# Worksheet_4b-solution

## PES University, Bangalore¶

Established under Karnataka Act No. 16 of 2013

### UE20CS312 - Data Analytics - Worksheet 4a¶

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

Designed by Harshith Mohan Kumar, Dept. of CSE - harshithmohankumar@pesu.pes.edu

#### 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¶

In [1]:

```
# 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 = 2
```

### 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

In [2]:

```
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=True)
```

```
    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}")
```

In [3]:

```
# Solution

# 1. From `sklearn.ensemble` import `AdaBoostClassifier`
from sklearn.ensemble import AdaBoostClassifier

# 2. Initialize the `AdaBoostClassifier` with n_estimators set to 30.
ada_boost_clf = AdaBoostClassifier(n_estimators=30)
# 3. Use the `fit()` method and pass the train dataset.
ada_boost_clf.fit(X_train, y_train)
# 4. Use the `evaluate(model, X_train, X_test, y_train, y_test)` method to print results.
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 |

### 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)

In [4]:

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

```
Collecting mlxtend
  Downloading mlxtend-0.21.0-py2.py3-none-any.whl (1.3 MB)
     |                 | 1.3 MB 2.6 MB/s eta 0:00:01
Requirement already satisfied: setuptools in /usr/local/lib/python3.8/dist-packages (from ml
Requirement already satisfied: numpy>=1.16.2 in /usr/local/lib/python3.8/dist-packages (from
Requirement already satisfied: joblib>=0.13.2 in /usr/local/lib/python3.8/dist-packages (fro
Requirement already satisfied: scikit-learn>=1.0.2 in /usr/local/lib/python3.8/dist-packages
Requirement already satisfied: scipy>=1.2.1 in /usr/local/lib/python3.8/dist-packages (from
Requirement already satisfied: pandas>=0.24.2 in /usr/local/lib/python3.8/dist-packages (fro
Requirement already satisfied: matplotlib>=3.0.0 in /usr/local/lib/python3.8/dist-packages (
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.8/dist-package
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.8/dist-packa
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.8/dist-packages (from
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.8/dist-packages (fr
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.8/dist-packages (from
Requirement already satisfied: pyparsing>=2.2.1 in /usr/local/lib/python3.8/dist-packages (f
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.8/dist-packages (
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.8/dist-packages (
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.8/dist-packages (from
Requirement already satisfied: six>=1.5 in /usr/lib/python3/dist-packages (from python-dateu
Installing collected packages: mlxtend
Successfully installed mlxtend-0.21.0
```

```
WARNING: You are using pip version 20.2.4; however, version 22.2.2 is available.
You should consider upgrading via the '/usr/bin/python3 -m pip install --upgrade pip' comman
```

In [5]:

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

In [6]:

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

In [7]:

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

Out[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 |

In [8]:

```
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 are
# and anything less the 0 is set to 0. This step will complete the one hot encoding of the d
def encode_units(x):
    if x <= 0:
        return 0
    if x >= 1:
        return 1

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

Out[8]:

Item

Adjustment

Afternoon with the baker

Alfajores

Argentina Night

Art Tray

Bacon

Baguette

Bakewell

Bare Popcorn

Basket

...

The BART

The Nomad

Tiffin

Toast

Truffles

Tshirt

Valentine's card

Vegan Feast

Vegan mincepie

Victorian Sponge

Transaction

1

0

0

0

0

0

0

0

0

0

0

...

0

0

0

0

0

0

0

0

0

0

2

0

0

0

0

0

0

0

0

0

0

...

0

0

0

0

0

0

0

0

0

0

3

0

0

0

0

0

0

0

0

0

0

...

0

0

0

0

0

0

0

0

0

0

4

0

0

0

0

0

0

0

0

0

0

...

0

0

0

0

0

0

0

0

0

0

5

0

0

0

0

0

0

0

0

0

0

...

0

0

0

0

0

0

0

0

0

0

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

...

9680

0

0

0

0

0

0

0

0

0

0

...

0

0

0

0

0

0

0

0

0

0

9681

0

0

0

0

0

0

0

0

0

0

...

0

0

0

0

1

0

0

0

0

0

9682

0

0

0

0

0

0

0

0

0

0

...

0

0

0

0

0

0

0

0

0

0

9683

0

0

0

0

0

0

0

0

0

0

...

0

0

0

0

0

0

0

0

0

0

9684

0

0

0

0

0

0

0

0

0

0

...

0

0

0

0

0

0

0

0

0

0

9531 rows × 95 columns

In [9]:

```
# Solution

# 1. Use the apriori algorithm, set min_support to 0.03 and use_colnames to True.
frequent_itemsets = apriori(basket_sets,min_support=0.03,use_colnames=True)
# 2. Print the output of the apriori method which provides the frequent_itemsets
print(frequent_itemsets)
```

```
     support           itemsets
0   0.036093        (Alfajores)
1   0.324940            (Bread)
2   0.039765          (Brownie)
3   0.103137             (Cake)
4   0.475081           (Coffee)
5   0.054034          (Cookies)
6   0.038926       (Farm House)
7   0.057916    (Hot chocolate)
8   0.038296            (Juice)
9   0.061379        (Medialuna)
10  0.038191           (Muffin)
11  0.079005             (NONE)
12  0.085510           (Pastry)
```

```
13  0.071346           (Sandwich)
14  0.034309              (Scone)
15  0.034204               (Soup)
16  0.141643                (Tea)
17  0.033365              (Toast)
18  0.089393      (Coffee, Bread)
19  0.054349       (Coffee, Cake)
20  0.034939  (Coffee, Medialuna)
21  0.042073       (NONE, Coffee)
22  0.047214     (Pastry, Coffee)
23  0.037981   (Sandwich, Coffee)
24  0.049523        (Tea, Coffee)
```

```
/usr/local/lib/python3.8/dist-packages/mlxtend/frequent_patterns/fpcommon.py:111: Deprecatio
  warnings.warn(
```

**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)

In [10]:

```
# Solution

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

Out[10]:

| | antecedents | consequents | antecedent support | consequent support | support | confidence | lift |
|---|---|---|---|---|---|---|---|
| 0 | (Coffee) | (Cake) | 0.475081 | 0.103137 | 0.054349 | 0.114399 | 1.10919 |
| 1 | (Cake) | (Coffee) | 0.103137 | 0.475081 | 0.054349 | 0.526958 | 1.10919 |
| 2 | (Coffee) | (Medialuna) | 0.475081 | 0.061379 | 0.034939 | 0.073542 | 1.19817 |
| 3 | (Medialuna) | (Coffee) | 0.061379 | 0.475081 | 0.034939 | 0.569231 | 1.19817 |
| 4 | (NONE) | (Coffee) | 0.079005 | 0.475081 | 0.042073 | 0.532537 | 1.12093 |