**Exp-2**

**#Rohitkumar pandey 211P002**

**import matplotlib.pyplot as plt**

**from matplotlib.figure import Figure**

**from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg**

**import pandas as pd**

**import numpy as np**

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

**from sklearn.linear\_model import LinearRegression**

**import tkinter as tk**

**df = pd.read\_csv("placement.csv")**

**root = tk.Tk()**

**root.title("CGPA vs Package Prediction")**

**root.geometry("800x600")**

**fig = Figure(figsize=(8, 6))**

**scatter\_plot = fig.add\_subplot(111)**

**x = df.iloc[:, 0].values.reshape(-1, 1)**

**y = df.iloc[:, -1].values**

**x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.2, random\_state=2)**

**model = LinearRegression()**

**model.fit(x\_train, y\_train)**

**model.score(x\_test, y\_test)**

**def predict\_package():**

**try:**

**cgpa = float(entry\_cgpa.get())**

**predicted\_package = model.predict([[cgpa]])**

**label\_prediction.config(text=f"Predicted Package: {predicted\_package[0]:.2f} LPA")**

**except ValueError:**

**label\_prediction.config(text="Please enter a valid CGPA")**

**label\_cgpa = tk.Label(root, text="Enter CGPA:")**

**label\_cgpa.pack(pady=10)**

**entry\_cgpa = tk.Entry(root, width=10)**

**entry\_cgpa.pack()**

**button\_predict = tk.Button(root, text="Predict Package", command=predict\_package)**

**button\_predict.pack(pady=10)**

**label\_prediction = tk.Label(root, text="")**

**label\_prediction.pack(pady=10)**

**scatter\_plot.scatter(df['cgpa'], df['package'])**

**scatter\_plot.plot(x\_train, model.predict(x\_train), color='red') # Use x\_train and predict on it**

**scatter\_plot.set\_xlabel('CGPA')**

**scatter\_plot.set\_ylabel('Package')**

**scatter\_plot.set\_title('CGPA vs Package')**

**canvas = FigureCanvasTkAgg(fig, master=root)**

**canvas.draw()**

**canvas.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)**

**tk.mainloop()**

**#PostLab 1**

**import matplotlib.pyplot as plt**

**import pandas as pd**

**import numpy as np**

**df = pd.read\_csv('./placement.csv')**

**df.head()**

**# plotting the scatter plot**

**plt.scatter(df['cgpa'],df['package'])**

**plt.xlabel('CGPA') # setting x label**

**plt.ylabel('Package(in lpa)') # setting y label**

**plt.title('By Rohitkumar Pandey')**

**plt.show()**

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

