PES University, Bangalore

Established under Karnataka Act No. 16 of 2013

UE20CS312 - Data Analytics - Worksheet 4b

Course instructor: Gowri Srinivasa, Professor Dept. of CSE, PES University

Name: Ishita Bharadwaj

SRN: PES1UG20CS648

Collaborated with-

Hita - PES1UG20CS645

Jeffrey - PES1UG20CS651

Prerequisites

- Revise the following concepts
  - Boosting
    - AdaBoost
  - Apriori Algorithm
- Install the following software
  - pandas
  - numpy
  - sklearn
  - matplotlib
  - mlxtend

Task

In this notebook you will be exploring how to implement and utilize AdaBoost and the Apriori algorithm. For AdaBoost, this notebook utilizes the standard dataset from sklearn. For Apriori, please ensure that you have downloaded the `BreadBasket_DMS.csv` within the same working directory.

Points

- Problem 1: 4 points
- Problem 2: 3 points
- Problem 3: 3 points

Loading the Dataset

```
# Imports
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import pandas as pd
import numpy as np

# Load the wine dataset
data = datasets.load_wine(as_frame = True)

# Load x & y variables
X = data.data
y = data.target

# Split the dataset into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state =
```

Problem 1 (4 points)

Fit and evaluate the `AdaBoostClassifier` from sklearn.ensemble on the wine dataset. Use the `evaluate` model to print results.

Solution Steps:

1. From `sklearn.ensemble` import `AdaBoostClassifier`
2. Initialize the `AdaBoostClassifier` with n_estimators set to 30.
3. Use the `fit()` method and pass the train dataset.
4. Use the `evaluate(model, X_train, X_test, y_train, y_test)` method to print results.

For further reference: https://www.kaggle.com/code/faressayah/ensemble-ml-algorithms-bagging-boosting-voting/notebook

```
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

# evaluate method to print results after training a particular model
def evaluate(model, X_train, X_test, y_train, y_test):
    y_test_pred = model.predict(X_test)
```

```
    y_train_pred = model.predict(X_train)

    print("TRAINIG RESULTS: \n===============================")
    clf_report = pd.DataFrame(classification_report(y_train, y_train_pred, output_dict=Tru
    print(f"CONFUSION MATRIX:\n{confusion_matrix(y_train, y_train_pred)}")
    print(f"ACCURACY SCORE:\n{accuracy_score(y_train, y_train_pred):.4f}")
    print(f"CLASSIFICATION REPORT:\n{clf_report}")

    print("TESTING RESULTS: \n===============================")
    clf_report = pd.DataFrame(classification_report(y_test, y_test_pred, output_dict=True)
    print(f"CONFUSION MATRIX:\n{confusion_matrix(y_test, y_test_pred)}")
    print(f"ACCURACY SCORE:\n{accuracy_score(y_test, y_test_pred):.4f}")
    print(f"CLASSIFICATION REPORT:\n{clf_report}")
```

```
from sklearn.ensemble import AdaBoostClassifier

ada_boost_clf = AdaBoostClassifier(n_estimators=30)
ada_boost_clf.fit(X_train, y_train)
evaluate(ada_boost_clf, X_train, X_test, y_train, y_test)
```

```
TRAINIG RESULTS: 
===============================
CONFUSION MATRIX:
[[46  0  0]
 [ 0 54  0]
 [ 0  0 33]]
ACCURACY SCORE:
1.0000
CLASSIFICATION REPORT:
              0     1     2  accuracy  macro avg  weighted avg
precision   1.0   1.0   1.0       1.0        1.0           1.0
recall      1.0   1.0   1.0       1.0        1.0           1.0
f1-score    1.0   1.0   1.0       1.0        1.0           1.0
support    46.0  54.0  33.0       1.0      133.0         133.0
TESTING RESULTS: 
===============================
CONFUSION MATRIX:
[[12  1  0]
 [ 0 16  1]
 [ 0  2 13]]
ACCURACY SCORE:
0.9111
CLASSIFICATION REPORT:
                   0          1          2  accuracy  macro avg  weighted avg
precision   1.000000   0.842105   0.928571  0.911111   0.923559      0.916541
recall      0.923077   0.941176   0.866667  0.911111   0.910307      0.911111
f1-score    0.960000   0.888889   0.896552  0.911111   0.915147      0.911986
support    13.000000  17.000000  15.000000  0.911111  45.000000     45.000000
```

INFERENCES:

1. AdaBoost (Adaptive Boosting) is a very popular boosting technique that aims at combining multiple weak classifiers to build one strong classifier.
2. In the given data - wine's have classes 0, 1 and 2

3. From the above report we can see that the wine calssification on the test data has been done with an accuracy of 0.9111
4. The f1 score for wine of class 0,1,2 are 0.96,0.88 and 0.896 which is commendable
5. This report indicates that the adaboost with 30 n_estimators does a good job to classify the classes of wine

Problem 2 (3 points)

Retrieve the frequent itemsets using the `apriori` method from mlxtend.frequent_patterns. The code below extracts the basket_sets and this is provided as input for the apriori method.

