Project: Unsupervised Learning

Objective: The main objective of the analysis is to predict the customer channel whether they are from hotel/restaurant/café or from retail business depending on their purchase history of different items. Here we focused on the clustering algorithm to build the model. The benefits of this analysis are to provide a better idea about the customers to enhance the facilities to the wholesale distributors.

Description of Dataset:

We will be using customer data from a Portuguese wholesale distributor for clustering. This data file is called Wholesale_Customers_Data.

It contains the following features:

Fresh: annual spending (m.u.) on fresh products

Milk: annual spending (m.u.) on milk products

Grocery: annual spending (m.u.) on grocery products

Frozen: annual spending (m.u.) on frozen products

Detergents_Paper: annual spending (m.u.) on detergents and paper products

Delicatessen: annual spending (m.u.) on delicatessen products

Channel: customer channel (1: hotel/restaurant/cafe or 2: retail)

Region: customer region (1: Lisbon, 2: Porto, 3: Other)

In this data, the values for all spending are given in an arbitrary unit (m.u. = monetary unit).

As a summary of the attributes, we can describe as follows,

```
data.describe()
```

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
count	440.000000	440.000000	440.000000	440.000000	440.000000	440.000000	440.000000	440.000000
mean	1.322727	2.543182	12000.297727	5796.265909	7951.277273	3071.931818	2881.493182	1524.870455
std	0.468052	0.774272	12647.328865	7380.377175	9503.162829	4854.673333	4767.854448	2820.105937
min	1.000000	1.000000	3.000000	55.000000	3.000000	25.000000	3.000000	3.000000
25%	1.000000	2.000000	3127.750000	1533.000000	2153.000000	742.250000	256.750000	408.250000
50%	1.000000	3.000000	8504.000000	3627.000000	4755.500000	1526.000000	816.500000	965.500000
75%	2.000000	3.000000	16933.750000	7190.250000	10655.750000	3554.250000	3922.000000	1820.250000
max	2.000000	3.000000	112151.000000	73498.000000	92780.000000	60869.000000	40827.000000	47943.000000

Data Cleaning: At the beginning, we need to check whether there are any null values or not. We observed that we have 440 items in each column and there is no null value.

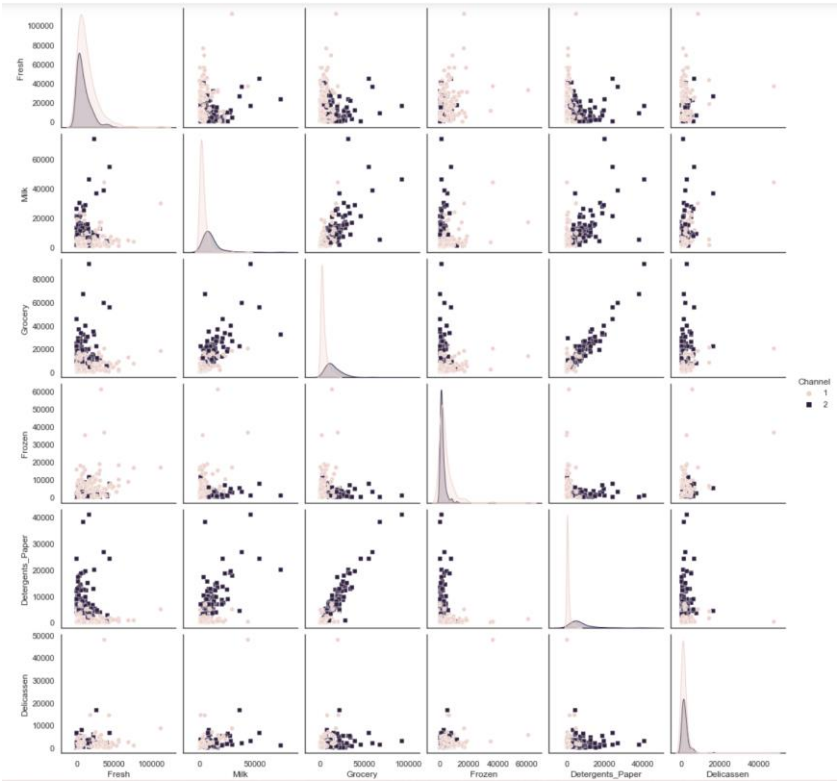
```
data.info()
```

```
<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
```

As we are focusing on the numeric values, we can drop Channel and Region columns. We can find the correlation matrix to observe the relationship among the features,

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
Fresh	0.000000	0.100510	-0.011854	0.345881	-0.101953	0.244690
Milk	0.100510	0.000000	0.728335	0.123994	0.661816	0.406368
Grocery	-0.011854	0.728335	0.000000	-0.040193	0.924641	0.205497
Frozen	0.345881	0.123994	-0.040193	0.000000	-0.131525	0.390947
Detergents_Paper	-0.101953	0.661816	0.924641	-0.131525	0.000000	0.069291
Delicassen	0.244690	0.406368	0.205497	0.390947	0.069291	0.000000

