**Lab 5: Classification**

1. **Write python program to load Breast Cancer dataset and train-test, with 80:20 ratio, using SVM and finally display the classification report, confusion matrix and visualization of decision boundary. [Support Vector Machine]**

from sklearn import datasets

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

from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score, confusion\_matrix

# Load dataset

data = datasets.load\_breast\_cancer()

X, y = data.data, data.target

# Split data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Create SVM model

model = SVC(kernel='linear')

# Train

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

# Predict

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

# Accuracy

print("Accuracy:", accuracy\_score(y\_test, y\_pred))

# Confusion matrix

print("Confusion Matrix:")

print(confusion\_matrix(y\_test, y\_pred))

**Your Work:**

1. **To visualize the decision boundary**
2. **To train-test the dataset with Polynomial kernel SVM**
3. **To train-test the dataset with Rbf Kerenl SVM**
4. **To train-test the dataset with Logistic regression Classifier.**
5. **Consider the following dataset. Given a new sample with X1 = 3, X2 = 7, classify it using KNN (k=3). What class will it belong to? Write python code. [K-Nearest Neighbor classifier]**

|  |  |  |
| --- | --- | --- |
| **X1** | **X2** | **Y (Classification)** |
| 7 | 7 | Bad |
| 7 | 4 | Bad |
| 3 | 4 | Good |
| 1 | 4 | Good |

from sklearn.neighbors import KNeighborsClassifier

import numpy as np

# Dataset

X = np.array([

    [7, 7],

    [7, 4],

    [3, 4],

    [1, 4]

])

y = np.array(['Bad', 'Bad', 'Good', 'Good'])

# New data point to classify

new\_sample = np.array([[3, 7]])

# Create and train KNN model with k=3

knn = KNeighborsClassifier(n\_neighbors=3)

knn.fit(X, y)

# Predict the class

predicted\_class = knn.predict(new\_sample)

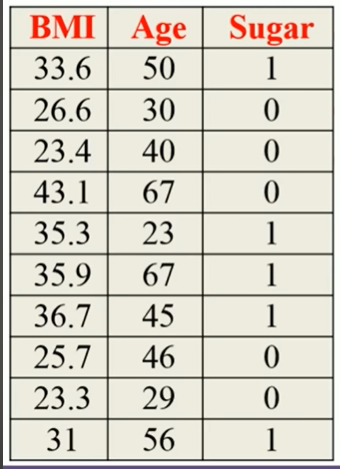
# Output result

print(f"The predicted class for point {new\_sample[0]} is: {predicted\_class[0]}")

**Your Work:**

**Show the visualization.**

1. **Write python code to implement KNN classifier (with K =3) for predicting the new point BMI= 43.6 and Age= 40 with reference to below dataset.**



import numpy as np

from sklearn.neighbors import KNeighborsClassifier

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

# Step 1: Define the dataset

bmi = np.array([33.6, 26.6, 23.4, 43.1, 35.3, 35.9, 36.7, 25.7, 23.3, 31])

age = np.array([50, 30, 40, 67, 23, 67, 45, 46, 29, 56])

sugar = np.array([1, 0, 0, 0, 1, 1, 1, 0, 0, 1])

# Step 2: Stack BMI and Age into one 2D array (features)

X = np.column\_stack((bmi, age))

# Step 3: Initialize the KNeighborsClassifier with K=3

knn = KNeighborsClassifier(n\_neighbors=3)

# Step 4: Fit the model to the data

knn.fit(X, sugar)

# Step 5: Define the new point (BMI=43.6 and Age=40)

new\_point = np.array([[43.6, 40]])

# Step 6: Predict the sugar level for the new point

prediction = knn.predict(new\_point)

# Step 7: Print the result

print(f"The predicted sugar level for BMI=43.6 and Age=40 is: {prediction[0]}")

**Your Work:**

**Show the visualization.**

1. **Create a Play Golf Dataset as below (Play is the Target attribute) and perform as below: [Decision Tree]**