**# Getting cgpa values as X and pacakge as Y**

**X, Y= df.iloc[:,0], df.iloc[:,1]**

**X\_train,X\_test,Y\_train,Y\_test = train\_test\_split(X,Y,test\_size=0.2,random\_state=0)**

**lr\_model = LinearRegression()**

**lr\_model.fit(np.array(X\_train).reshape(-1,1),np.array(Y\_train).reshape(-1,1))**

**Y\_pred = lr\_model.predict(np.array(7).reshape(-1, 1))**

**print(Y\_pred)**

**class LinearRegression:**

**def \_\_init\_\_(self):**

**self.m = None**

**self.c = None**

**def fit(self,x\_train,y\_train):**

**num, den = 0,0**

**x\_mean, y\_mean = x\_train.mean(), y\_train.mean()**

**for i in range(len(x\_train)):**

**num = num + ((x\_train[i]-x\_mean)\*(y\_train[i]-y\_mean))**

**den = den + ((x\_train[i]-x\_mean)\*\*2)**

**m = num/den**

**c = y\_mean - (m\*x\_mean)**

**self.m = m**

**self.c = c**

**def predict(self,x\_test):**

**if self.m is None or self.c is None:**

**print('Model is not trained yet, Call the fit method.')**

**return None**

**return self.m\*x\_test + self.c**

**import tkinter as tk**

**from matplotlib.figure import Figure**

**from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg**

**root = tk.Tk()**

**root.title("Package (in LPA) Prediction")**

**root.geometry('800x400')**

**fig = Figure(figsize=(8, 6))**

**scatter\_plot = fig.add\_subplot(111)**

**def predict\_package():**

**try:**

**cgpa = float(entry\_cgpa.get())**

**predicted\_package = lr\_model.predict([[cgpa]])**

**label\_prediction.config(text=f"Predicted Package: {predicted\_package[0][0]:.2f} LPA")**

**except ValueError:**

**label\_prediction.config(text="Please enter a valid CGPA")**

**label\_cgpa = tk.Label(root, text="Enter CGPA:")**

**label\_cgpa.pack(pady=10)**

**entry\_cgpa = tk.Entry(root, width=10)**

**entry\_cgpa.pack()**

**button\_predict = tk.Button(root, text="Predict Package", command=predict\_package)**

**button\_predict.pack(pady=10)**

**label\_prediction = tk.Label(root, text="")**

**label\_prediction.pack(pady=10)**

**scatter\_plot.scatter(df['cgpa'], df['package'])**

**scatter\_plot.plot(X\_train, lr\_model.predict(np.array(X\_train).reshape(-1,1)), color='orange')**

**scatter\_plot.set\_xlabel('CGPA')**

**scatter\_plot.set\_ylabel('Package')**

**scatter\_plot.set\_title('Package Prediction based on CGPA By Rohitkumar Pandey')**

**canvas = FigureCanvasTkAgg(fig, master=root)**

**canvas.draw()**

**canvas.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)**

**tk.mainloop()**

**# Postlab 2**

**import matplotlib.pyplot as plt**

**import pandas as pd**

**import numpy as np**

**df = pd.read\_csv('./placement.csv')**

**df.head()**

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

