DATA ANALYTICS- WORKSHEET 4B

GAURAV MAHAJAN

PES1UG20CS150

```
!pip install mlxtend
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/
Requirement already satisfied: mlxtend in /usr/local/lib/python3.7/dist-packages (0.
Requirement already satisfied: matplotlib>=1.5.1 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: scipy>=0.17 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: numpy>=1.10.4 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: pandas>=0.17.1 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: scikit-learn>=0.18 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/loca
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist
```

Loading the Dataset¶

```
# Imports
import matplotlib.pyplot as plt
from sklearn.ensemble import AdaBoostClassifier
from sklearn import datasets
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn import metrics
from sklearn.metrics import confusion_matrix
import pandas as pd
import numpy as np
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules

# 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

```
#calling the adaboost function
adb = AdaBoostClassifier(n_estimators=30,learning_rate=1)
```

```
#fitting the values
classifier_model = adb.fit(X_train, y_train)
```

```
#predicted values
y_pred = classifier_model.predict(X_test)
y_pred
```

```
array([1, 2, 1, 2, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 2, 2, 1, 1,
       2, 1, 1, 2, 0, 2, 2, 2, 1, 1, 2, 1, 0, 2, 1, 0, 2, 1, 2, 1, 0, 0,
       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("TRAINING 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}")
```

Problem 2 (3 points)¶

Retrieve the frequent itemsets using the apriori method from mlx- tend.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 fre- quent_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/
Requirement already satisfied: mlxtend in /usr/local/lib/python3.7/dist-packages (0.
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: scikit-learn>=0.18 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: numpy>=1.10.4 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: matplotlib>=1.5.1 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: scipy>=0.17 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: pandas>=0.17.1 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/loca
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist
```

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

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

1 to 7 of 7 entries Filter

index	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

```
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
```

entina Night	Art Tray	Bacon	Baguette	Bakewell	Bare Popcorn	Basket	...	The BART	The Nomad	Tiffin	To
0	0	0	0	0	0	0	...	0	0	0	
0	0	0	0	0	0	0	...	0	0	0	
0	0	0	0	0	0	0	...	0	0	0	
0	0	0	0	0	0	0	...	0	0	0	
0	0	0	0	0	0	0	...	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	
0	0	0	0	0	0	0	...	0	0	0	
0	0	0	0	0	0	0	...	0	0	0	
0	0	0	0	0	0	0	...	0	0	0	
0	0	0	0	0	0	0	...	0	0	0	
0	0	0	0	0	0	0	...	0	0	0	

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

1 to 25 of 25 entries Filter

index	support	itemsets
0	0.0360927499737698	frozenset({'Alfajores'})
1	0.3249396705487357	frozenset({'Bread'})
2	0.039764977442031266	frozenset({'Brownie'})
3	0.10313713146574337	frozenset({'Cake'})
4	0.4750813136082258	frozenset({'Coffee'})
5	0.054034204175847235	frozenset({'Cookies'})
6	0.0389256111635715	frozenset({'Farm House'})
7	0.05791627321372364	frozenset({'Hot chocolate'})
8	0.03829608645472668	frozenset({'Juice'})
9	0.061378659112370164	frozenset({'Medialuna'})
10	0.03819116566991921	frozenset({'Muffin'})
11	0.07900535096002519	frozenset({'NONE'})
12	0.08551043961808834	frozenset({'Pastry'})
13	0.07134613366907984	frozenset({'Sandwich'})
14	0.034309096632042804	frozenset({'Scone'})
15	0.03420417584723534	frozenset({'Soup'})
16	0.141643059490085	frozenset({'Tea'})
17	0.03336480956877558	frozenset({'Toast'})
18	0.08939250865596475	frozenset({'Bread', 'Coffee'})
19	0.054348966530269646	frozenset({'Cake', 'Coffee'})
20	0.03493862134088763	frozenset({'Medialuna', 'Coffee'})
21	0.042073234707795615	frozenset({'Coffee', 'NONE'})
22	0.047214353163361665	frozenset({'Pastry', 'Coffee'})
23	0.03798132410030427	frozenset({'Sandwich', 'Coffee'})
24	0.04952261042912601	frozenset({'Tea', 'Coffee'})

Show 25 per page

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)

```
rules = association_rules(repetetive_items, metric="lift", min_threshold=1)
rules.head()
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift
0	(Cake)	(Coffee)	0.103137	0.475081	0.054349	0.526958	1.109196
1	(Coffee)	(Cake)	0.475081	0.103137	0.054349	0.114399	1.109196
2	(Medialuna)	(Coffee)	0.061379	0.475081	0.034939	0.569231	1.198175
3	(Coffee)	(Medialuna)	0.475081	0.061379	0.034939	0.073542	1.198175