Load libraries

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

Read Dataset

In [3]:
```
data = pd.read_csv(r"C:\Users\hp\Downloads\Wholesale customers data.csv")
```

EDA

In [4]:
```
data.shape
```

Out[4]: (440, 8)

In [5]:
```
data.head()
```

Out[5]:

	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

In [6]:
```
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
```

In [7]:
```
data.describe()
```

Out[7]:

	Channel	Region	Fresh	Milk	Grocery	Frozen
count	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
std	0.468052	0.774272	12647.328865	7380.377175	9503.162829	4854.673333
min	1.000000	1.000000	3.000000	55.000000	3.000000	25.000000
25%	1.000000	2.000000	3127.750000	1533.000000	2153.000000	742.250000
50%	1.000000	3.000000	8504.000000	3627.000000	4755.500000	1526.000000
75%	2.000000	3.000000	16933.750000	7190.250000	10655.750000	3554.250000
max	2.000000	3.000000	112151.000000	73498.000000	92780.000000	60869.000000

In [8]:
```
data.isnull().sum()
```

Out[8]:
```
Channel             0
Region              0
Fresh               0
Milk                0
Grocery             0
Frozen              0
Detergents_Paper    0
Delicassen          0
dtype: int64
```

Declare feature vector and target variable

In [9]:
```
x = data.drop("Channel", axis=1)

y = data["Channel"]
```

In [10]:
```
x.head()
```

Out[10]:

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

In [11]:
```
y.head()
```

Out[11]:
```
0    2
1    2
2    2
3    1
4    2
Name: Channel, dtype: int64
```

In [12]:
```
y[y == 2] = 0

y[y == 1] = 1
```

In [13]:
```
y.head()
```

Out[13]:
```
0    0
1    0
2    0
3    1
4    0
Name: Channel, dtype: int64
```

In [14]:
```
import xgboost as xgb
```

In [15]:
```
data_dmatrix = xgb.DMatrix(data=x,label=y)
```

Split x and y into training and testing sxets

In [16]:
```
from sklearn.model_selection import train_test_split
```

In [17]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3,
```

Train test XGBoost classifier

In [18]:
```
from xgboost import XGBClassifier
```

In [19]:
```
params = {
            "objective":"binary:logistic",
            "max_depth": 4,
            "learning_rate": 1.0,
            "n_estimators":100
        }
```

In [20]:
```
# instantiate the clasifier
xgb_clf = XGBClassifier(**params)
```

In [21]:
```
# fit the classifier to the training data
xgb_clf.fit(x_train, y_train)
```

Out[21]:
```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=
None,
              gamma=None, gpu_id=None, grow_policy=None, importance_type
=None,
              interaction_constraints=None, learning_rate=1.0, max_bin=N
one,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=4, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=N
one,
              n_estimators=100, n_jobs=None, num_parallel_tree=None,
              predictor=None, random_state=None, ...)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [22]:
```
# we can view the parameters of the xgb trained model as follows

print(xgb_clf)
```

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=
None,
              gamma=None, gpu_id=None, grow_policy=None, importance_type
=None,
              interaction_constraints=None, learning_rate=1.0, max_bin=N
one,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=4, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=N
one,
              n_estimators=100, n_jobs=None, num_parallel_tree=None,
              predictor=None, random_state=None, ...)
```

Make predictions with XGBoost Classifier

In [23]:
```
y_pred = xgb_clf.predict(x_test)
```

check the accuracy score

In [24]:
```
from sklearn.metrics import accuracy_score
```

In [25]:
```
print("XGBoost model accuracy score: {0:0.4f}".format(accuracy_score(y_tes
```

