Agrupamiento por el método KNN

Presenta: Juliho Castillo Colmenares

Antes de empezar

El objetivo de esta actividad es que realices análisis de agrupación por "K vecinos más cercanos" y de patrones de consumo por "Market basket" a diversas bases de datos de manera que puedas generar conclusiones adecuadas.

Paso a paso:

Problema 1:

- Extraiga del internet la base de datos "wine-clustering.csv" de la siguiente liga y agréguela a un DataFrame en Python.

Alcohol	Malic_Acid	Ash	Ash_Alcanity	Magnesium	Total_Phenol	Flavanoids	Nonflavanoid	Proanthocyar	Color_Intensi	Hue	OD280	Proline
14.23	1.71	2.43	15.6	127	2.8	3.06	0.28	2.29	5.64	1.04	3.92	1065
13.2	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.4	1050
13.16	2.36	2.67	18.6	101	2.8	3.24	0.3	2.81	5.68	1.03	3.17	1185
14.37	1.95	2.5	16.8	113	3.85	3.49	0.24	2.18	7.8	0.86	3.45	1480
13.24	2.59	2.87	21	118	2.8	2.69	0.39	1.82	4.32	1.04	2.93	735
14.2	1.76	2.45	15.2	112	3.27	3.39	0.34	1.97	6.75	1.05	2.85	1450
14.39	1.87	2.45	14.6	96	2.5	2.52	0.3	1.98	5.25	1.02	3.58	1290
14.06	2.15	2.61	17.6	121	2.6	2.51	0.31	1.25	5.05	1.06	3.58	1295
14.83	1.64	2.17	14	97	2.8	2.98	0.29	1.98	5.2	1.08	2.85	1045
13.86	1.35	2.27	16	98	2.98	3.15	0.22	1.85	7.22	1.01	3.55	1045
14.1	2.16	2.3	18	105	2.95	3.32	0.22	2.38	5.75	1.25	3.17	1510
14.12	1.48	2.32	16.8	95	2.2	2.43	0.26	1.57	5	1.17	2.82	1280
13.75	1.73	2.41	16	89	2.6	2.76	0.29	1.81	5.6	1.15	2.9	1320
14.75	1.73	2.39	11.4	91	3.1	3.69	0.43	2.81	5.4	1.25	2.73	1150

- Mediante el algoritmo de K vecinos más cercanos determine el valor promedio de alcohol que tendrían los 5 vinos más parecidos a aquel con las características siguientes:

Alcohol	Malic_Acid	Ash	Ash_Alcanity	Magnesium	Total_Phenols	Flavanoids	Nonflavanoid_Phenols	Proanthocyanins	Color_Intensity	Hue	OD280	Proline
14	2	2.5	16	115	3	2.5	0.4	2	9	1	3.5	800

- Asegúrese de detallar las concentraciones de alcohol de cada elemento que haya sido utilizado para obtener dicho valor promedio.

In [ ]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
```

```
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import accuracy_score
```

In [ ]:
```
dataframe = pd.read_csv("wine-clustering.csv")
dataframe
```

Out[ ]:

	Alcohol	Malic_Acid	Ash	Ash_Alcanity	Magnesium	Total_Phenols	Flavanoids	Nonflavanoid_Phenols	Proanthocyanins	Color_Intensity	Hue
0	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.64	1.04
1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05
2	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03
3	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86
4	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04
...	...	...	...	...	...	...	...	...	...	...	...
173	13.71	5.65	2.45	20.5	95	1.68	0.61	0.52	1.06	7.70	0.64
174	13.40	3.91	2.48	23.0	102	1.80	0.75	0.43	1.41	7.30	0.70
175	13.27	4.28	2.26	20.0	120	1.59	0.69	0.43	1.35	10.20	0.59
176	13.17	2.59	2.37	20.0	120	1.65	0.68	0.53	1.46	9.30	0.60
177	14.13	4.10	2.74	24.5	96	2.05	0.76	0.56	1.35	9.20	0.61

178 rows × 13 columns

In [ ]:
```
# Diccionario para almacenar los datos
datos_vino = {
    "Alcohol": 14,
    "Malic_Acid": 2,
    "Ash": 2.5,
    "Ash_Alcanity": 16,
    "Magnesium": 115,
    "Total_Phenols": 3,
    "Flavanoids": 2.5,
    "Nonflavanoid_Phenols": 0.4,
    "Proanthocyanins": 2,
    "Color_Intensity": 9,
    "Hue": 1,
    "OD280": 35,
```

