Practical No.02

Aim:- Implementation of placement Prediction model.

Program :-

```
import os
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
import tkinter as tk
from tkinter import messagebox, font
from PIL import Image, ImageTk
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg


# ---------- Load & Train Model ----------
file_path = r"C:\Users\ASUS\OneDrive\Desktop\capstone\2nd Practical\placementdata.xlsx"
if not os.path.exists(file_path):
    raise FileNotFoundError(f"❌ File not found: {file_path}")

data = pd.read_excel(file_path)
data['ExtracurricularActivities'] = data['ExtracurricularActivities'].map({'Yes': 1, 'No': 0})
data['PlacementTraining'] = data['PlacementTraining'].map({'Yes': 1, 'No': 0})
data['PlacementStatus'] = data['PlacementStatus'].map({'Placed': 1, 'NotPlaced': 0})

X = data.drop(columns=['StudentID', 'PlacementStatus'], errors='ignore')
y = data['PlacementStatus']

model = RandomForestClassifier(random_state=42)
model.fit(X, y)
features = X.columns.tolist()

field_hints = {
    "ExtracurricularActivities": "Enter Yes or No",
    "PlacementTraining": "Enter Yes or No",
    "CGPA": "Enter CGPA (0–10)",
    "SSC_Marks": "Enter Marks out of 100",
```

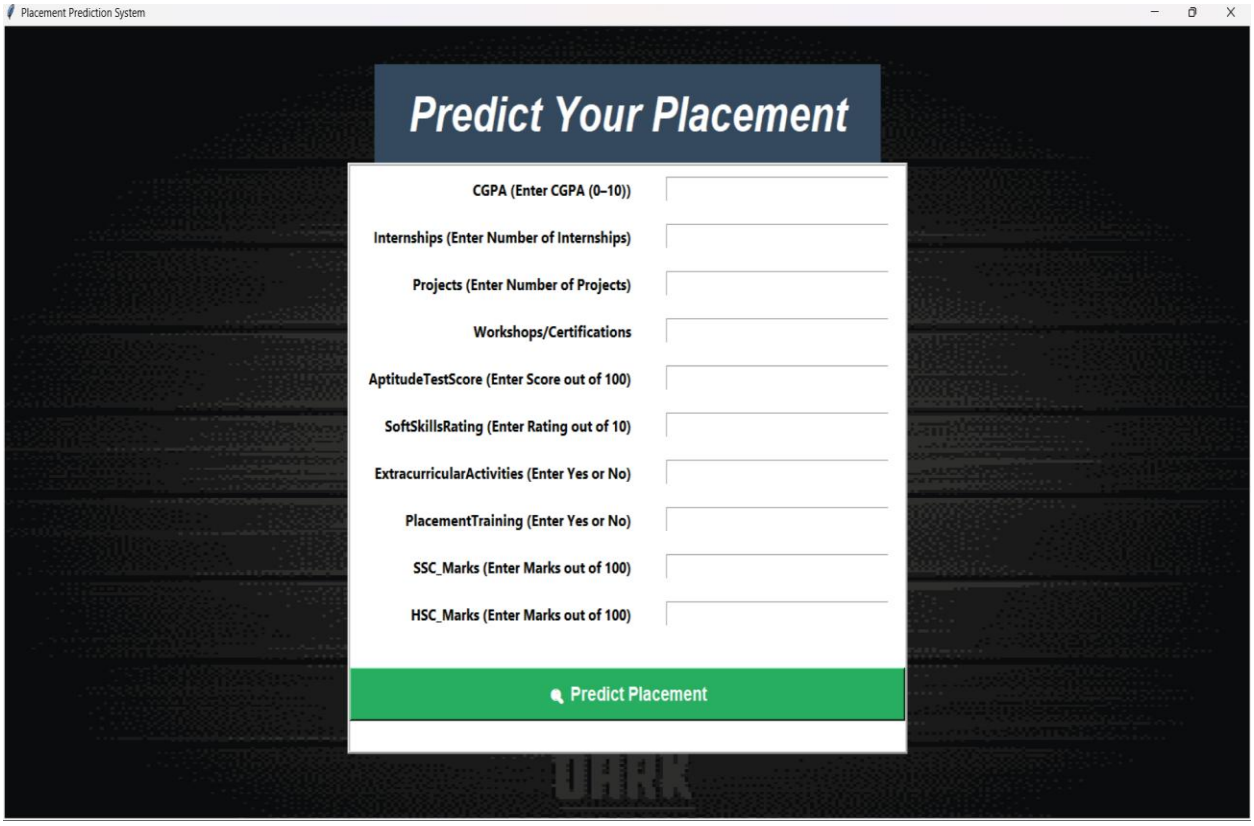
```
    "HSC_Marks": "Enter Marks out of 100",
    "AptitudeTestScore": "Enter Score out of 100",
    "SoftSkillsRating": "Enter Rating out of 10",
    "Internships": "Enter Number of Internships",
    "Projects": "Enter Number of Projects",
    "Certifications": "Enter Number of Certifications"
}

def show_report_window(result_text, prob):
    # Create a new window on top of main window
    report_win = tk.Toplevel(root)
    report_win.title("Placement Prediction Report")
    report_win.geometry("480x400")
    report_win.configure(bg="white")

    # Display only result and chart (no entered details)
    result_label = tk.Label(report_win, text=f"Prediction: {result_text}\nChance of Placement: {prob:.2%}",
                            font=("Helvetica", 20, "bold"),
                            fg="#2c3e50", bg="white", pady=20)
    result_label.pack()

    # Create matplotlib figure for bar chart: "Placed" vs "Not Placed"
    fig, ax = plt.subplots(figsize=(3.5,2.5), dpi=100)
    labels = ['Placed', 'Not Placed']
    probabilities = [prob, 1 - prob]
    colors = ['#27ae60', '#c0392b']  # green and red

    bars = ax.bar(labels, probabilities, color=colors, width=0.5)
    ax.set_ylim(0, 1)
    ax.set_ylabel('Probability')
    ax.set_title('Placement Probability')
    ax.set_xticklabels(labels, fontsize=12)
    for bar, prob_val in zip(bars, probabilities):
        height = bar.get_height()
        ax.annotate(f'{prob_val:.1%}', xy=(bar.get_x() + bar.get_width() / 2, height),
                    xytext=(0, 4), textcoords="offset points", ha='center', va='bottom', fontsize=12)
```

