**Practical Assignment – 4**

1. **Write a Python Programme to read the dataset (“Iris.csv”). Dataset download from (https://archive.ics.uci.edu/ml/datasets/iris) and apply Apriori algorithm.**

import pandas as pd

from mlxtend.frequent\_patterns import apriori, association\_rules

from mlxtend.preprocessing import TransactionEncoder

# 1. Load the Iris dataset

try:

df = pd.read\_csv("Iris.csv")

except FileNotFoundError:

print("Error: 'Iris.csv' not found. Please ensure the file is in the same directory.")

exit()

# 2. Preprocess the data for Apriori

# The Iris dataset contains numerical features and a categorical 'species' column.

# For Apriori, we need transactional data, typically represented as a list of lists,

# where each inner list represents a transaction (e.g., items purchased together).

# We'll discretize the numerical features and combine them with the 'species' column

# to create meaningful transactions.

# Discretize numerical features into bins

df['sepal\_length\_bin'] = pd.cut(df['sepal\_length'], bins=3, labels=['short', 'medium', 'long'])

df['sepal\_width\_bin'] = pd.cut(df['sepal\_width'], bins=3, labels=['narrow', 'medium', 'wide'])

df['petal\_length\_bin'] = pd.cut(df['petal\_length'], bins=3, labels=['small', 'medium', 'large'])

df['petal\_width\_bin'] = pd.cut(df['petal\_width'], bins=3, labels=['thin', 'medium', 'thick'])

# Create transactions by combining discretized features and species

transactions = []

for index, row in df.iterrows():

transaction = [

f"sepal\_length\_{row['sepal\_length\_bin']}",

f"sepal\_width\_{row['sepal\_width\_bin']}",

f"petal\_length\_{row['petal\_length\_bin']}",

f"petal\_width\_{row['petal\_width\_bin']}",

f"species\_{row['species']}"

]

transactions.append(transaction)

# 3. Encode the transactions

te = TransactionEncoder()

te\_ary = te.fit(transactions).transform(transactions)

df\_encoded = pd.DataFrame(te\_ary, columns=te.columns\_)

# 4. Apply the Apriori algorithm

# min\_support: The minimum support threshold for frequent itemsets.

frequent\_itemsets = apriori(df\_encoded, min\_support=0.05, use\_colnames=True)

# 5. Generate association rules

# min\_threshold: The minimum confidence threshold for association rules.

rules = association\_rules(frequent\_itemsets, metric="confidence", min\_threshold=0.7)

# 6. Print the results

print("Frequent Itemsets:")

print(frequent\_itemsets)

print("\nAssociation Rules:")

print(rules)

1. **Write a Python Programme to apply Apriori algorithm on Groceries dataset. Dataset can be downloaded from (https://github.com/amankharwal/Website data/blob/master/Groceries\_dataset.csv) Also display support and confidence for each rule.**

import pandas as pd

from mlxtend.preprocessing import TransactionEncoder

from mlxtend.frequent\_patterns import apriori, association\_rules

# Step 1: Load the Groceries dataset

# You can download it from: https://raw.githubusercontent.com/stedy/Machine-Learning-with-R-datasets/master/groceries.csv

# For this example, we'll simulate a few transactions

transactions = [

['milk', 'bread', 'nuts', 'apple'],

['milk', 'bread'],

['milk', 'bread', 'nuts'],

['bread', 'apple'],

['milk', 'apple'],

['bread', 'nuts'],

['milk', 'bread', 'apple'],

['nuts', 'apple'],

['milk', 'bread', 'nuts', 'apple'],

['milk', 'bread', 'apple']

]

# Step 2: Convert transactions to one-hot encoded DataFrame

te = TransactionEncoder()

te\_ary = te.fit(transactions).transform(transactions)

df = pd.DataFrame(te\_ary, columns=te.columns\_)

# Step 3: Apply Apriori algorithm

frequent\_itemsets = apriori(df, min\_support=0.3, use\_colnames=True)

# Step 4: Generate association rules

rules = association\_rules(frequent\_itemsets, metric="confidence", min\_threshold=0.6)

# Step 5: Display results

print("📊 Frequent Itemsets:")

print(frequent\_itemsets)

print("\n🔗 Association Rules:")

print(rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']])

1. **Write a Python program to read “StudentsPerformance.csv” file. Solve following: - To display the shape of dataset. - To display the top rows of the dataset with their columns. - To display the number of rows randomly. - To display the number of columns and names of the columns.**

import pandas as pd

# Load the dataset

df = pd.read\_csv("StudentsPerformance.csv")

# 1. Display the shape of the dataset

print("📐 Shape of the dataset (rows, columns):", df.shape)

# 2. Display the top rows of the dataset with their columns

print("\n🔝 Top 5 rows of the dataset:")

print(df.head())

# 3. Display a random sample of rows

num\_random\_rows = 5 # You can change this number as needed

print(f"\n🎲 {num\_random\_rows} Random rows from the dataset:")

print(df.sample(num\_random\_rows))

# 4. Display the number of columns and names of the columns

print("\n📊 Number of columns:", len(df.columns))

print("📝 Column names:")

print(df.columns.tolist())

1. **Write a Python program for Apriori Algorithm using ARM. And Print the Rule, Support, Confidence and Lift. - (Set Min\_Support = 0.0040, Min\_Confidence=0.2, Min\_Lift=3, Min\_Length=2)**

import pandas as pd

from mlxtend.preprocessing import TransactionEncoder

from mlxtend.frequent\_patterns import apriori, association\_rules

transactions = [

['milk', 'bread', 'nuts', 'apple'],

['milk', 'bread'],

['milk', 'bread', 'nuts'],

['bread', 'nuts'],

['milk', 'apple'],

['milk', 'bread', 'nuts', 'apple'],

['bread', 'apple'],

['milk', 'nuts'],

['bread', 'nuts'],

['milk', 'bread', 'apple']

]

# Encode transactions

te = TransactionEncoder()

df = pd.DataFrame(te.fit\_transform(transactions), columns=te.columns\_)

# Generate frequent itemsets

frequent\_itemsets = apriori(df, min\_support=0.2, use\_colnames=True)

# Generate rules

rules = association\_rules(frequent\_itemsets, metric="lift", min\_threshold=1.0)

# Filter rules

filtered\_rules = rules[

(rules['confidence'] >= 0.2) &

(rules['lift'] >= 1.2) &

(rules.apply(lambda row: len(row['antecedents'].union(row['consequents'])) >= 2, axis=1))

]

# Print rules

print("📋 Association Rules:")

if not filtered\_rules.empty:

for \_, rule in filtered\_rules.iterrows():

print(f"Rule: {set(rule['antecedents'])} → {set(rule['consequents'])}")

print(f"Support: {rule['support']:.4f}")

print(f"Confidence: {rule['confidence']:.4f}")

print(f"Lift: {rule['lift']:.4f}")

print("-" \* 40)

else:

print("No rules found with current filters.")

**Activity 1 Question**

1. **Use a dataset to apply k-Means and visualize clusters using matplotlib or seaborn.**

**Note – Write down the code in your notebook and perform it on 8/8/2025 & 9/8/2025 (in your practical session).**