**Name:- Nandini Dnyaneshwar Patil Date:-25/02/2024**

**Batch:-05 Roll No:-151**

**Practial Name:-02 Read CSV File.**

import pandas as pd

# Load the data from a CSV file

file\_path = 'sales\_data.csv' # Replace with your file path

sales\_data = pd.read\_csv(file\_path)

# Display basic data structure

print("Display rows of the dataset:")

print(sales\_data)

**Name:- Nandini Dnyaneshwar Patil Date:-25/02/2024**

**Batch:-05 Roll No:-151**

**Practial Name:- 6: - Classification using Random Forest.**

# Import necessary libraries  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.model\_selection import train\_test\_split  
from sklearn.datasets import load\_iris  
from sklearn.metrics import accuracy\_score, classification\_report  
  
# Load the Iris dataset  
data = load\_iris()  
X = data.data # Features  
y = data.target # Target variable (class labels)  
  
# Split the data into training and testing sets (70% train, 30% test)  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)  
  
# Initialize the Random Forest Classifier  
rf\_classifier = RandomForestClassifier(n\_estimators=100, random\_state=42)  
  
# Train the model  
rf\_classifier.fit(X\_train, y\_train)  
  
# Predict on the test set  
y\_pred = rf\_classifier.predict(X\_test)  
  
# Evaluate the model  
accuracy = accuracy\_score(y\_test, y\_pred)  
print(f"Accuracy: {accuracy \* 100:.2f}%")  
  
# Detailed classification report (Precision, Recall, F1-score for each class)  
print("Classification Report:\n", classification\_report(y\_test, y\_pred))  
  
# Feature Importance: Display the importance of each feature  
print("Feature Importance:")  
for feature, importance in zip(data.feature\_names, rf\_classifier.feature\_importances\_):  
 print(f"{feature}: {importance:.4f}")

**Output:-**

Accuracy: 100.00%

Classification Report:

precision recall f1-score support

0 1.00 1.00 1.00 19

1 1.00 1.00 1.00 13

2 1.00 1.00 1.00 13

accuracy 1.00 45

macro avg 1.00 1.00 1.00 45

weighted avg 1.00 1.00 1.00 45

Feature Importance:

sepal length (cm): 0.1041

sepal width (cm): 0.0446

petal length (cm): 0.4173

petal width (cm): 0.4340

**Name:- Nandini Dnyaneshwar Patil Date:-25/02/2024**

**Batch:-05 Roll No:-151**

**Practial Name 8: - Association Rule Mining using Apriori.**

# Import necessary libraries  
import pandas as pd  
from mlxtend.frequent\_patterns import apriori, association\_rules  
  
# Sample dataset (You can load your own dataset as well)  
# Here, each row represents a transaction with items purchased (1 if item is bought, 0 if not)  
data = {'Milk': [1, 1, 0, 1, 1],  
 'Bread': [1, 1, 1, 1, 0],  
 'Butter': [0, 1, 1, 1, 1],  
 'Cheese': [1, 0, 1, 1, 1]}  
  
df = pd.DataFrame(data)  
  
# Apply the Apriori algorithm to find frequent itemsets with a minimum support of 0.6  
frequent\_itemsets = apriori(df, min\_support=0.6, use\_colnames=True)  
  
# Generate association rules with a minimum confidence of 0.7  
rules = association\_rules(frequent\_itemsets, metric="confidence", min\_threshold=0.7)  
  
# Display the frequent itemsets  
print("Frequent Itemsets:")  
print(frequent\_itemsets)  
  
# Display the association rules  
print("\nAssociation Rules:")  
print(rules)

**Output:-**

**Frequent Itemsets:**

**support itemsets**

0 0.8 (Milk)

1 0.8 (Bread)

2 0.8 (Butter)

3 0.8 (Cheese)

4 0.6 (Bread, Milk)

5 0.6 (Milk, Butter)

6 0.6 (Milk, Cheese)

