Exercise 7 - Association Rules

First name: Brian

Last name: Schweigler

Matriculation number: 16-102-071

(1) Problem: Decide whether to wait for a table at a restaurant, based on various attributes.

In [1]:

```
%load_ext autoreload
%autoreload 2
%matplotlib inline

import pandas as pd
import numpy as np
from sklearn import preprocessing
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn import svm
from mlxtend.frequent_patterns import apriori, association_rules
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori

df = pd.read_csv("data/restaurant.csv")
pd.set_option('display.max_colwidth', None)
le = preprocessing.LabelEncoder()
x_train = df[["choice", "bar", "day", "hungry", "patron", "price",
              "rain", "booking", "type"]]
x_train = pd.DataFrame(columns=x_train.columns,
                       data=le.fit_transform(x_train.values.flatten())
                       .reshape(x_train.shape))
x_train["time"] = df["time"]

y = le.fit(df["wait"])
y = le.transform(df["wait"])

Xd_train, Xd_test, y_train, y_test = train_test_split(
    x_train, y, test_size=0.4)

print(Xd_train)
print("y_train = ", y_train, "\n")

print(Xd_test)
print("y_test = ", y_test)
```

```
    choice  bar  day  hungry  patron  price  rain  booking  type  time
8        3    4    4       3       6      0     4        3    10    60
0        4    3    3       4       9      2     3        4     5     0
2        4    4    3       3       9      0     3        3    10     0
9        4    4    4       4       6      2     3        4     7    20
7        3    3    3       4       9      1     4        4    11     0
10       3    3    3       3       8      0     3        3    11     0
6        3    4    3       3       8      0     4        3    10    20
y_train =  [0 1 1 0 1 0 0]

    choice  bar  day  hungry  patron  price  rain  booking  type  time
5        3    4    3       4       9      1     4        3     7     0
11       4    4    4       4       6      0     3        3    10    40
4        4    3    4       3       6      2     3        4     5    60
3        4    3    4       4       6      0     3        3    11    20
1        4    3    3       4       6      0     3        3    11    40
y_test =  [1 1 0 1 0]
```

In [2]:

```
# Create a svm classifier using one kernel (linear, polynomial, and radial basis)
clf = svm.SVC(kernel='linear')
clf.fit(Xd_train, y_train)

y_pred = clf.predict(Xd_test)
NB_Accuracy = accuracy_score(y_test, y_pred)

print("y_test with linear kernel = ", y_test)
print("y_pred with linear kernel = ", y_pred, "\n")
print("NB_Accuracy with linear kernel = ", NB_Accuracy, "\n")
print("Confusion Matrix with linear kernel \n",
      confusion_matrix(y_test, y_pred))
```

```
y_test with linear kernel =  [1 1 0 1 0]
y_pred with linear kernel =  [1 0 0 0 0]

NB_Accuracy with linear kernel =  0.6
```

```
Confusion Matrix with linear kernel
 [[2 0]
 [2 1]]
```

Co-occurence matrix

In [3]:

```
cooc_mat = x_train.T.dot(x_train)
cooc_mat
```

Out[3]:

	choice	bar	day	hungry	patron	price	rain	booking	type	time
choice	157	150	148	155	312	30	141	144	383	960
bar	150	150	144	150	308	27	141	139	378	920
day	148	144	143	147	294	28	136	137	367	980
hungry	155	150	147	157	315	30	143	144	386	900
patron	312	308	294	315	668	60	296	294	789	1600
price	30	27	28	30	60	14	26	31	52	160
rain	141	141	136	143	296	26	136	133	362	860
booking	144	139	137	144	294	31	133	136	352	860
type	383	378	367	386	789	52	362	352	1032	2300
time	960	920	980	900	1600	160	860	860	2300	11600

Calculate support, confidence, completeness, lift, and leverage for the following rules. Build the co-occurrence matrix. (again?)