**# Getting cgpa values as X and pacakge as Y**

**X, Y= df.iloc[:,0], df.iloc[:,1]**

**X\_train,X\_test,Y\_train,Y\_test = train\_test\_split(X,Y,test\_size=0.2,random\_state=0)**

**from sklearn.linear\_model import LinearRegression**

**lr\_model = LinearRegression() # Creating the model**

**lr\_model.fit(np.array(X\_train).reshape(-1,1),Y\_train) # Fitting the model**

**# To show the errors in prediction MAE, MSE etc**

**import tkinter as tk**

**from matplotlib.figure import Figure**

**from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg**

**from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error, r2\_score**

**root = tk.Tk()**

**root.title("Package (in LPA) Prediction")**

**root.geometry('800x600')**

**fig = Figure(figsize=(8, 6))**

**scatter\_plot = fig.add\_subplot(111)**

**def predict\_package():**

**try:**

**cgpa = float(entry\_cgpa.get())**

**predicted\_package = lr\_model.predict([[cgpa]])**

**label\_prediction.config(text=f"Predicted Package: {predicted\_package[0]:.2f} LPA")**

**show\_errors()**

**except ValueError:**

**label\_prediction.config(text="Please enter a valid CGPA")**

**def show\_errors():**

**y\_pred = lr\_model.predict(np.array(X\_test).reshape(-1, 1))**

**# Calculate metrics**

**mse = mean\_squared\_error(Y\_test, y\_pred)**

**mae = mean\_absolute\_error(Y\_test, y\_pred)**

**r2 = r2\_score(Y\_test, y\_pred)**

**rmse = np.sqrt(mse)**

**error\_text = f"Regression Metrics:- \n"\**

**f"Mean Squared Error (MSE): {mse:.2f}\n" \**

**f"Mean Absolute Error (MAE): {mae:.2f}\n" \**

**f"R-squared (R2): {r2:.2f}\n" \**

**f"Root Mean Squared Error (RMSE): {rmse:.2f}"**

**label\_errors.config(text=error\_text)**

**label\_cgpa = tk.Label(root, text="Enter CGPA:")**

**label\_cgpa.pack(pady=10)**

**entry\_cgpa = tk.Entry(root, width=10)**

**entry\_cgpa.pack()**

**button\_predict = tk.Button(root, text="Predict Package", command=predict\_package)**

**button\_predict.pack(pady=10)**

**label\_prediction = tk.Label(root, text="")**

**label\_prediction.pack(pady=10)**

**label\_errors = tk.Label(root, text="")**

**label\_errors.pack(pady=10)**

**scatter\_plot.scatter(df['cgpa'], df['package'])**

**scatter\_plot.plot(X\_train, lr\_model.predict(np.array(X\_train).reshape(-1,1)), color='orange')**

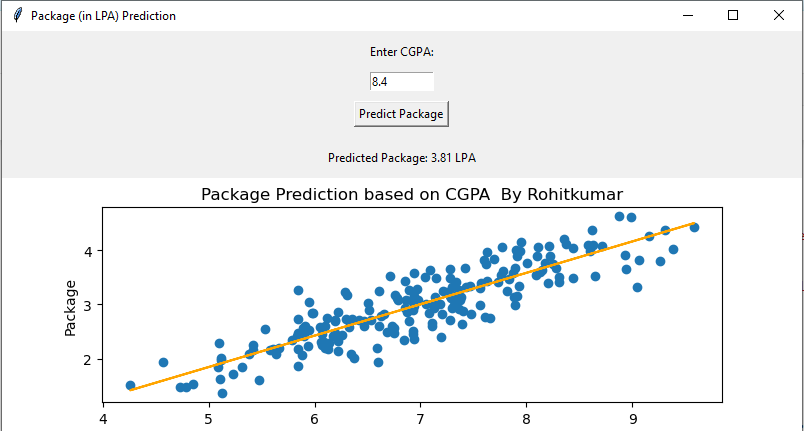
**scatter\_plot.set\_xlabel('CGPA')**

**scatter\_plot.set\_ylabel('Package')**

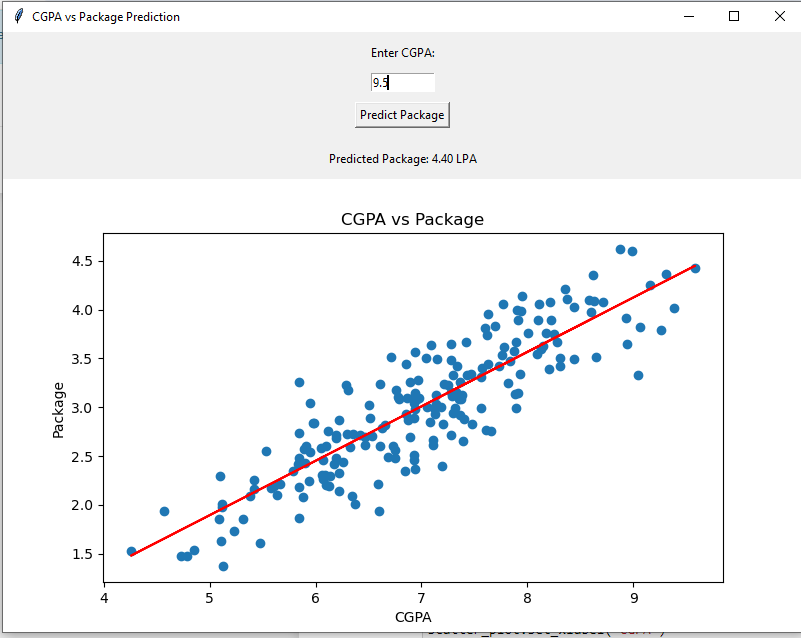
**scatter\_plot.set\_title('Package Prediction based on CGPA By Rohitkumar Pandey')**

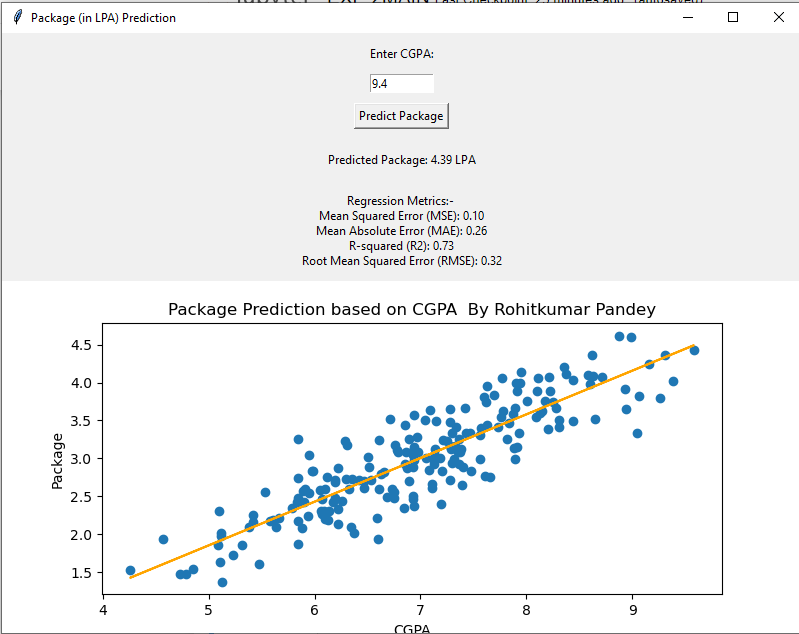
**canvas = FigureCanvasTkAgg(fig, master=root)**

