

**Engineering Exploration Project report on COLLEGE PREDICTOR**

# M G Shahbaz Jahan 160120733050

Under the mentorship of **Smt. Sri Devi** Associate Professor.

# Core branch: CSE 1

**Submitted to Computer Science Engineering Department**

**Chaitanya Bharathi Institute of Technology (A) July 2021**

**ABSTRACT**

This project Engineering Admission Predictor System is a Python-based application in which students can enter their rank and other details for the prediction of the 10 best colleges for admission and can predict the accuracy of the result by analyzing the previous year’s data. The accuracy of the predictions increases the more we use them. It is user-friendly, fast, efficient, and reliable. Using this software, a student can get a better idea about the colleges that he can join.

**ACKNOWLEDGEMENTS**

I would like to express my special thanks of gratitude to my mentor Smt. Sri Devi associate professor as well as our principal who gave me the golden opportunity to do this wonderful project on the topic COLLEGE PREDICTOR, which also helped me in doing a lot of research and I came to know about so many new things I am thankful to them.

**TABLE OF CONTENT:**

|  |  |  |
| --- | --- | --- |
| S No. | Description | Page No. |
| 1 | Motivation | 5 |
| 2 | Problem definition | 5 |
| 3 | Requirements specification and designing | 5-6 |
| 4 | Methodology/Mechanism & Design | 7 |
| 5 | Code | 8-44 |
| 6 | Future | 46 |
| 7 | References | 47 |

**Motivation:**

College predictor helped me to select a good college for my Engineering and also to compare the colleges that I wished to join both in Eamcet and mains. So I wanted to recreate a college predictor to understand the functioning of the predictor and make furthermore improvements in the predictor that a candidate aspiring to join a good college would be looking for.

**Problem Definition:**

Students face a lot of difficulties to secure admission to the college of their choice. The current scenario of an engineering admission process is a little complicated and not so easy in terms of selecting an appropriate college according to the scores and field of interest. Accurate choice, varying with the entrance exam result and academic scores, is very important to the candidates to fill in the application form. Many colleges are offering multiple engineering courses. So it becomes troublesome for students to organize and list out the proper colleges of their choice for courses according to their performance score. This project consists of a smart list generator working together with the help of a college predictor, to aid students in the admission process. The college admission predictor uses historical colleges cut-off student's admission data for predicting the most probable colleges. The system analyzes student academic merits, background, and college admission criteria. Based on that, it predicts the likelihood of a university college that a student may enter. The smart list generator would enable the student to prepare the list of colleges, which could be needed to be filled in during the admission process. The system would also get feedback from the users, which would prove helpful for prediction evaluation and improving the performance factor.

**Requirements Specification & Design**

**Hardware Requirements**

|  |  |
| --- | --- |
| **Number** | **Description** |
| 1 | PC with 250 GB or more Hard disk. |
| 2 | PC with 2 GB RAM. |
| 3 | PC with Pentium 1 and Above. |

**Software Requirements**

|  |  |  |
| --- | --- | --- |
| **Number** | **Description** | **Type** |
| 1 | Operating System | Windows XP / Windows |
| 2 | Language | Python |
| 3 | Files | Excel File |
| 4 | IDE | IDLE, python3 |
| 5 | Modules | Tkinter, openpyxl |

Python is a high-level, general-purpose, and very popular programming language. Python programming language (latest Python 3) is being used in web development, Machine Learning applications, along all cutting edge technology in Software Industry. Python Programming Language is very well suited for Beginners, also for experienced programmers with other programming languages like C++ and Java.

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Openpyxl module, which is used to deal with Excel files without involving third-party Microsoft application software. By using this module, we can have control over excel without opening the application. It is used to perform excel tasks such as reading data from an excel file or writing data to the excel file, drawing some charts, accessing excel sheet, renaming sheet, modification (adding and deleting) in excel sheet, formatting, styling in the sheet, and any other task. Openpyxl is very efficient to perform these tasks for you. Data scientists often use Openpyxl to perform different operations such as data copying to data mining as well as data analysis.