We get the following pair plots for the features,



The strong related features are as follows,

```
Fresh                          Frozen
Milk                         Grocery
Grocery              Detergents_Paper
Frozen                     Delicassen
Detergents_Paper             Grocery
Delicassen                      Milk
```

We checked the skewness of the features which we found that most of the features has skewness,

```
Delicassen          11.151586
Frozen               5.907986
Milk                 4.053755
Detergents_Paper     3.631851
Grocery              3.587429
Fresh                2.561323
Channel              0.760951
Region              -1.283627
```

Therefore, we will need to take care of the skewness, we build a pipeline to do the log transfer and scaling the data together as follows,

```
#Using log Transformer and Scaling data with pipeline

from sklearn.preprocessing import FunctionTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import MinMaxScaler
# The custom NumPy log transformer
log_transformer = FunctionTransformer(np.log1p)

# The pipeline
estimators = [('log1p', log_transformer), ('minmaxscale', MinMaxScaler())]
pipeline = Pipeline(estimators)

# Convert the data
#data_pipe = pipeline.fit_transform(data)

data[float_columns] = pipeline.fit_transform(data[float_columns])
```

After log transferring and scaling we got the dataset which is ready to evaluate with different clustering models:

```
data.head()
```

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
0	2	3	0.787076	0.717315	0.750579	0.272291	0.704748	0.619005
1	2	3	0.729947	0.719518	0.773997	0.543498	0.727298	0.649139
2	2	3	0.719687	0.704513	0.752184	0.583577	0.734394	0.807256
3	1	3	0.791565	0.426513	0.692596	0.709778	0.524784	0.649856
4	2	3	0.843653	0.636636	0.745685	0.646361	0.660499	0.763182

Different Clustering Models:

We chose three different clustering models such as KMeans, Agglomerative and Mean Shift.

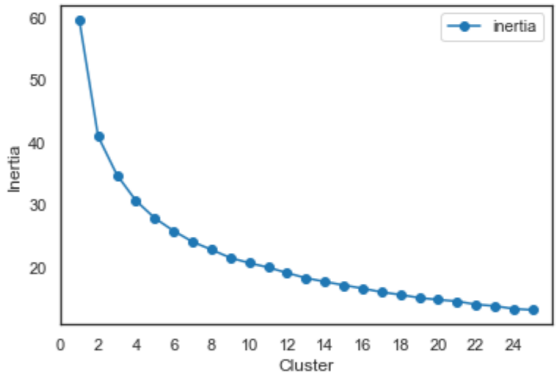
At first, we built the KMeans clustering algorithm and use the number of clusters is as 2.

```
from sklearn.cluster import KMeans

km = KMeans(n_clusters=2, random_state=42)
km = km.fit(data[float_columns])

data['kmeans'] = km.predict(data[float_columns])
```

As a result, we got the number of channel 1 and 2 for two clusters as follows,

		number
kmeans	Channel	
0	1	254
	2	6
1	1	44
	2	136

Now we can fit K-Means models with cluster values ranging from 0 to 25. For each model, we will store the number of clusters and the inertia value. After that, we will plot the cluster number vs inertia.



Then, we built agglomerative clustering model and saved the predicted values to a column of the main dataset,

```
# Now we can build an agglomerative clustering model

from sklearn.cluster import AgglomerativeClustering

ag = AgglomerativeClustering(n_clusters=2, linkage='ward', compute_full_tree=True)
ag = ag.fit(data[float_columns])
data['agglom'] = ag.fit_predict(data[float_columns])
```

Then, MeanShift model and saved the predicted values to a column of the main dataset,

```
# Now we can build Meanshift model
from sklearn.cluster import MeanShift
mean_shift = MeanShift(bandwidth=2)
mean_shift = mean_shift.fit(data[float_columns])
data['mean_shift'] = mean_shift.fit_predict(data[float_columns])
```

Comparison: For evaluating these three models we chose ROC_AUC score.

```
def get_avg_roc_10splits(estimator, X, y):
    roc_auc_list = []
    for train_index, test_index in sss.split(X, y):
        X_train, X_test = X.iloc[train_index], X.iloc[test_index]
        y_train, y_test = y.iloc[train_index], y.iloc[test_index]
        estimator.fit(X_train, y_train)
        y_predicted = estimator.predict(X_test)
        y_scored = estimator.predict_proba(X_test)[:, 1]
        roc_auc_list.append(roc_auc_score(y_test, y_scored))
    return np.mean(roc_auc_list)
```

The comparison result:

	Algorithm	Without Clustering	With Clustering
0	KMeans	0.9631	0.9657
1	Aglomerative	0.9636	0.9611
2	MeanShift	0.9630	0.9669

Discussion: We have evaluated three models with and without clustering and checked ROC_AUC score. From the comparison result it has been observed that MeanShift clustering model shows better result, therefore, we will choose this model to get accurate result. In future, it would be great if we could add more features when we are collecting data.