**canvas.draw()**

**canvas.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)**

**tk.mainloop()**





Exp-3

**import tkinter as tk**

**import pandas as pd**

**data=pd.read\_csv('add.csv')**

**x=data[['x','y']]**

**y=data['sum']**

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

**x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.33,random\_state=42)**

**from sklearn.linear\_model import LinearRegression**

**model=LinearRegression()**

**model.fit(x\_train,y\_train)**

**def display\_inputs():**

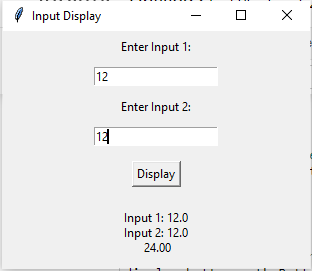
**# Retrieve inputs from the entry widgets**

**input1 = float(entry1.get())**

**input2 = float(entry2.get())**

**y\_pred=model.predict([[input1,input2]])[0]**

**# Display the inputs in the label widget**

**result\_label.config(text=f"Input 1: {input1}\nInput 2: {input2}\n {y\_pred:.2f}")**

**# Create the main window**

**root = tk.Tk()**

**root.title("Input Display")**

**# Create and place the first label and entry widget**

**label1 = tk.Label(root, text="Enter Input 1:")**

**label1.pack(pady=5)**

**entry1 = tk.Entry(root)**

**entry1.pack(pady=5)**

**# Create and place the second label and entry widget**

**label2 = tk.Label(root, text="Enter Input 2:")**

**label2.pack(pady=5)**

**entry2 = tk.Entry(root)**

**entry2.pack(pady=5)**

**# Create and place the display button**

**display\_button = tk.Button(root, text="Display", command=display\_inputs)**

**display\_button.pack(pady=10)**

**# Create and place the result label**

**result\_label = tk.Label(root, text="")**

**result\_label.pack(pady=10)**

**# Start the Tkinter event loop**

**root.mainloop()**

**#postlab1**

**import numpy as np**

**import tkinter as tk**

**from tkinter import ttk**

**from sklearn.datasets import load\_diabetes**

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

**from sklearn.metrics import r2\_score**

**#211P002**

**data = load\_diabetes()**

**X = data.data[:, :2]**

**y = data.target**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.33, random\_state=42)**

**class MyLinearRegression:**

**def \_\_init\_\_(self):**

**self.intercept\_ = None**

**self.coef\_ = None**

**def fit(self, X\_train, y\_train):**

**X\_b = np.c\_[np.ones((X\_train.shape[0], 1)), X\_train]**

**self.coef\_ = np.linalg.inv(X\_b.T @ X\_b) @ X\_b.T @ y\_train**

**self.intercept\_ = self.coef\_[0]**

**self.coef\_ = self.coef\_[1:]**

**def predict(self, X\_test):**

**X\_b = np.c\_[np.ones((X\_test.shape[0], 1)), X\_test]**

**return X\_b @ np.concatenate(([self.intercept\_], self.coef\_))**

**my\_model = MyLinearRegression()**

**my\_model.fit(X\_train, y\_train)**

**y\_pred\_my = my\_model.predict(X\_test)**

**r2\_my = r2\_score(y\_test, y\_pred\_my)**

**intercept\_my = my\_model.intercept\_**

**coefficients\_my = my\_model.coef\_**

**class RegressionApp:**

**def \_\_init\_\_(self, root):**

**self.root = root**

**self.root.title("Multiple Linear Regression By 222P005")**

**self.create\_widgets()**

**def create\_widgets(self):**

**ttk.Label(self.root, text="Feature 1:").grid(column=0, row=0, padx=10, pady=10)**

**self.feature1\_var = tk.DoubleVar()**

**self.feature1\_entry = ttk.Entry(self.root, textvariable=self.feature1\_var)**

**self.feature1\_entry.grid(column=1, row=0, padx=10, pady=10)**

**ttk.Label(self.root, text="Feature 2:").grid(column=0, row=1, padx=10, pady=10)**

**self.feature2\_var = tk.DoubleVar()**

**self.feature2\_entry = ttk.Entry(self.root, textvariable=self.feature2\_var)**

**self.feature2\_entry.grid(column=1, row=1, padx=10, pady=10)**

**self.calc\_button = ttk.Button(self.root, text="Calculate", command=self.calculate)**

**self.calc\_button.grid(column=0, row=4, columnspan=2, pady=10)**

**# Result Labels**

**ttk.Label(self.root, text="Prediction:").grid(column=0, row=5, padx=10, pady=10)**

**self.result\_var = tk.StringVar()**

**self.result\_label = ttk.Label(self.root, textvariable=self.result\_var)**

**self.result\_label.grid(column=1, row=5, padx=10, pady=10)**

**# Custom Model Info**

**ttk.Label(self.root, text="R² Score:").grid(column=0, row=6, padx=10, pady=10)**

**self.r2\_var = tk.StringVar(value=f"{r2\_my:.4f}")**

**self.r2\_label = ttk.Label(self.root, textvariable=self.r2\_var)**

**self.r2\_label.grid(column=1, row=6, padx=10, pady=10)**

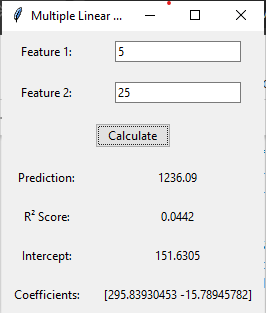
**ttk.Label(self.root, text="Intercept:").grid(column=0, row=7, padx=10, pady=10)**

