import pandas as pd

from sklearn.datasets import load\_iris

import matplotlib.pyplot as plt

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

from sklearn.svm import SVC

# Load Iris dataset and create DataFrame

iris = load\_iris()

df = pd.DataFrame(iris.data, columns=iris.feature\_names)

df['target'] = iris.target

df['flower\_name'] = df['target'].apply(lambda x: iris.target\_names[x])

# Split dataset by class

df0, df1, df2 = df[:50], df[50:100], df[100:]

# Plot Sepal and Petal dimensions for Setosa and Versicolor

def plot\_dimensions(x, y, xlabel, ylabel, title):

plt.title(f"By Rohitkumar Pandey 211P002\n{title}")

plt.xlabel(xlabel)

plt.ylabel(ylabel)

plt.scatter(df0[x], df0[y], color="red", marker='+', label="Setosa")

plt.scatter(df1[x], df1[y], color="blue", marker='.', label="Versicolor")

plt.legend()

plt.show()

plot\_dimensions('sepal length (cm)', 'sepal width (cm)', 'Sepal Length', 'Sepal Width', "Plot Sepal Length vs Sepal Width")

plot\_dimensions('petal length (cm)', 'petal width (cm)', 'Petal Length', 'Petal Width', "Plot Petal Length vs Petal Width")

# Prepare training and testing data

X = df.drop(['target', 'flower\_name'], axis='columns')

y = df['target']

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

# Create and train SVM models with different parameters

models = {

'default': SVC(),

'C=1': SVC(C=1),

'C=10': SVC(C=10),

'gamma=10': SVC(gamma=10),

'linear': SVC(kernel='linear')

}

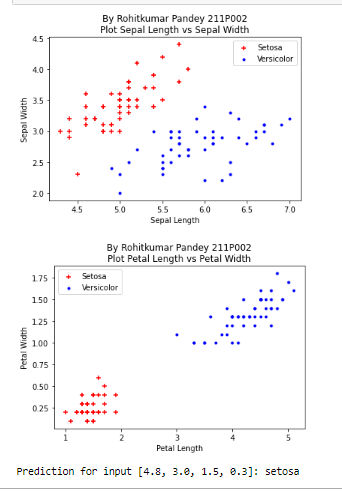
for name, model in models.items():

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

# Example prediction

prediction = models['default'].predict([[4.8, 3.0, 1.5, 0.3]])

print(f"Prediction for input [4.8, 3.0, 1.5, 0.3]: {iris.target\_names[prediction[0]]}")



import tkinter as tk

from tkinter import ttk, messagebox

import pandas as pd

from sklearn.datasets import load\_diabetes

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

from sklearn.linear\_model import LinearRegression

# Load the Diabetes dataset

diabetes = load\_diabetes()

df = pd.DataFrame(diabetes.data, columns=diabetes.feature\_names)

df['target'] = diabetes.target

# Select fewer features correctly

X = df[['age', 'bmi', 'bp', 's5']] # 'bp' is blood pressure and 's5' is a selected feature

y = df['target']

# Split data into training and testing sets

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

# Train the model once

model = LinearRegression()

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

# Create the GUI window

window = tk.Tk()

window.title("Diabetes Progression Prediction")

window.geometry("400x400")

# Create input fields for selected features

labels = {

'age': "Age (years)",

'bmi': "Body Mass Index (BMI)",

'bp': "Blood Pressure (mm Hg)",

's5': "S5 Measurement"

}

entries = []

for i, (key, label\_text) in enumerate(labels.items()):

label = ttk.Label(window, text=label\_text + ":")

label.grid(row=i, column=0, padx=5, pady=5)

entry = ttk.Entry(window)

entry.grid(row=i, column=1, padx=5, pady=5)

entries.append(entry)

# Create a button to predict

def predict():

try:

# Get input values

input\_values = [float(entry.get()) for entry in entries]

# Make prediction

prediction = model.predict([input\_values])[0]

# Display prediction

result\_label.config(text="Disease Progression: {:.2f}".format(prediction))

except ValueError:

messagebox.showerror("Input Error", "Invalid input. Please enter valid numbers for all features.")

except Exception as e:

messagebox.showerror("Prediction Error", f"An error occurred: {str(e)}")

# Create a button to predict

predict\_button = ttk.Button(window, text="Predict", command=predict)

predict\_button.grid(row=len(labels), column=0, columnspan=2, padx=5, pady=10)

# Create a label to display the prediction

result\_label = ttk.Label(window, text="")

result\_label.grid(row=len(labels) + 1, column=0, columnspan=2, padx=5, pady=5)

# Create a reset button to clear inputs

def reset():

for entry in entries:

entry.delete(0, tk.END)

result\_label.config(text="")

reset\_button = ttk.Button(window, text="Reset", command=reset)

reset\_button.grid(row=len(labels) + 2, column=0, columnspan=2, padx=5, pady=5)

# Start the GUI event loop

window.mainloop()