**data = {**

**"Outlook": ["Sunny", "Sunny", "Overcast", "Rain", "Rain", "Rain", "Overcast",**

**"Sunny", "Sunny", "Rain", "Sunny", "Overcast", "Overcast", "Rain"],**

**"Temp": ["Hot", "Hot", "Hot", "Mild", "Cool", "Cool", "Cool",**

**"Mild", "Cool", "Mild", "Mild", "Mild", "Hot", "Mild"],**

**"Humidity": ["High", "High", "High", "High", "Normal", "Normal", "Normal",**

**"High", "Normal", "Normal", "Normal", "High", "Normal", "High"],**

**"Windy": [False, True, False, False, False, True, True,**

**False, False, False, True, True, False, True],**

**"Play": ["No", "No", "Yes", "Yes", "Yes", "No", "Yes",**

**"No", "Yes", "Yes", "Yes", "Yes", "Yes", "No"]**

**}**

1. **Encode this dataset using Labelencoder() function.**
2. **Display this new featured dataset.**
3. **Divide this new dataset into Train and Test dataset with 80:20 ratio.**
4. **Develop a model with DecisionTreeClassifier() using entropy.**
5. **Train your model with the Training dataset. And then test your model with testing dataset. Find the accuracy of your model.**
6. **Display the Decision Tree.**

import pandas as pd

from sklearn.tree import DecisionTreeClassifier, export\_text, plot\_tree

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

from sklearn.preprocessing import LabelEncoder

from sklearn.metrics import accuracy\_score

import matplotlib.pyplot as plt

# Create the Play Golf dataset

data = {

    "Outlook": ["Sunny", "Sunny", "Overcast", "Rain", "Rain", "Rain", "Overcast",

                "Sunny", "Sunny", "Rain", "Sunny", "Overcast", "Overcast", "Rain"],

    "Temp": ["Hot", "Hot", "Hot", "Mild", "Cool", "Cool", "Cool",

             "Mild", "Cool", "Mild", "Mild", "Mild", "Hot", "Mild"],

    "Humidity": ["High", "High", "High", "High", "Normal", "Normal", "Normal",

                 "High", "Normal", "Normal", "Normal", "High", "Normal", "High"],

    "Windy": [False, True, False, False, False, True, True,

              False, False, False, True, True, False, True],

    "Play": ["No", "No", "Yes", "Yes", "Yes", "No", "Yes",

             "No", "Yes", "Yes", "Yes", "Yes", "Yes", "No"]

}

# Load data into a pandas DataFrame

df = pd.DataFrame(data)

df

# Encode categorical variables into numeric values

encoder = LabelEncoder()

df["Outlook"] = encoder.fit\_transform(df["Outlook"]) # Convert(Overcast=0, Rain=1, Sunny=2)

df["Temp"] = encoder.fit\_transform(df["Temp"]) # Convert(Cool=0, Mild=1, Hot=2)

df["Humidity"] = encoder.fit\_transform(df["Humidity"]) # Convert(High=0, Normal=1)

df["Windy"] = df["Windy"].astype(int)  # Convert boolean to integer (False=0, True=1)

df["Play"] = encoder.fit\_transform(df["Play"]) # Convert(No=0, Yes=1)

# Features and target variable

X = df[["Outlook", "Temp", "Humidity", "Windy"]]

y = df["Play"]

# Train-test split with 80:20 ratio

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Train a Decision Tree Classifier

clf = DecisionTreeClassifier(criterion="entropy", random\_state=42)  # Using Information Gain

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

# Evaluate the model

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

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

print(f"Accuracy on Test Data: {accuracy:.2f}")

# Display the Decision Tree Rules

tree\_rules = export\_text(clf, feature\_names=list(X.columns))

print("\nDecision Tree Rules:")

print(tree\_rules)

# Visualize the Decision Tree

plt.figure(figsize=(12, 8))

plot\_tree(clf, feature\_names=list(X.columns), class\_names=encoder.classes\_, filled=True, rounded=True)

plt.title("Decision Tree for Play Golf Dataset")

plt.show()

**Your Work:**

**Perform test with a new data on your model: Outlook=Sunny, Temp=Cool, Humidity=High, Windy=False**

1. **Perform train-test with Random forest of 5 trees for below dataset. Finally, test a new dataset [ Outlook=Sunny, Temp=Cool, Humidity=High, Windy=False ] for which class does it belong. Also, show the predictions by each model, and then show the majority voting.**

