```
#Import XGBoost, Pandas, and sklearn for the function that we will use to calculate the accur
#The accuracy is required to understand how our model is performing.

from sklearn.ensemble import AdaBoostClassifier
from sklearn import datasets
from sklearn import metrics
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np
import xgboost as xgb
from sklearn.metrics import accuracy_score
```

```
#Import the wholesale customer dataset - 1 point

from google.colab import files
uploaded = files.upload()

data_set = pd.read_csv("wholesale-data.csv",header=None)
data = data_set.iloc[1:,:]
print(data)
```

Choose Files No file chosen

```
     0  1      2      3      4      5      6     7
1    2  3  12669   9656   7561    214   2674  1338
2    2  3   7057   9810   9568   1762   3293  1776
3    2  3   6353   8808   7684   2405   3516  7844
4    1  3  13265   1196   4221   6404    507  1788
5    2  3  22615   5410   7198   3915   1777  5185
..  .. ..    ...    ...    ...    ...    ...   ...
436  1  3  29703  12051  16027  13135    182  2204
437  1  3  39228   1431    764   4510     93  2346
438  2  3  14531  15488  30243    437  14841  1867
439  1  3  10290   1981   2232   1038    168  2125
440  1  3   2787   1698   2510     65    477    52
```

Saved successfully! ×

```
#Create training and test sets - 80:20 - 1 point
X = data.iloc[:,1:]
Y = data.iloc[:,0]

X = X.astype(int)
#Y = Y.astype(int)

Y = Y.map({'1':0, '2':1})
#print(Y)

x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2)
#print(x_train)
```

```
#print(x_train)
#print(x_test)
print(y_test.shape)
```

```
    (88,)
```

```
#Convert the pandas dataframe into a DMatrix, an internal data structure that is used by XGBo
# - 2 points

xgb_x_train = xgb.DMatrix(x_train, label=y_train)
xgb_x_test = xgb.DMatrix(x_test, label=y_test)
```

DMatrix is internal data structure used by XGBoost which is optimized for both memory efficiency and training speed.

```
#Specify the training parameters and train the model.

#xgb_clf = xgb.XGBClassifier()
#print(x_train.dtypes)
#xgb_clf.fit(x_train, y_train)

param = {
    'max_depth': 3,  # the maximum depth of each tree
    'eta': 0.3,  # the training step for each iteration
    'silent': 1,  # logging mode - quiet
    'objective': 'multi:softprob',
    'num_class':2}  # error evaluation for multiclass training  # the number of classes that
num_round = 20

bst = xgb.train(param, xgb_x_train, num_round)
```

```
#Predict the "Channel" values of the test set using the model that we just created. - 1 point
```

Saved successfully!

```
print(y_pred.shape)
best_preds = np.asarray([np.argmax(line) for line in y_pred])
print(best_preds.shape)
print(best_preds)
```

```
    (88, 2)
    (88,)
    [0 0 1 0 1 0 0 1 1 0 1 0 1 1 0 0 0 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0
     0 1 1 0 1 0 0 1 1 1 1 0 0 0 0 0 0 1 0 0 1 0 1 1 0 1 0 0 0 0 0 0 0 0 1 1 0
     0 1 0 0 1 0 1 0 1 0 1 1 1 0]
```

```
#Get the accuracy of the model that we have trained for the test dataset. - 1 point
```

```
#accuracy = accuracy_score(y_test, y_pred)
#print(accuracy)

from sklearn.metrics import precision_score

print (precision_score(y_test, best_preds, average='macro'))

     0.9226579520697167
```

✓ 0s completed at 9:29 AM ● ✕

Saved successfully! ✕