UNIVERSIDAD PERUANA DE CIENCIAS APLICADAS
CIENCIAS DE LA COMPUTACIÓN

MACHINE LEARNING
Laboratorio Apriori
(Primer Semestre del 2019)

Objetivos de aprendizaje:

- Generar reglas de asociación usando el algoritmo Apriori.

---

1. Actividad en Weka

1.1. weather.nominal (15 minutos)

Genere reglas de asociación a partir del conjunto de datos `weather.nominal.arff`. Para esto siga los siguientes pasos:

- Seleccione el conjunto de datos `weather.nominal.arff` en la pestaña `Preprocess`.
- En la pestaña `Associate` seleccione el algoritmo `Apriori`.
- Presione el botón `Start`.

Preguntas de discusión

- ¿Qué reglas ha generado? Descríbalas.
- Modifique el soporte mínimo para 0.25. Use el parámetro `lowerBoundMinSupport` del algoritmo.
- ¿Qué cambios puede verificar?

1.2. vote (15 minutos)

Genere reglas de asociación a partir del conjunto de datos `vote.arff`.

Preguntas de discusión

- ¿Qué reglas ha generado? Descríbalas.
- Modifique el soporte mínimo para 0.5. Use el parámetro `lowerBoundMinSupport` del algoritmo.
- ¿Qué cambios puede verificar?

1.3. supermarket (15 minutos)

Genere reglas de asociación a partir del conjunto de datos `supermarket.arff`.

Preguntas de discusión

- ¿Qué reglas ha generado? Descríbalas.

2. Actividad en Python

2.1. Algoritmo apriori (45 minutos)

Ejecute el siguiente código y analice qué sucede en cada línea de código.

```
import numpy as np
import scipy as sc
from pandas import Series, DataFrame
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib as mpl
import seaborn as sns
from collections import OrderedDict
from fractions import Fraction

#se crea el dataset
transaction_df=pd.DataFrame({'Beer': [1,0,1,0,1,0,0,1],
                             'Coke': [0,1,1,0,1,0,0,1],
                             'Pepsi': [1,0,0,1,0,0,1,0],
                             'Milk': [0,1,0,1,1,1,0,1],
                             'Juice': [0,0,1,0,0,1,1,1]})
print(transaction_df)

#se calcula las ocurrencias (support) para cada producto en todas las transacciones
product_support_dict = {}
for column in transaction_df:
        product_support_dict[column] = sum(transaction_df[column]>0)

#visualizamos el soporte
pd.Series(product_support_dict).plot(kind='bar')
plt.show()
```

```
#algoritmo apriori
#se toma el dataframe como matriz
transaction_matrix = transaction_df.as_matrix()
print(transaction_matrix)

rows, columns = transaction_matrix.shape

#inicializamos una nueva matriz
frequent_items_matrix=np.zeros((columns, columns))
print(frequent_item_matrix)

#comparamos cada producto con el otro
for this_column in range(0, columns-1):
    for next_column in range (this_column+1, columns):
        product_vector = transaction_matrix[:, this_column]*transaction_matrix[:,
            next_column]
        count_matches = sum(product_vector>0)
        frequent_items_matrix[this_column,next_column]=count_matches

print(frequent_item_matrix)
plt.matshow(frequent_items_matrix)
plt.show()
```

```
#item mas frecuentes
frequent_items_df = pd.DataFrame(frequent_items_matrix,
                                 columns = transaction_df.columns.values,
                                 index = transaction_df.columns.values)
sns.heatmap(frequent_items_df)
plt.show()
```

```
#extraccion de items frecuentes
#se extraen pares de with la frecuencia minima pasada como parametro

```

```
product_name = transaction_df.columns.values

def extract_pairs(treshold):
        output={}
        #seleccionamos los indices mayores o iguales a treshold
        matrix_coord_list = np.where(frequent_items_matrix>=treshold)
        #tomamos los valores
        row_coords = matrix_coord_list[0]
        column_coords = matrix_coord_list[1]
        #generamos los pares
        for index, value in enumerate(row_coords):
                row = row_coords[index]
                colum = column_coords[index]
                #obtenemos el nombre de los productos
                first_product = product_name[row]
                second_product = product_name[colum]
                #obtenemos las ocurrencias
                matches = frequent_items_matrix[row][colum]
                #ponemos el par en el diccionario
                output[first_product+"-"+second_product]=matches
        #retornamos la lista ordenada
        sorted_output=OrderedDict(sorted(output.items(), key=lambda x:x[1]))
        return sorted_output

reglas=extract_pairs(1)
print(reglas)
pd.Series(reglas).plot(kind='barh')
plt.show()
```

Monterrico, 18 de junio de 2019.