```
XGBoost model accuracy score: 0.9167
```

k-fold Cross Validation using XGBoost

In [27]:
```
from xgboost import cv
```

In [28]:
```
params = {"objective":"binary:logistic",'colsample_bytree': 0.3,'learning_
                'max_depth': 5, 'alpha': 10}
```

In [29]:
```
xgb_cv = cv(dtrain=data_dmatrix, params=params, nfold=3,
                    num_boost_round=50, early_stopping_rounds=10, metrics=
```

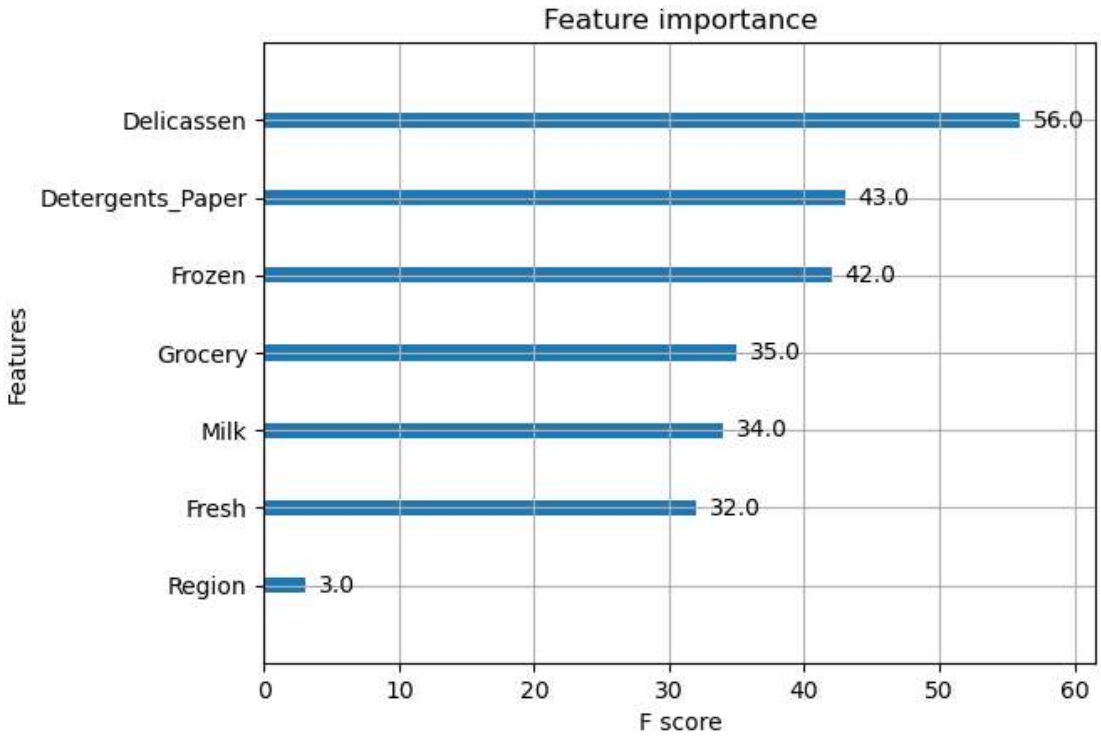
In [30]:
```
xgb_cv.head()
```

Out[30]:

	train-auc-mean	train-auc-std	test-auc-mean	test-auc-std
0	0.914999	0.009704	0.880965	0.021050
1	0.934374	0.013263	0.923562	0.022810
2	0.936252	0.013723	0.924433	0.025777
3	0.943878	0.009032	0.927152	0.022228
4	0.957881	0.008845	0.935191	0.016437

Feature importance with XGBoost

In [31]:
```
xgb.plot_importance(xgb_clf)
plt.figure(figsize = (16, 12))
plt.show()
```




```
<Figure size 1600x1200 with 0 Axes>
```

Result and conclusion

:- In this kernel, we implement XGBoost with Python and Scikit-Learn to classify the customers from two different channels as Horeca (Hotel/Retail/Café) customers or Retail channel (nominal) customers.

:- The y labels contain values as 1 and 2. We have converted them into 0 and 1 for further analysis.

:- We have trained the XGBoost classifier and found the accuracy score to be 91.67%.

:- We have performed k-fold cross-validation with XGBoost.

:- We have find the most important feature in XGBoost. We did it using the plot_importance() function in XGBoost that helps us to achieve this task