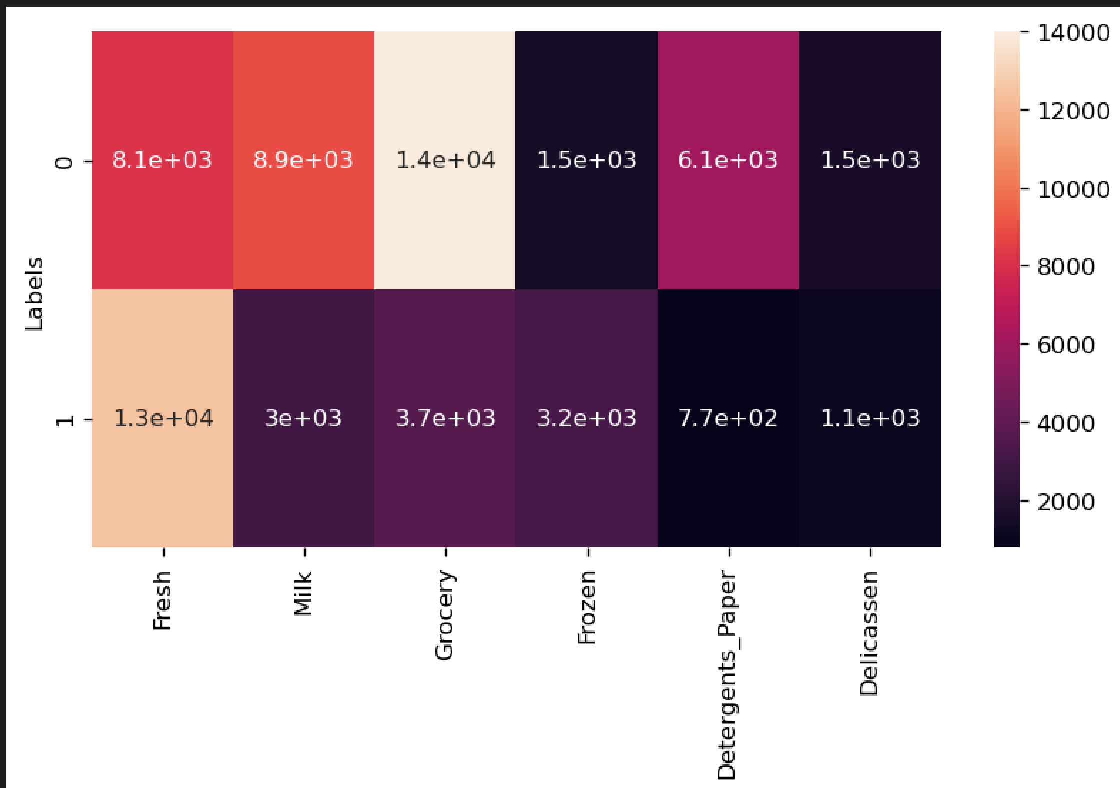
```
<AxesSubplot:ylabel='Labels'>
```



```python
plt.figure(figsize=(8,4), dpi=120)
sns.heatmap(cat_mean.loc[[0,1]], annot=True)
```

✓ 0.5s

```
<AxesSubplot:ylabel='Labels'>
```



```python
from sklearn.preprocessing import MinMaxScaler
```

✓ 0.5s

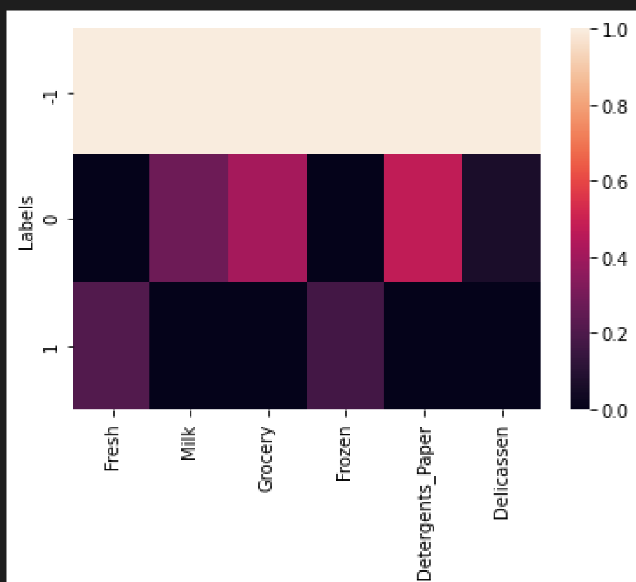
```python
scaler = MinMaxScaler()
data=scaler.fit_transform(cat_mean)
scaled_cat = pd.DataFrame(data, cat_mean.index, cat_mean.columns)
```

✓ 0.2s

```python
sns.heatmap(pd.DataFrame(data, cat_mean.index, cat_mean.columns))
```

✓ 0.4s

```
<AxesSubplot:ylabel='Labels'>
```



```
1  sns.heatmap(scaled_cat.loc[[0,1]],annot=True)
```

✓ 0.5s

```
<AxesSubplot:ylabel='Labels'>
```