Solution Steps:

1. Use the apriori algorithm, set min_support to 0.03 and use_colnames to True.
2. Print the output of the apriori method which provides the frequent_itemsets

For further reference: https://www.kaggle.com/code/victorcabral/bread-basket-analysis-apriori-association-rules/notebook (Cells 26 onwards)

```
# Install mlxtend
!pip install mlxtend
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/p
Requirement already satisfied: mlxtend in /usr/local/lib/python3.7/dist-packages (0.1
Requirement already satisfied: numpy>=1.10.4 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: scikit-learn>=0.18 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: matplotlib>=1.5.1 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: pandas>=0.17.1 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: scipy>=0.17 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-
```

```
# Imports
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
```

```
#Loading the dataset file
df = pd.read_csv('/content/BreadBasket_DMS.csv')
```

```
df['Quantity'] = 1
df.head(7)
```

	Date	Time	Transaction	Item	Quantity
0	2016-10-30	09:58:11	1	Bread	1
1	2016-10-30	10:05:34	2	Scandinavian	1
2	2016-10-30	10:05:34	2	Scandinavian	1
3	2016-10-30	10:07:57	3	Hot chocolate	1
4	2016-10-30	10:07:57	3	Jam	1
5	2016-10-30	10:07:57	3	Cookies	1
6	2016-10-30	10:08:41	4	Muffin	1

```
basket = df.groupby(['Transaction', 'Item'])['Quantity'].sum().unstack().fillna(0)
# There are a lot of zeros in the data but we also need to make sure any positive values a
# and anything less the 0 is set to 0. This step will complete the one hot encoding of the
def encode_units(x):
    if x <= 0:
        return 0
    if x >= 1:
        return 1

basket_sets = basket.applymap(encode_units)
basket_sets
```

Item	Adjustment	Afternoon with the	Alfajores	Argentina	Art	Bacon	Baguette	Bak

```
from mlxtend.frequent_patterns import apriori
```

```
frequence = apriori(basket_sets, min_support=0.03, use_colnames=True)
```

3	0	0	0	0	0	0	0

```
frequence = frequence.loc[frequence['support'] > 0.03]
frequence.sort_values(by = 'support' , ascending = False ,inplace = True)
frequence
```

	support	itemsets
4	0.475081	(Coffee)

INFERENCES

1. frequncy of itemsets is calculated based on their support values
2. We have sorted the most frequent item sets in descending order based on their support values
3. in the given dataset coffee has the highest support of 0.475 followed by bread which has a support of 0.32

9	0.061379	(Medialuna)

Double-click (or enter) to edit

19	0.054349	(Coffee, Cake)

Problem 3 (3 points)

Now use the `association_rules` method and pass the frequent_itemsets as input (achieved using problem 2). Use `.head()` to display the top five rules.

Solution Steps:

1. Use the `association_rules` method, set metric to lift and min_threshold to 1.
2. Print the top five rules using `.head()`.

For further reference: https://www.kaggle.com/code/victorcabral/bread-basket-analysis-apriori-association-rules/notebook (Cell 32 and 33)

23	0.037981	(Coffee, Sandwich)

```
from mlxtend.frequent_patterns import association_rules
rules = association_rules(frequence, metric="lift", min_threshold=1)
rules.sort_values('confidence', ascending = False, inplace=True)
rules.head()
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift
2	(Medialuna)	(Coffee)	0.061379	0.475081	0.034939	0.569231	1.198175
6	(Pastry)	(Coffee)	0.085510	0.475081	0.047214	0.552147	1.162216
5	(NONE)	(Coffee)	0.079005	0.475081	0.042073	0.532537	1.120938
9	(Sandwich)	(Coffee)	0.071346	0.475081	0.037981	0.532353	1.120551

INFERENCES:

1. We can see that all rules have a support greater than 0.03 so the confidence level decides the frequency of the itemsets (i.e. how frequent an item will be bought given that another item has already been bought)
2. Here we can see that Coffee is very frequently bought given that medialuna and pastry are already bought with confidences 0.569 and 0.552 respectively.