**self.intercept\_var = tk.StringVar(value=f"{intercept\_my:.4f}")**

**self.intercept\_label = ttk.Label(self.root, textvariable=self.intercept\_var)**

**self.intercept\_label.grid(column=1, row=7, padx=10, pady=10)**

**ttk.Label(self.root, text="Coefficients:").grid(column=0, row=8, padx=10, pady=10)**

**self.coef\_var = tk.StringVar(value=str(coefficients\_my))**

**self.coef\_label = ttk.Label(self.root, textvariable=self.coef\_var)**

**self.coef\_label.grid(column=1, row=8, padx=10, pady=10)**

**def calculate(self):**

**feature1 = self.feature1\_var.get()**

**feature2 = self.feature2\_var.get()**

**features = np.array([[feature1, feature2]])**

**prediction = my\_model.predict(features)[0]**

**self.result\_var.set(f"{prediction:.2f}")**

**root = tk.Tk()**

**app = RegressionApp(root)**

**root.mainloop()**

**Exp4**

**import pandas as pd**

**import seaborn as sns**

**import matplotlib.pyplot as plt**

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

**from sklearn.linear\_model import LogisticRegression**

**from sklearn.metrics import confusion\_matrix**

**from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report**

**import tkinter as tk**

**from tkinter import ttk**

**import matplotlib.pyplot as plt**

**from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg**

**titanic\_data = pd.read\_csv('titanic.csv')**

**titanic\_data.head()**

**titanic\_data.isnull().sum()**

**titanic\_data['age'].fillna(titanic\_data['age'].mean(), inplace=True)**

**titanic\_data['cabin'].fillna(titanic\_data['cabin'].mode()[0], inplace=True)**

**titanic\_data['embarked'].fillna(titanic\_data['embarked'].mode()[0], inplace=True)**

**titanic\_data['fare'].fillna(titanic\_data['fare'].mean(), inplace=True)**

**titanic\_data.isnull().sum()**

**sns.countplot(x='survived', data=titanic\_data)**

**plt.show()**

**sns.countplot(x='survived', hue='sex', data=titanic\_data)**

**plt.show()**

**sns.countplot(x='survived', hue='pclass', data=titanic\_data)**

**plt.show()**

**titanic\_data['age'].plot.hist()**

**titanic\_data = pd.get\_dummies(titanic\_data, columns=['sex','embarked'])**

**titanic\_data.head()**

**titanic\_data.drop(['sex\_female', 'embarked\_C'], axis=1, inplace=True)**

**titanic\_data.head()**

**X = titanic\_data.drop('survived', axis=1).select\_dtypes(exclude=['object'])**

**y = titanic\_data['survived']**

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

**model = LogisticRegression()**

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

**predictions = model.predict(X\_test)**

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

**print("\nConfusion Matrix:\n", confusion\_matrix(y\_test, predictions))**

**print("\nClassification Report:\n", classification\_report(y\_test, predictions))**

**def plot\_graph():**

**hue\_choice = hue\_var.get()**

**sns.countplot(x='survived', hue=hue\_choice, data=titanic\_data)**

**plt.title(f'Survival Countplot by {hue\_choice.title()}')**

**canvas = FigureCanvasTkAgg(plt.gcf(), master=root)**

**canvas.draw()**

**canvas.get\_tk\_widget().grid(row=2, column=0, columnspan=2)**

**root = tk.Tk()**

**root.title("Titanic Survival Visualization")**

**hue\_var = tk.StringVar(value='pclass')**

**ttk.Label(root, text="Select Hue:").grid(row=0, column=0, padx=5, pady=5)**

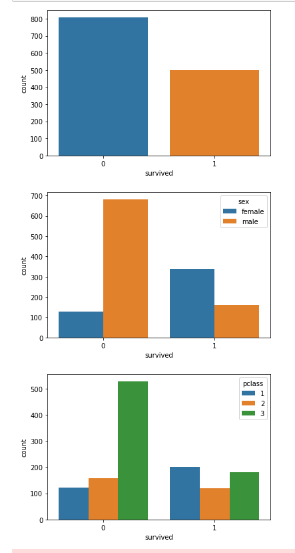
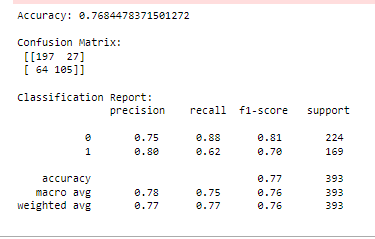
**ttk.Radiobutton(root, text="Age", variable=hue\_var, value='age').grid(row=0, column=1, sticky='w')**