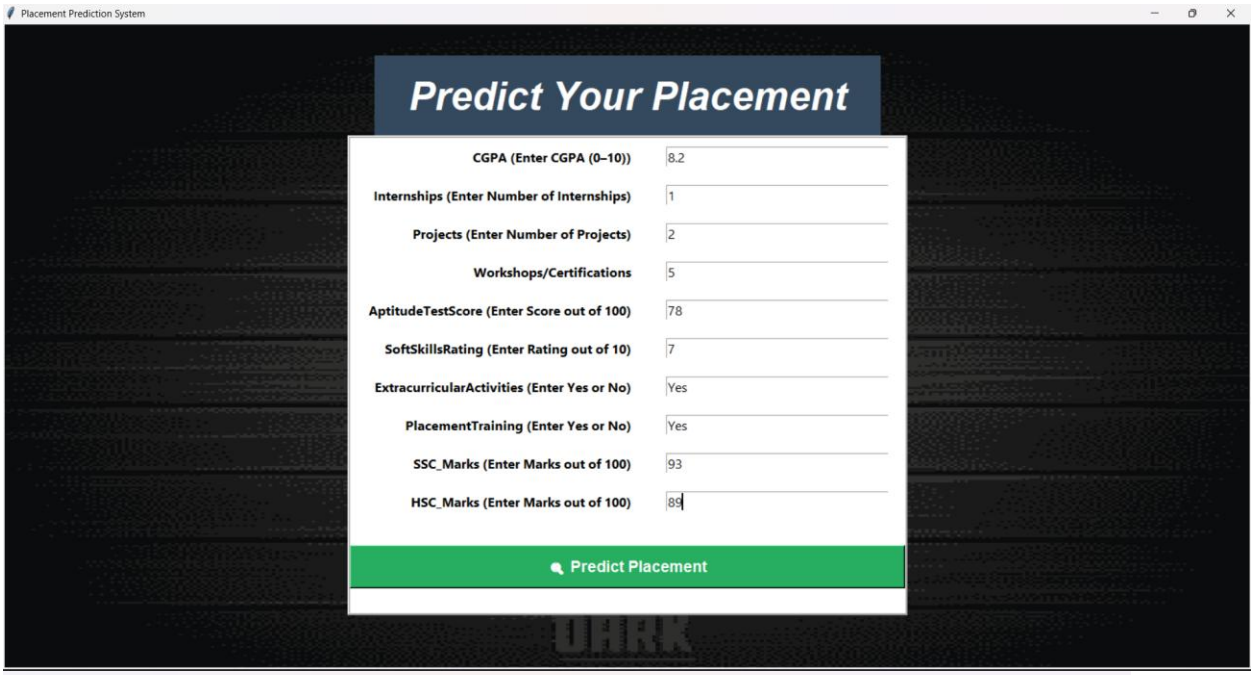
```
    fig.tight_layout()

    # Integrate matplotlib figure into Tkinter window
    canvas_fig = FigureCanvasTkAgg(fig, master=report_win)
    canvas_fig.draw()
    canvas_fig.get_tk_widget().pack(pady=10)

def predict():
    try:
        input_data = []
        for feature in features:
            val = entries[feature].get().strip()
            if val == "":
                messagebox.showerror("Input Error", f"⚠ Please enter a value for '{feature}'")
                return
            # Handle Yes/No fields
            if feature in ['ExtracurricularActivities', 'PlacementTraining']:
                if val.lower() == 'yes':
                    input_val = 1
                elif val.lower() == 'no':
                    input_val = 0
                else:
                    messagebox.showerror("Input Error", f"⚠ '{feature}' must be Yes or No")
                    return
            else:
                input_val = float(val)
            input_data.append(input_val)

        prediction = model.predict([input_data])[0]
        prob = model.predict_proba([input_data])[0][1]
        result_text = "✅ Placed" if prediction == 1 else "❌ Not Placed"

        # Show summary result and graph in new window
        show_report_window(result_text, prob)
```

```
    except ValueError:
        messagebox.showerror("Input Error", "⚠ Please enter numeric values where required.")

# ------------ Create main window ------------
root = tk.Tk()
root.title("Placement Prediction System")
root.state('zoomed')  # Full screen

screen_width = root.winfo_screenwidth()
screen_height = root.winfo_screenheight()

background_image_path = r"C:\Users\ASUS\OneDrive\Desktop\capstone\2nd Practical\background.webp"
bg_image = Image.open(background_image_path)