**Methodology/Mechanism & Design:**

This project is made possible using one of the object-oriented programming languages Python. Using classes in python the attributes are accessed in different methods to perform specific tasks. When the user runs the program a GUI appears and on the home page, the user is asked to select the examination the user wishes to know about or to know the colleges that he can join if he tries to go with that examination. After the user selects an examination a new window appears in which the candidate is asked to enter the details regarding rank, gender, caste, special abilities, and the courses in which the candidate is interested to join in. After the user enters all the details submit button is to be clicked. Now rank is confirmed to be a positive integer and if it fails the conditions a popup window appears saying error box saying invalid rank. If there are no colleges for the entered rank by the user a popup window pops saying the user couldn’t crack the exam. According to the information entered by the candidate, an excel file is opened using the module Openpyxl, and a list of colleges and their cut-offs for that particular category is selected based on the information entered by the user, all the colleges having cutoffs greater than the rank of the candidate are added to a new list and new dictionary is created with the key being the rank and the value being the college name. This list is sorted and the least 10 cut-offs are used as keys to call the values that are the college names are printed on the screen in a new window using the pack function. This list of colleges is the 10 best colleges that the user can try to join.

The accuracy of the result is calculated based on similar operations carried on the previous 3 years' data. The final list of all four years' data is compared to get the matching percentage of the result of the current year.

All the above steps are done in the GUI using Tkinter.

The accuracy of the predictor is predicted by comparing the result of the current year with the previous three years' list of colleges. The top 10 college list of all years is added to form a new list and then the list of current year is matched. If an element is repeated thrice that implies that the college has high probability of being a good possibility and contributes 4 out of 4. If an element is repeated twice then it contributes for 3 out of 4. If an element is repeated once then it contributes for 2 out of 4. If an element does not repeat then it contributes

for 1 out of 4. Now the average of all the values are taken out and percentage is calculated. This percentage is been stored in a text file which help to show the overall accuracy of the predictor.

**Code:**

from tkinter import \*

import openpyxl

from tkinter import messagebox

class predictor:

def homeabt(self):

return None

def my\_accuracy(self):

racc=Tk()

racc.geometry("400x150")

racc.title("Accuracy of Result")

lab=Label(racc,text="Accuracy of this result\n",font=(20))

lab.place(relx=0.29,rely=0.25)

self.lal=Label(racc,text=str((self.c/self.t)\*100),font=(20))

self.lal.place(relx=0.3,rely=0.45)

return None

def eabout(self):

with open("eaccuracy.txt","r") as f:

ac=f.readlines()

oacc=Tk()

oacc.title("About the predictor")

oacc.geometry("350x150")

self.lac1=Label(oacc,text="Over all accuracy is "+ac[0],font=('arial',13))

self.lac1.place(relx=0.1,rely=0.1)

self.lac2=Label(oacc,text="Number of results "+ac[1],font=('arial',13))

self.lac2.place(relx=0.1,rely=0.5)

return None

def jabout(self):

with open("jaccuracy.txt","r") as f:

ac=f.readlines()

oacc=Tk()

oacc.title("About the predictor")

oacc.geometry("350x150")

self.lac1=Label(oacc,text="Over all accuracy is "+ac[0],font=('arial',13))

self.lac1.place(relx=0.1,rely=0.1)

self.lac2=Label(oacc,text="Number of results "+ac[1],font=('arial',13))

self.lac2.place(relx=0.1,rely=0.5)

return None

def ecalculate(self):

self.ltotal=[]

self.dtotal={}

g=self.r.get()

i=self.course.get()

r=self.er.get()

try:

r=int(r)

except ValueError:

messagebox.showerror("Error","Invalid rank")

#2018 calculations done frm here

if(r>0):

flag=0

c=self.category.get()

wb=openpyxl.load\_workbook("eamcet2018.xlsx")

s=wb['Table 1']

if(g==1 and c=='OC'):

m='J'

if(g==2 and c=='OC'):

m='I'

if(g==1 and c=='BC-A'):

m='L'

if(g==2 and c=='BC-A'):

m='K'

if(g==1 and c=='BC-B'):

m='N'

if(g==2 and c=='BC-B'):

m='M'

if(g==1 and c=='BC-C'):

m='P'

if(g==2 and c=='BC-C'):

m='O'

if(g==1 and c=='BC-D'):

m='R'

if(g==2 and c=='BC-D'):

m='Q'

if(g==1 and c=='BC-E'):

m='T'

if(g==2 and c=='BC-E'):

m='S'

if(g==1 and c=='SC'):

m='V'

if(g==2 and c=='SC'):

m='U'

if(g==1 and c=='ST'):

m='X'

if(g==2 and c=='ST'):

m='W'

d2018={}

l2018=[]

for a in range(3,833):

j=m+str(a)

co=s[j].value

if (co=='NA'):

continue

j="B"+str(a)

n=s[j].value

j='G'+str(a)

p=s[j].value

if(p==i and r<co):

d2018[co]=n

l2018.append(co)

flag=1

l2018.sort()