```
    "Proline": 800
}

print(datos_vino)
```

```
{'Alcohol': 14, 'Malic_Acid': 2, 'Ash': 2.5, 'Ash_Alcanity': 16, 'Magnesium': 115, 'Total_Phenols': 3, 'Flavanoids': 2.5,
'Nonflavanoid_Phenols': 0.4, 'Proanthocyanins': 2, 'Color_Intensity': 9, 'Hue': 1, 'OD280': 35, 'Proline': 800}
```

In [ ]:
```
# Convertir el diccionario a un DataFrame
sample = pd.DataFrame([datos_vino])
sample
```

Out[ ]:

	Alcohol	Malic_Acid	Ash	Ash_Alcanity	Magnesium	Total_Phenols	Flavanoids	Nonflavanoid_Phenols	Proanthocyanins	Color_Intensity	Hue
0	14	2	2.5	16	115	3	2.5	0.4	2	9	1

In [ ]:
```
X_sample = sample.drop(columns=["Alcohol"])
y_sample = sample["Alcohol"]
```

In [ ]:
```
# Separar las características y la variable objetivo
X = dataframe.drop(columns=['Alcohol'])
y = dataframe['Alcohol']
```

In [ ]:
```
# Dividir el conjunto de datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [ ]:
```
# Entrenar el modelo KNN para regresión
knn = KNeighborsRegressor(n_neighbors=5)
knn.fit(X_train, y_train)
```

Out[ ]:

KNeighborsRegressor

```
KNeighborsRegressor()
```

In [ ]:
```
# Encontrar los 5 vinos más parecidos
distances, indices = knn.kneighbors(X_sample, n_neighbors=5)
```

In [ ]:
```
distances
```

Out[ ]:
```
array([[32.21784909, 38.88398771, 45.72354864, 45.86212817, 48.95697397]])
```

In [ ]:
```
indices
```

Out[ ]:
```
array([[ 47, 136, 117,  49, 113]])
```

In [ ]:
```
# Obtener los valores de alcohol de los 5 vinos más parecidos
alcohol_values = y.iloc[indices[0]]
alcohol_values
```

Out[ ]:
```
47     13.90
136    12.25
117    12.42
49     13.94
113    11.41
Name: Alcohol, dtype: float64
```

In [ ]:
```
# Calcular el valor promedio de alcohol
promedio_alcohol = alcohol_values.mean()
print(f'El valor promedio de alcohol de los 5 vinos más parecidos es: {promedio_alcohol}')
```

```
El valor promedio de alcohol de los 5 vinos más parecidos es: 12.784
```

Problema 2:

- Considere la lista de 11 compras desglosada de la manera siguiente:

```
my_basket=[['bread','butter','wine','bananas','coffee','carrots'],
['tomatoes','onions','cheese','milk','potatoes'],
['beer','chips','asparagus','salsa','milk','apples'],
['olive oil','bread','butter','tomatoes','steak','carrots'],
['tomatoes','onions','chips','wine','ketchup','orange juice'],
['bread','butter','beer','chips','milk'],
['butter','tomatoes','carrots','coffee','sugar'],
['tomatoes','onions','cheese','milk','potatoes'],
['bread','butter','ketchup','coffee','chicken wings'],
['butter','beer','chips','asparagus','apples'],
['tomatoes','onion','beer','chips','milk','coffee']]
```

- De acuerdo a la información anterior, se entiende que el primer ticket de compra de la lista consideró la adquisición de los siguientes productos:
  - bread
  - butter
  - wine
  - bananas
  - coffee
  - carrots
- Modifica el código explicado en este módulo para que sea capaz de realizar un análisis mediante el algoritmo “Market basket” a dicha lista de compras con la finalidad de detectar patrones de consumo. Realice a continuación las conclusiones que sean pertinentes sobre sus resultados.
- Nota importante: tu programa debe ser capaz de correr cualquier lista de compras que venga en el mismo formato que el utilizado para esta actividad.

