K mean CLUSTERING

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs
X, y = make_blobs(n_samples=300, centers=4, cluster_std=0.60, random_state=0)
plt.scatter(X[:, 0], X[:, 1], s=30)
plt.title('Data Points')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.grid()
plt.show()
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
kmeans = KMeans(n_clusters=4)
kmeans.fit(X)
predicted_labels = kmeans.predict(X)
# Plotting the clusters and their centers
plt.scatter(X[:, 0], X[:, 1], c=predicted_labels, s=30, cmap='viridis')
centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='red', s=200, alpha=0.75, marker='X')
plt.title('Clusters and their Centers')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.grid()
plt.show()
```

naïve bayes

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.datasets import make_classification
X, y = make_classification(n_samples=1000, n_features=20, n_classes=2, random_state=42)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
model = GaussianNB()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)
print("Confusion Matrix:\n", cm)
print("Accuracy:", accuracy)
```

association mining

```
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules
data = pd.DataFrame({
   'milk': [1, 0, 1, 1, 0],
   'bread': [1, 1, 0, 1, 0],
   'butter': [0, 1, 1, 0, 1],
   'cheese': [1, 1, 1, 1, 1]
}).astype(bool)  # Convert to boolean
frequent_itemsets = apriori(data, min_support=0.6, use_colnames=True)
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)

print(frequent_itemsets)
print(rules)
```

olap operations

```
import pandas as pd

# Data (simplified)
data = {'Product': ['A', 'A', 'B', 'B', 'C', 'C'],
        'Region': ['North', 'South', 'North', 'South', 'North', 'South'],
        'Time': ['2023-Q1', '2023-Q1', '2023-Q2', '2023-Q2', '2023-Q3', '2023-Q3'],
        'Sales': [100, 150, 120, 180, 200, 220]}

df = pd.DataFrame(data)

# Cube: Pivot Table (Mimicking Cube)
cube = df.pivot_table(values='Sales', index=['Product', 'Time'], columns='Region',
aggfunc='sum')

# 1. Roll-up (Aggregate data from detailed to summarized level)
rollup = df.groupby('Product')['Sales'].sum()

# 2. Drill-down (More detailed level of data)
drilldown = df[df['Product'] == 'A']  # Detailed data for Product A

# 3. Slice (Select a single dimension to create a sub-cube)
slice_data = cube.loc['A']  # Sales data for Product A

# 4. Dice (Select two or more dimensions to create a sub-cube)
dice_data = df[(df['Product'] == 'A') & (df['Region'] == 'North')]  # Sales for Product A in North
region

# 5. Pivot (Rotate the cube, reorienting the data)
pivot_data = df.pivot_table(values='Sales', index='Region', columns='Product', aggfunc='sum')

# Display results
print("Cube:\n", cube)
print("\nRoll-up (Summarize Sales across Products):\n", rollup)
print("\nDrill-down (Detailed Sales data for Product A):\n", drilldown)
print("\nSlice (Sales data for Product A):\n", slice_data)
print("\nDice (Sales for Product A in North region):\n", dice_data)
print("\nPivot (Reoriented data):\n", pivot_data)
```