Ml project code:

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

from sklearn.ensemble import RandomForestClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.svm import SVC

import pandas as pd

import numpy as np

import warnings

# Load the data

df = pd.read\_csv("C:/Users/HP/Downloads/admission.csv")

# Data cleaning

df = df.dropna()

df = df.drop\_duplicates()

# Prepare data and split into train and test sets

X = df[['AIEEE Rank', '12th Marks', '10th Marks']]

y = df['College']

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

# Train models

random\_forest = RandomForestClassifier(n\_estimators=100, random\_state=42)

random\_forest.fit(X\_train, y\_train)

decision\_tree = DecisionTreeClassifier(random\_state=42)

decision\_tree.fit(X\_train, y\_train)

logistic\_regression = LogisticRegression(max\_iter=50000)

logistic\_regression.fit(X\_train, y\_train)

svm\_classifier = SVC()

svm\_classifier.fit(X\_train, y\_train)

# Calculate accuracies

rf\_accuracy = random\_forest.score(X\_test, y\_test)

dt\_accuracy = decision\_tree.score(X\_test, y\_test)

lr\_accuracy = logistic\_regression.score(X\_test, y\_test)

svm\_accuracy = svm\_classifier.score(X\_test, y\_test)

# Print accuracies

print("Accuracy of Random Forest Classifier:", rf\_accuracy)

print("Accuracy of Decision Tree Classifier:", dt\_accuracy)

print("Accuracy of Logistic Regression Classifier:", lr\_accuracy)

print("Accuracy of SVM Classifier:", svm\_accuracy)

# Function to suggest college based on rank, 12th marks, and 10th marks

def suggest\_college(rank, marks\_12th, marks\_10th):

user\_data = np.array([[rank, marks\_12th, marks\_10th]])

predicted\_college = random\_forest.predict(user\_data)[0]

return predicted\_college

# Take input from the user

try:

user\_rank = int(input("Enter your AIEEE rank: "))

user\_12th\_marks = float(input("Enter your 12th marks: "))

user\_10th\_marks = float(input("Enter your 10th marks: "))

except ValueError:

print("Please enter valid numerical values for rank, 12th marks, and 10th marks.")

exit()

# Suggest college based on rank, 12th marks, and 10th marks

suggested\_college = suggest\_college(user\_rank, user\_12th\_marks, user\_10th\_marks)

# Print the suggested college

print(f"Based on your rank, 12th marks, and 10th marks, we suggest {suggested\_college}.")

import tkinter as tk

from tkinter import messagebox

from sklearn.ensemble import RandomForestClassifier

import pandas as pd

import numpy as np

# Load the data

df = pd.read\_csv("C:/Users/HP/Downloads/admission.csv")

# Data cleaning

df = df.dropna()

df = df.drop\_duplicates()

# Prepare data

X = df[['AIEEE Rank', '12th Marks', '10th Marks']]

y = df['College']

# Train Random Forest classifier

random\_forest = RandomForestClassifier(n\_estimators=100, random\_state=42)

random\_forest.fit(X, y)

# Function to suggest college based on rank, 12th marks, and 10th marks

def suggest\_college(rank, marks\_12th, marks\_10th):

user\_data = np.array([[rank, marks\_12th, marks\_10th]])

predicted\_college = random\_forest.predict(user\_data)[0]

return predicted\_college

# Function to handle button click event

def suggest\_college\_gui():

global rank\_entry, marks\_12th\_entry, marks\_10th\_entry # Declare entry widgets as global variables

try:

rank = int(rank\_entry.get())

marks\_12th = float(marks\_12th\_entry.get())

marks\_10th = float(marks\_10th\_entry.get())

except ValueError:

messagebox.showerror("Error", "Please enter valid numerical values for rank, 12th marks, and 10th marks.")

return

suggested\_college = suggest\_college(rank, marks\_12th, marks\_10th)

messagebox.showinfo("College Suggestion", f"We suggest {suggested\_college} based on your input.", icon='info')

# Create main window

window = tk.Tk()

window.title("College Suggestion")

window.configure(bg="#ADD8E6") # Set background color

window.geometry("800x400") # Set window size

# Create labels and entries for user input with centered text

label\_font = ("Arial", 14, "bold")

entry\_font = ("Arial", 12)

# Create labels and entries with centered text

labels = ["Enter your AIEEE rank:", "Enter your 12th marks:", "Enter your 10th marks:"]

entries = [tk.Entry(window, font=entry\_font) for \_ in range(3)]

for i, (label\_text, entry) in enumerate(zip(labels, entries)):

tk.Label(window, text=label\_text, font=label\_font, bg="#ADD8E6", anchor="center").grid(row=i, column=0, padx=10, pady=5)

entry.grid(row=i, column=1, padx=10, pady=5)

# Assign entry widgets to global variables

rank\_entry, marks\_12th\_entry, marks\_10th\_entry = entries

# Create button to suggest college with colorful styling

button\_font = ("Arial", 14, "bold")

suggest\_button = tk.Button(window, text="Suggest College", font=button\_font, bg="#FF6347", fg="#FFFFFF", command=suggest\_college\_gui)

suggest\_button.grid(row=3, column=0, columnspan=2, padx=10, pady=10, sticky="WE")

# Run the GUI

window.mainloop()