7 0.6 (Bread, Butter)

8 0.6 (Bread, Cheese)

9 0.6 (Cheese, Butter)

**Association Rules:**

antecedents consequents antecedent support ... jaccard certainty kulczynski

0 (Bread) (Milk) 0.8 ... 0.6 -0.25 0.75

1 (Milk) (Bread) 0.8 ... 0.6 -0.25 0.75

2 (Milk) (Butter) 0.8 ... 0.6 -0.25 0.75

3 (Butter) (Milk) 0.8 ... 0.6 -0.25 0.75

4 (Milk) (Cheese) 0.8 ... 0.6 -0.25 0.75

5 (Cheese) (Milk) 0.8 ... 0.6 -0.25 0.75

6 (Bread) (Butter) 0.8 ... 0.6 -0.25 0.75

7 (Butter) (Bread) 0.8 ... 0.6 -0.25 0.75

8 (Bread) (Cheese) 0.8 ... 0.6 -0.25 0.75

9 (Cheese) (Bread) 0.8 ... 0.6 -0.25 0.75

10 (Cheese) (Butter) 0.8 ... 0.6 -0.25 0.75

11 (Butter) (Cheese) 0.8 ... 0.6 -0.25 0.75

[12 rows x 14 columns]

**Name:- Nandini Dnyaneshwar Patil Date:-25/02/2024**

**Batch:-05 Roll No:-151**

**Practial Name 2:- Read CSV file.**

import pandas as pd  
  
# Load the data from a CSV file  
file\_path = 'hh.csv' # Replace with your file path  
sales\_data = pd.read\_csv(file\_path)  
  
# Display basic data structure  
print("Display rows of the dataset:")  
print(sales\_data)

Output:-

Display rows of the dataset:

**Date Region Product Sales**

0 1/5/2023 North A 100

1 1/12/2023 South B 200

2 2/1/2023 North A 150

3 2/14/2023 East C 300

4 2/21/2023 West B 250

**Name:- Nandini Dnyaneshwar Patil Date:-25/02/2024**

**Batch:-05 Roll No:-151**

**Practial Name . 3:- Perform data filtering, and calculate aggregate statistics.**

import pandas as pd  
  
# Load the data from a CSV file  
file\_path = 'hh.csv' # Replace with your file path  
sales\_data = pd.read\_csv(file\_path)  
  
# Display basic data structure  
print("First few rows of the dataset:")  
print(sales\_data.head())  
  
# \*\*Data Filtering\*\*: Select data where Sales > 150  
filtered\_data = sales\_data[sales\_data['Sales'] > 150]  
print("\nFiltered Data (Sales > 150):")  
print(filtered\_data)  
  
# \*\*Aggregate Statistics\*\*: Calculate total and average sales by region  
region\_sales = sales\_data.groupby('Region')['Sales'].agg(['sum', 'mean']).reset\_index()  
print("\nTotal and Average Sales by Region:")  
print(region\_sales)  
  
# \*\*Aggregate Statistics\*\*: Calculate total sales and count by product  
product\_stats = sales\_data.groupby('Product')['Sales'].agg(['sum', 'count']).reset\_index()  
print("\nTotal Sales and Transaction Count by Product:")  
print(product\_stats)

Output:-

First few rows of the dataset:

Date Region Product Sales

0 1/5/2023 North A 100

1 1/12/2023 South B 200

2 2/1/2023 North A 150

3 2/14/2023 East C 300

4 2/21/2023 West B 250

Filtered Data (Sales > 150):

Date Region Product Sales

1 1/12/2023 South B 200

3 2/14/2023 East C 300

4 2/21/2023 West B 250

Total and Average Sales by Region:

Region sum mean

0 East 300 300.0

1 North 250 125.0

2 South 200 200.0

3 West 250 250.0

Total Sales and Transaction Count by Product:

Product sum count

0 A 250 2

1 B 450 2

2 C 300 1