**ttk.Radiobutton(root, text="Pclass", variable=hue\_var, value='pclass').grid(row=1, column=1, sticky='w')**

**plot\_button = ttk.Button(root, text="Plot Graph", command=plot\_graph)**

**plot\_button.grid(row=3, column=0, columnspan=2, pady=10)**

**root.mainloop()**

****

**#postlab1**

**import numpy as np**

**import matplotlib.pyplot as plt**

**from sklearn import datasets**

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

**from sklearn.linear\_model import LogisticRegression**

**from sklearn.metrics import accuracy\_score**

**import tkinter as tk**

**from tkinter import \***

**from tkinter import messagebox**

**from PIL import Image, ImageTk, ImageDraw**

**import matplotlib.pyplot as plt**

**from sklearn.datasets import load\_digits**

**# Load the digits dataset**

**digits = load\_digits()**

**X = digits.data**

**y = digits.target**

**# Split the dataset into training and test sets**

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

**# Create and train the logistic regression model**

**model = LogisticRegression(max\_iter=10000)**

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

**# Predict the test set**

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

**# Display accuracy**

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

**print(f'Accuracy: {accuracy:.4f}')**

**# Function to predict digit based on drawn image**

**def predict\_digit(img):**

**# Resize image to 8x8 pixels and convert to grayscale**

**img = img.resize((8, 8), Image.Resampling.LANCZOS).convert('L') # Use LANCZOS instead of ANTIALIAS**

**img = np.array(img)**

**# Invert the image color (as the model was trained on white digits on a black background)**

**img = 16 - (img / 16)**

**img = img.reshape(1, -1)**

**# Predict the digit**

**res = model.predict(img)**

**return res[0]**

**# GUI to draw and predict digit**

**class DigitApp:**

**def \_\_init\_\_(self, root):**

**self.root = root**

**self.root.title("Digit Recognizer by Rohitkumar Pandey")**

**self.root.geometry("400x400")**

**self.canvas = Canvas(self.root, width=200, height=200, bg="white")**

**self.canvas.pack(pady=20)**

**self.clear\_btn = Button(self.root, text="Clear", command=self.clear\_canvas)**