bg_image = bg_image.resize((screen_width, screen_height), Image.LANCZOS)
bg_photo = ImageTk.PhotoImage(bg_image)

canvas = tk.Canvas(root, width=screen_width, height=screen_height)
canvas.pack(fill="both", expand=True)
canvas.create_image(0, 0, image=bg_photo, anchor="nw")

# Large stylish heading at the top center
title_font = font.Font(family="Helvetica", size=36, weight="bold", slant="italic")
title_label = tk.Label(root, text="Predict Your Placement", bg="#34495e", fg="white", font=title_font, padx=40, pady=20)
canvas.create_window(screen_width // 2, 40, window=title_label, anchor="n")

# Center form below heading
form_frame = tk.Frame(root, bg='#ffffff', bd=3, relief="ridge")
form_x = screen_width // 2
form_y = screen_height // 2
canvas.create_window(form_x, form_y, window=form_frame, anchor="center")

label_font = font.Font(family="Segoe UI", size=12, weight="bold")
entry_font = font.Font(family="Segoe UI", size=12)
button_font = font.Font(family="Arial", size=14, weight="bold")

entries = {}
```

```
pad_y = 10

for i, feature in enumerate(features):
    hint = field_hints.get(feature, "")
    label_text = f"{feature} ({hint})" if hint else feature
    lbl = tk.Label(form_frame, text=label_text, font=label_font, bg="#ffffff", anchor="w")
    lbl.grid(row=i, column=0, padx=20, pady=pad_y, sticky="e")
    ent = tk.Entry(form_frame, font=entry_font, width=30)
    ent.grid(row=i, column=1, padx=20, pady=pad_y)
    entries[feature] = ent

predict_btn = tk.Button(form_frame, text="🔍 Predict Placement", font=button_font, bg="#27ae60", fg="white",
                        activebackground="#2ecc71", padx=10, pady=8, command=predict)
predict_btn.grid(row=len(features), column=0, columnspan=2, pady=30, sticky="ew")

root.mainloop()
```

Screenshots :-





Prediction: ☑ Placed

Chance of Placement: 78.00%