**data = {**

**"Outlook": ["Sunny", "Sunny", "Overcast", "Rain", "Rain", "Rain", "Overcast",**

**"Sunny", "Sunny", "Rain", "Sunny", "Overcast", "Overcast", "Rain"],**

**"Temp": ["Hot", "Hot", "Hot", "Mild", "Cool", "Cool", "Cool",**

**"Mild", "Cool", "Mild", "Mild", "Mild", "Hot", "Mild"],**

**"Humidity": ["High", "High", "High", "High", "Normal", "Normal", "Normal",**

**"High", "Normal", "Normal", "Normal", "High", "Normal", "High"],**

**"Windy": [False, True, False, False, False, True, True,**

**False, False, False, True, True, False, True],**

**"Play": ["No", "No", "Yes", "Yes", "Yes", "No", "Yes",**

**"No", "Yes", "Yes", "Yes", "Yes", "Yes", "No"]**

**}**

import pandas as pd

from sklearn.tree import DecisionTreeClassifier, export\_text, plot\_tree

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

from sklearn.preprocessing import LabelEncoder

from sklearn.metrics import accuracy\_score

from sklearn.ensemble import RandomForestClassifier

from collections import Counter

import matplotlib.pyplot as plt

# Create the Play Golf dataset

data = {

    "Outlook": ["Sunny", "Sunny", "Overcast", "Rain", "Rain", "Rain", "Overcast",

                "Sunny", "Sunny", "Rain", "Sunny", "Overcast", "Overcast", "Rain"],

    "Temp": ["Hot", "Hot", "Hot", "Mild", "Cool", "Cool", "Cool",

             "Mild", "Cool", "Mild", "Mild", "Mild", "Hot", "Mild"],

    "Humidity": ["High", "High", "High", "High", "Normal", "Normal", "Normal",

                 "High", "Normal", "Normal", "Normal", "High", "Normal", "High"],

    "Windy": [False, True, False, False, False, True, True,

              False, False, False, True, True, False, True],

    "Play": ["No", "No", "Yes", "Yes", "Yes", "No", "Yes",

             "No", "Yes", "Yes", "Yes", "Yes", "Yes", "No"]

}

# Load data into a pandas DataFrame

df = pd.DataFrame(data)

# Separate LabelEncoders for each categorical column

outlook\_encoder = LabelEncoder()

temp\_encoder = LabelEncoder()

humidity\_encoder = LabelEncoder()

play\_encoder = LabelEncoder()

df["Outlook"] = outlook\_encoder.fit\_transform(df["Outlook"])  # Overcast=0, Rain=1, Sunny=2

df["Temp"] = temp\_encoder.fit\_transform(df["Temp"])          # Cool=0, Mild=1, Hot=2

df["Humidity"] = humidity\_encoder.fit\_transform(df["Humidity"])  # High=0, Normal=1

df["Windy"] = df["Windy"].astype(int)                        # False=0, True=1

df["Play"] = play\_encoder.fit\_transform(df["Play"])          # No=0, Yes=1

# Features and target variable

X = df[["Outlook", "Temp", "Humidity", "Windy"]]

y = df["Play"]

# Train-test split (80:20)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Train a Random Forest Classifier with 5 trees

model = RandomForestClassifier(n\_estimators=5, random\_state=42)

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

# Evaluate the model using the test set

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

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

print(f"Accuracy on Test Data: {accuracy:.2f}")

# Prepare new data point for prediction

new\_data = pd.DataFrame({

    "Outlook": [outlook\_encoder.transform(["Sunny"])[0]],  # Sunny = 2

    "Temp": [temp\_encoder.transform(["Cool"])[0]],        # Cool = 0

    "Humidity": [humidity\_encoder.transform(["High"])[0]], # High = 0

    "Windy": [0]  # False = 0

})

# Show individual tree predictions for the new data

print("\nIndividual tree predictions for new data:")

votes = []

label\_map = {i: label for i, label in enumerate(play\_encoder.classes\_)}

for i, tree in enumerate(model.estimators\_):

    pred = tree.predict(new\_data)[0]

    votes.append(pred)

    print(f"Tree {i+1}: {label\_map[pred]}")

# Majority voting result

majority\_vote = Counter(votes).most\_common(1)[0][0]

print(f"\nMajority Voting Result for new data: {label\_map[majority\_vote]}")

\*\*\*