**self.clear\_btn.pack(side=LEFT, padx=20)**

**self.predict\_btn = Button(self.root, text="Predict", command=self.classify\_handwritten\_digit)**

**self.predict\_btn.pack(side=RIGHT, padx=20)**

**self.canvas.bind("<B1-Motion>", self.draw)**

**self.image = Image.new("RGB", (200, 200), (255, 255, 255))**

**self.draw\_img = ImageDraw.Draw(self.image)**

**def clear\_canvas(self):**

**self.canvas.delete("all")**

**self.image = Image.new("RGB", (200, 200), (255, 255, 255))**

**self.draw\_img = ImageDraw.Draw(self.image)**

**def draw(self, event):**

**x, y = event.x, event.y**

**r = 8**

**self.canvas.create\_oval(x - r, y - r, x + r, y + r, fill="black")**

**self.draw\_img.ellipse([x - r, y - r, x + r, y + r], fill="black")**

**def classify\_handwritten\_digit(self):**

**digit = predict\_digit(self.image)**

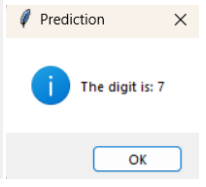
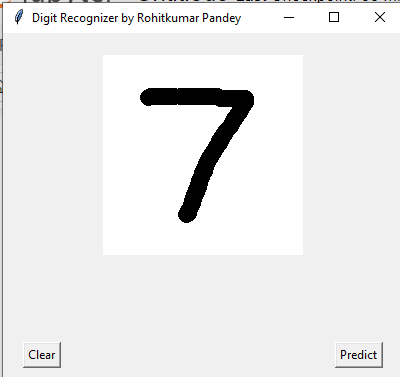
**messagebox.showinfo("Prediction", f"The digit is: {digit}")**

**# Create GUI window**

**root = Tk()**

**app = DigitApp(root)**

**root.mainloop()**

****

**Exp-5**

**# Importing the required libraries**

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**import warnings**

**warnings.filterwarnings("ignore")**

**# Reading the csv file and putting it into 'df' object.**

**df = pd.read\_csv('adult\_dataset.csv')**

**df.info()**

**df.head()**

**df.isin(['?']).sum(axis=0)**

**df=df.replace('?',np.nan)**

**df=df.dropna()**

**df.isin(['?']).sum(axis=0)**

**from sklearn.preprocessing import LabelEncoder**

**le=LabelEncoder()**

**df=df.apply(le.fit\_transform)**

**df.head()**

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

**from sklearn.tree import DecisionTreeClassifier,plot\_tree**

**from sklearn.metrics import accuracy\_score**

**x=df.drop('income',axis=1)**

**y=df['income']**

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

**clf=DecisionTreeClassifier(max\_depth=3)**

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

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

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

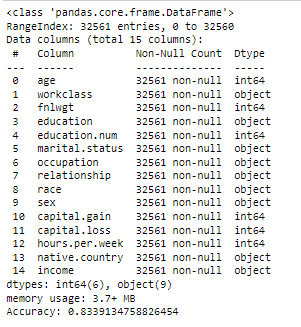
**print("Accuracy:",accuracy)**

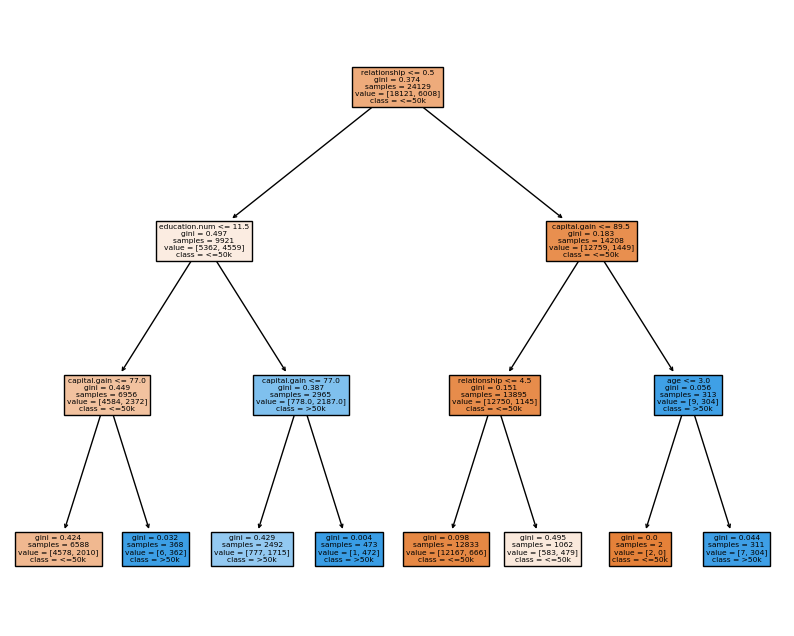
**plt.title("Rohitkumar Pandey -- 211P002")**