In [ ]:
```
# Importar bibliotecas necesarias
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules
import warnings
warnings.filterwarnings('ignore')
```

```python
# Convertir la lista en un DataFrame
my_basket = [
    ['bread', 'butter', 'wine', 'bananas', 'coffee', 'carrots'],
    ['tomatoes', 'onions', 'cheese', 'milk', 'potatoes'],
    ['beer', 'chips', 'asparagus', 'salsa', 'milk', 'apples'],
    ['olive oil', 'bread', 'butter', 'tomatoes', 'steak', 'carrots'],
    ['tomatoes', 'onions', 'chips', 'wine', 'ketchup', 'orange juice'],
    ['bread', 'butter', 'beer', 'chips', 'milk'],
    ['butter', 'tomatoes', 'carrots', 'coffee', 'sugar'],
    ['tomatoes', 'onions', 'cheese', 'milk', 'potatoes'],
    ['bread', 'butter', 'ketchup', 'coffee', 'chicken wings'],
    ['butter', 'beer', 'chips', 'asparagus', 'apples'],
    ['tomatoes', 'onion', 'beer', 'chips', 'milk', 'coffee']
]

# Crear un DataFrame con las transacciones
df = pd.DataFrame(my_basket)

# Preprocesar datos
# Convertir el DataFrame en una matriz de transacciones
basket = df.stack().reset_index().pivot_table(index='level_0', columns=0, aggfunc='size', fill_value=0)
basket = basket.applymap(lambda x: 1 if x > 0 else 0)

# Aplicar algoritmo Apriori
frequent_itemsets = apriori(basket, min_support=0.2, use_colnames=True)

# Generar reglas de asociación
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)

# Analizar resultados
# Filtrar reglas con un alto nivel de confianza
rules = rules[(rules['confidence'] > 0.5) & (rules['lift'] > 1)]
rules = pd.DataFrame(rules)

# Mostrar las reglas
rules
```

Out[ ]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	zhangs_metric
0	(chips)	(beer)	0.454545	0.363636	0.363636	0.800000	2.200000	0.198347	3.181818	1.000000
1	(beer)	(chips)	0.363636	0.454545	0.363636	1.000000	2.200000	0.198347	inf	0.857143
2	(milk)	(beer)	0.454545	0.363636	0.272727	0.600000	1.650000	0.107438	1.590909	0.722222
3	(beer)	(milk)	0.363636	0.454545	0.272727	0.750000	1.650000	0.107438	2.181818	0.619048
4	(bread)	(butter)	0.363636	0.545455	0.363636	1.000000	1.833333	0.165289	inf	0.714286
5	(butter)	(bread)	0.545455	0.363636	0.363636	0.666667	1.833333	0.165289	1.909091	1.000000
7	(carrots)	(butter)	0.272727	0.545455	0.272727	1.000000	1.833333	0.123967	inf	0.625000
8	(coffee)	(butter)	0.363636	0.545455	0.272727	0.750000	1.375000	0.074380	1.818182	0.428571
10	(chips)	(milk)	0.454545	0.454545	0.272727	0.600000	1.320000	0.066116	1.363636	0.444444
11	(milk)	(chips)	0.454545	0.454545	0.272727	0.600000	1.320000	0.066116	1.363636	0.444444
13	(milk)	(tomatoes)	0.454545	0.545455	0.272727	0.600000	1.100000	0.024793	1.136364	0.166667
14	(onions)	(tomatoes)	0.272727	0.545455	0.272727	1.000000	1.833333	0.123967	inf	0.625000
16	(chips, milk)	(beer)	0.272727	0.363636	0.272727	1.000000	2.750000	0.173554	inf	0.875000
17	(chips, beer)	(milk)	0.363636	0.454545	0.272727	0.750000	1.650000	0.107438	2.181818	0.619048
18	(milk, beer)	(chips)	0.272727	0.454545	0.272727	1.000000	2.200000	0.148760	inf	0.750000
19	(chips)	(milk, beer)	0.454545	0.272727	0.272727	0.600000	2.200000	0.148760	1.818182	1.000000
20	(milk)	(chips, beer)	0.454545	0.363636	0.272727	0.600000	1.650000	0.107438	1.590909	0.722222
21	(beer)	(chips, milk)	0.363636	0.272727	0.272727	0.750000	2.750000	0.173554	2.909091	1.000000

