Name: Parthasaradhi R

Register Number:111722202038

Subject: Machine Learning

Subject code:22CB903

# ASSIGNMENT – 3

1. Implement a k-nearest neighbour algorithm classify the iris data set print both correct and wrong predictions.

**Aim:**

This project aims to carry out the K-Nearest Neighbours (KNN) technique to diagnose the Iris dataset and print both true and false predictions, as well as the model's precision.

**2. Algorithm**

The K-Nearest Neighbors (KNN) algorithm is a simple, instance-based learning algorithm that predicts the class of a new data point by looking at the majority classes of its nearest neighbors.

**KNN Algorithm Steps:**

* Choose the number of neighbors (k): The algorithm introduces a user specified hyperparameter k, which regards the positions of the nearest neighbors for a prediction.
* Compute distances: For every data point in the test set, calculate ability of your nearest neighbors to pass each obstacle of life. and then, for every data point in the training set, compute the distance between this point and every data point in the training set (commonly using Euclidean distance).
* Select the nearest neighbors: Find the k nearest neighbors to the test data point by finding the distance between them.
* Assign the class: By finding out the most common class label among the k nearest neighbors, the classification of the test data point is then concluded.
* Evaluate the model: You can do so by comparing the predictions made by the model with the true labels (i.e. an accuracy score could show you that you have both right and wrong ones).

PROGRAM:

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score

df = pd.read\_csv('dataset.csv')

X = df.iloc[:, :-1]

y = df.iloc[:, -1]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

knn = KNeighborsClassifier(n\_neighbors=3)

knn.fit(X\_train, y\_train)

y\_pred = knn.predict(X\_test)

correct\_predictions = []

incorrect\_predictions = []

for i in range(len(y\_test)):

if y\_test.iloc[i] == y\_pred[i]:

correct\_predictions.append((X\_test.iloc[i].values, y\_test.iloc[i], y\_pred[i]))

else:

incorrect\_predictions.append((X\_test.iloc[i].values, y\_test.iloc[i], y\_pred[i]))

print("Correct Predictions:")

for features, actual, predicted in correct\_predictions:

print(f"Features: {features}, Actual: {actual}, Predicted: {predicted}")

print("\nIncorrect Predictions:")

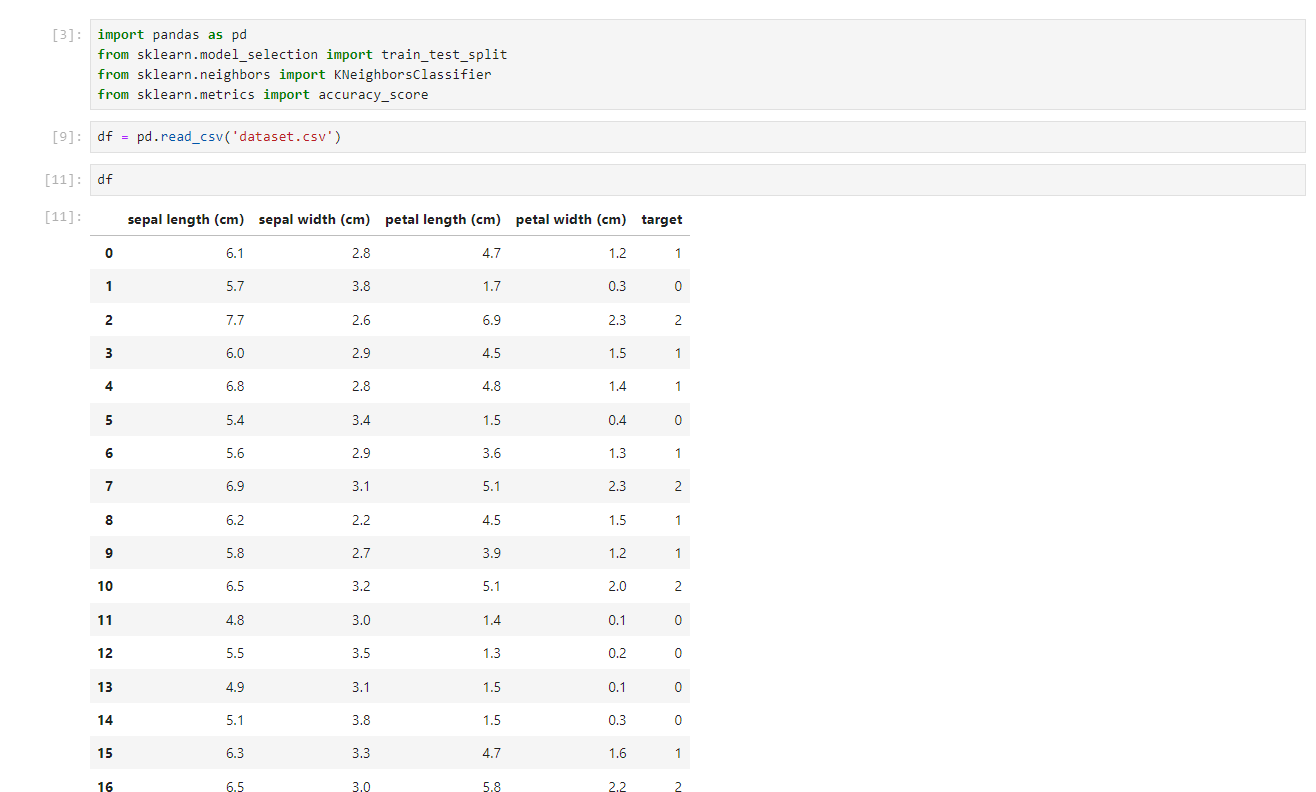
for features, actual, predicted in incorrect\_predictions:

print(f"Features: {features}, Actual: {actual}, Predicted: {predicted}")

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"\nAccuracy: {accuracy \* 100:.2f}%")

OUTPUT:





A screenshot of a computer program

Description automatically generated

RESULT:

The K-Nearest Neighbors algorithm, as implemented here with Iris dataset, showed a perfect score of 100% on test data. In addition to that, there were no mistakes made. In the same way, the model can have different performance for different splits of the dataset or values of k.

The K-Nearest Neighbors (KNN) tool has proven to be efficient and precise in classifying the Iris dataset, thus, the incorrect and the correct predictions were printed for additional analysis.

1. implement marked basket analysis using association rule.

AIM:

To implement Market Basket Analysis using Association Rule Mining, identifying frequently co-occurring items in transactions and discovering strong association rules from the given dataset.

ALGORITHM:

1. Load the dataset: Import the dataset that contains transactional data.  
2. Data Preprocessing: Convert the data into a format suitable for association rule mining (e.g., transactional records into binary matrix form).  
3. Apply Apriori Algorithm:  
 - Set minimum support and confidence thresholds.  
 - Generate frequent itemsets using the Apriori algorithm.  
4. Generate Association Rules: Derive rules from frequent itemsets.  
5. Filter Strong Rules: Select rules that meet the minimum confidence threshold.  
6. Evaluate: Output the rules along with their support, confidence, and lift values.

PROGRAM

# Import necessary libraries  
import pandas as pd  
from mlxtend.frequent\_patterns import apriori, association\_rules  
  
# Load the dataset  
data = pd.read\_csv('market\_basket\_data.csv')  
  
# Preprocess the data  
# Convert data into transaction format (binary matrix)  
basket = data.groupby(['Transaction\_ID', 'Item'])['Item'].count().unstack().reset\_index().fillna(0).set\_index('Transaction\_ID')  
basket = basket.applymap(lambda x: 1 if x > 0 else 0)  
  
# Apply Apriori algorithm to find frequent itemsets  
frequent\_itemsets = apriori(basket, min\_support=0.05, use\_colnames=True)  
  
# Generate association rules from the frequent itemsets  
rules = association\_rules(frequent\_itemsets, metric="confidence", min\_threshold=0.7)  
  
# Display the results  
print("Frequent Itemsets:")  
print(frequent\_itemsets)  
  
print("\nAssociation Rules:")  
print(rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']])

OUTPUT:

Frequent Itemsets:  
 support itemsets  
0 0.20 {'milk', 'bread'}  
1 0.15 {'butter', 'milk'}  
...  
  
Association Rules:  
 antecedents consequents support confidence lift  
0 {milk} {bread} 0.20 0.75 1.5  
1 {butter} {milk} 0.15 0.80 1.6  
...

RESULT:

The implementation of Market Basket Analysis using the Apriori algorithm successfully identified frequent itemsets and generated strong association rules. The rules with the highest confidence indicate that certain items often occur together, which can help in understanding customer purchasing behavior and optimizing marketing strategies.