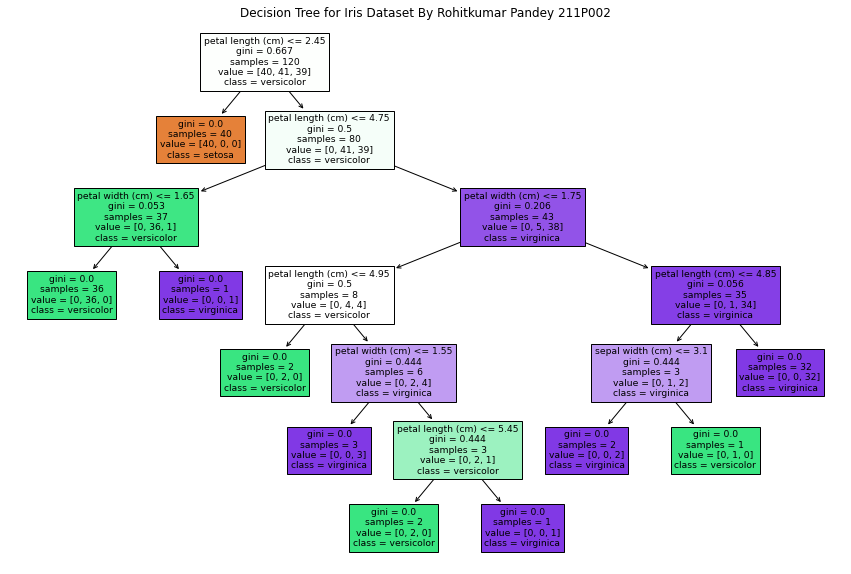
**plt.figure(figsize=(10,8))**

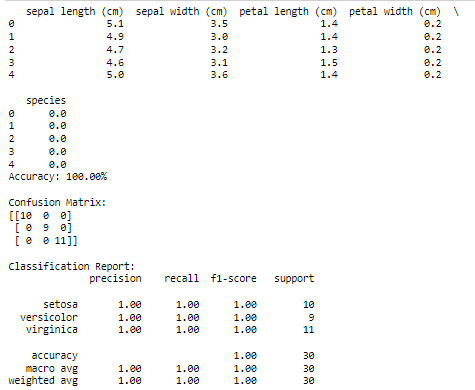
**plot\_tree(decision\_tree=clf,feature\_names=x.columns,class\_names=['<=50k','>50k'],filled=True)**

**plt.show()**



****

****

****

**#postlab 1**

**# Import necessary libraries**

**import numpy as np**

**import pandas as pd**

**from sklearn.datasets import load\_iris**

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

**from sklearn.tree import DecisionTreeClassifier**

**from sklearn import metrics**

**from sklearn.tree import plot\_tree**

**import matplotlib.pyplot as plt**

**# Load the Iris dataset**

**iris = load\_iris()**

**X = iris.data # Features (sepal length, sepal width, petal length, petalwidth)**

**y = iris.target # Labels (0: Iris-setosa, 1: Iris-versicolour, 2:Iris-virginica)**

**# Convert to a DataFrame for easy analysis (optional)**

**df = pd.DataFrame(data=np.c\_[X, y], columns=iris['feature\_names'] +['species'])**

**print(df.head()) # Show first few rows**

**# Split the dataset into training and test sets (80% training, 20%testing)**

**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(random\_state=42)**

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

**# Make predictions on the test set**

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

**# Evaluate the model**

**accuracy = metrics.accuracy\_score(y\_test, y\_pred)**

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

**# Display the confusion matrix**

**confusion\_matrix = metrics.confusion\_matrix(y\_test, y\_pred)**

**print("\nConfusion Matrix:")**

**print(confusion\_matrix)**

**# Display classification report (precision, recall, F1-score for each class)**

**classification\_report = metrics.classification\_report(y\_test, y\_pred,**

**target\_names=iris.target\_names)**

**print("\nClassification Report:")**

**print(classification\_report)**

**# Visualize the Decision Tree**

**plt.figure(figsize=(15, 10))**

**plot\_tree(clf, filled=True, feature\_names=iris.feature\_names,class\_names=iris.target\_names)**

**plt.title("Decision Tree for Iris Dataset By Rohitkumar Pandey 211P002")**

**plt.show()**

**# Feature importance analysis**

**feature\_importances = clf.feature\_importances\_**

**for name, importance in zip(iris.feature\_names, feature\_importances):**

**print(f"{name}: {importance:.2f}")**