Para interpretar los resultados del Market Basket Analysis, es importante entender las métricas clave que se generan. Aquí tienes una explicación de cada una de las columnas en el DataFrame de reglas de asociación:

1. **antecedents**: Los productos que se encuentran en el lado izquierdo de la regla (antes de la flecha).
2. **consequents**: Los productos que se encuentran en el lado derecho de la regla (después de la flecha).
3. **antecedent support**: La proporción de transacciones que contienen los productos en los antecedentes.
4. **consequent support**: La proporción de transacciones que contienen los productos en los consecuentes.
5. **support**: La proporción de transacciones que contienen tanto los antecedentes como los consecuentes.

6. **confidence**: La probabilidad de que los productos en los consecuentes estén en una transacción dada que ya contiene los productos en los antecedentes. Se calcula como `support(antecedents + consequents) / support(antecedents)`.
7. **lift**: La razón de la confianza observada a la confianza esperada si los antecedentes y los consecuentes fueran independientes. Un lift mayor que 1 indica una asociación positiva.
8. **leverage**: La diferencia entre el soporte observado de los antecedentes y los consecuentes juntos y el soporte esperado si fueran independientes. Un valor positivo indica una asociación positiva.
9. **conviction**: Una medida de la implicación de los antecedentes en la ocurrencia de los consecuentes. Un valor mayor que 1 indica una asociación positiva.
10. **zhangs_metric**: Una métrica que combina la confianza y el lift para evaluar la fuerza de la asociación.

Ejemplo de Interpretación

Tomemos la primera segunda como ejemplo:

In [ ]:
```
rules.iloc[1]
```

Out[ ]:
```
antecedents             (beer)
consequents            (chips)
antecedent support    0.363636
consequent support    0.454545
support               0.363636
confidence                 1.0
lift                       2.2
leverage              0.198347
conviction                 inf
zhangs_metric         0.857143
Name: 1, dtype: object
```

- **Regla**: Si alguien compra `beer`, también comprará `chips`.
- **antecedent support**: El 36.36% de las transacciones contienen `beer`.
- **consequent support**: El 45.45% de las transacciones contienen `chips`.
- **support**: El 36.36% de las transacciones contienen tanto `beer` como `chips`.
- **confidence**: El 100% de las transacciones que contienen `beer` también contienen `chips`.
- **lift**: La probabilidad de comprar `chips` aumenta en un 120% (2.2 veces) cuando se compra `beer`.
- **leverage**: Hay un 19.83% más de transacciones que contienen ambos productos de lo que se esperaría si fueran independientes.
- **conviction**: La convicción es infinita, lo que indica una fuerte asociación.
- **zhangs_metric**: Un valor de 0.857143 indica una fuerte asociación positiva.

Conclusión

Las reglas con alta confianza, lift, y convicción son las más interesantes, ya que indican asociaciones fuertes entre productos. Puedes utilizar estas reglas para tomar decisiones de negocio, como promociones conjuntas, disposición de productos en la tienda, o recomendaciones de productos.