In [4]:

```
te = TransactionEncoder()
te_ary = te.fit(x_train).transform(x_train)
df_new = pd.DataFrame(te_ary, columns=te.columns_)

frequent_itemsets = apriori(df_new,
                            min_support=0.001, use_colnames=True)
frequent_itemsets
```

Out[4]:

	support	itemsets
0	0.333333	(a)
1	0.166667	(b)
2	0.166667	(c)
3	0.083333	(d)
4	0.333333	(e)
...	...	...
256	0.083333	(y, h, n, r, u)
257	0.083333	(t, o, n, p, r)
258	0.083333	(a, t, o, n, p, r)
259	0.083333	(b, k, i, o, n, g)
260	0.083333	(y, h, u, n, r, g)

261 rows × 2 columns

In [5]:

```
association_rules(frequent_itemsets,
                  metric="confidence", min_threshold=0.001)
```

Out[5]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(a)	(b)	0.333333	0.166667	0.083333	0.25	1.5	0.027778	1.111111
1	(b)	(a)	0.166667	0.333333	0.083333	0.50	1.5	0.027778	1.333333
2	(a)	(d)	0.333333	0.083333	0.083333	0.25	3.0	0.055556	1.222222
3	(d)	(a)	0.083333	0.333333	0.083333	1.00	3.0	0.055556	inf
4	(a)	(i)	0.333333	0.416667	0.083333	0.25	0.6	-0.055556	0.777778
...	...	...	...	...	...	...	...	...	...

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
2287	(h)	(y, u, n, r, g)	0.166667	0.083333	0.083333	0.50	6.0	0.069444	1.833333
2288	(u)	(y, h, n, r, g)	0.083333	0.083333	0.083333	1.00	12.0	0.076389	inf
2289	(n)	(y, h, u, r, g)	0.333333	0.083333	0.083333	0.25	3.0	0.055556	1.222222
2290	(r)	(y, h, u, n, g)	0.416667	0.083333	0.083333	0.20	2.4	0.048611	1.145833
2291	(g)	(y, h, n, r, u)	0.166667	0.083333	0.083333	0.50	6.0	0.069444	1.833333

2292 rows × 9 columns

As I did not manage to enter the rules specifically, I decided to just have the min_support and min_threshold values as low as possible, to essentially just include all rules possible.

This is bound to include the 10 rules mentioned.

Explain these measures

Support: The fraction of the total number of data entries in which the variable occurs.

Confidence: The conditional probability of occurrence of consequent given the antecedent.

Completeness: The ratio the overall transactions where the predicted item appear that is covered by the rule.

Lift: Controls for the support (frequency) of consequent while calculating the conditional probability of occurrence of Y given X.

Leverage: Leverage measures the difference of XX and YY appearing together in the data set and what would be expected if XX and YY were statistically dependent.

Use the Apriori algorithm to find frequent item sets. We are only interested in item sets having a support value of at least 50%.

In [6]:

```
# Inspiration: https://www.kaggle.com/code/sangwookchn/association-rule-learning-with-scikit-learn
from apyori import apriori

rules = apriori(x_train[['choice', 'bar', 'day', 'hungry', 'patron', 'rain', 'booking', 'type']],
                min_support=0.5,
                min_confidence=0.01)
results = list(rules)
results = pd.DataFrame(results)
results.head(5)
```

Out[6]:

	items	support	ordered_statistics
0	(a)	0.5	[((), (a), 0.5, 1.0)]
1	(n)	0.5	[((), (n), 0.5, 1.0)]
2	(r)	0.5	[((), (r), 0.5, 1.0)]

Alternative variant, where support is at max 0.25 for some reason:

In [7]:

```
rules = association_rules(frequent_itemsets,
                          metric="support", min_threshold=0.25)
rules
```

Out[7]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(a)	(r)	0.333333	0.416667	0.25	0.75	1.8	0.111111	2.333333
1	(r)	(a)	0.416667	0.333333	0.25	0.60	1.8	0.111111	1.666667
2	(e)	(i)	0.333333	0.416667	0.25	0.75	1.8	0.111111	2.333333
3	(i)	(e)	0.416667	0.333333	0.25	0.60	1.8	0.111111	1.666667
4	(n)	(r)	0.333333	0.416667	0.25	0.75	1.8	0.111111	2.333333
5	(r)	(n)	0.416667	0.333333	0.25	0.60	1.8	0.111111	1.666667