#2019 calculations done frm here

if(r>0):

flag=0

c=self.category.get()

wb=openpyxl.load\_workbook("eamcet2019.xlsx")

s=wb['Table 1']

if(g==1 and c=='OC'):

m='J'

if(g==2 and c=='OC'):

m='I'

if(g==1 and c=='BC-A'):

m='L'

if(g==2 and c=='BC-A'):

m='K'

if(g==1 and c=='BC-B'):

m='N'

if(g==2 and c=='BC-B'):

m='M'

if(g==1 and c=='BC-C'):

m='P'

if(g==2 and c=='BC-C'):

m='O'

if(g==1 and c=='BC-D'):

m='R'

if(g==2 and c=='BC-D'):

m='Q'

if(g==1 and c=='BC-E'):

m='T'

if(g==2 and c=='BC-E'):

m='S'

if(g==1 and c=='SC'):

m='V'

if(g==2 and c=='SC'):

m='U'

if(g==1 and c=='ST'):

m='X'

if(g==2 and c=='ST'):

m='W'

d2019={}

l2019=[]

for a in range(3,833):

j=m+str(a)

co=s[j].value

if (co=='NA'):

continue

j="B"+str(a)

n=s[j].value

j='G'+str(a)

p=s[j].value

if(p==i and r<co):

d2019[co]=n

l2019.append(co)

flag=1

l2019.sort()

#2017 calculations done frm here

if(r>0):

flag=0

c=self.category.get()

wb=openpyxl.load\_workbook("eamcet2017.xlsx")

s=wb['Table 1']

if(g==1 and c=='OC'):

m='J'

if(g==2 and c=='OC'):

m='I'

if(g==1 and c=='BC-A'):

m='L'

if(g==2 and c=='BC-A'):

m='K'

if(g==1 and c=='BC-B'):

m='N'

if(g==2 and c=='BC-B'):

m='M'

if(g==1 and c=='BC-C'):

m='P'

if(g==2 and c=='BC-C'):

m='O'

if(g==1 and c=='BC-D'):

m='R'

if(g==2 and c=='BC-D'):

m='Q'

if(g==1 and c=='BC-E'):

m='T'

if(g==2 and c=='BC-E'):

m='S'

if(g==1 and c=='SC'):

m='V'

if(g==2 and c=='SC'):

m='U'

if(g==1 and c=='ST'):

m='X'

if(g==2 and c=='ST'):

m='W'

d2017={}

l2017=[]

for a in range(3,833):

j=m+str(a)

co=s[j].value

if (co=='NA'):

continue

j="B"+str(a)

n=s[j].value

j='G'+str(a)

p=s[j].value

if(p==i and r<co):

d2017[co]=n

l2017.append(co)

flag=1

l2017.sort()

#2016 calculations done frm here

if(r>0):

flag=0

c=self.category.get()

wb=openpyxl.load\_workbook("eamcet2016.xlsx")

s=wb['Table 1']

if(g==1 and c=='OC'):

m='J'

if(g==2 and c=='OC'):

m='I'

if(g==1 and c=='BC-A'):

m='L'

if(g==2 and c=='BC-A'):

m='K'

if(g==1 and c=='BC-B'):

m='N'

if(g==2 and c=='BC-B'):

m='M'

if(g==1 and c=='BC-C'):

m='P'

if(g==2 and c=='BC-C'):

m='O'

if(g==1 and c=='BC-D'):

m='R'

if(g==2 and c=='BC-D'):

m='Q'

if(g==1 and c=='BC-E'):

m='T'

if(g==2 and c=='BC-E'):

m='S'

if(g==1 and c=='SC'):

m='V'

if(g==2 and c=='SC'):

m='U'

if(g==1 and c=='ST'):

m='X'

if(g==2 and c=='ST'):

m='W'

d2016={}

l2016=[]

for a in range(3,833):

j=m+str(a)

co=s[j].value

if (co=='NA'):

continue

j="B"+str(a)

n=s[j].value

j='G'+str(a)

p=s[j].value

if(p==i and r<co):

d2016[co]=n

l2016.append(co)

flag=1

if(flag==0):

messagebox.showerror("Error","You could not crack EAMCET\nBetter luck next time")

l2016.sort()

for x in range(10):

if(x==len(l2018)):

break

self.ltotal.append(d2018[l2018[x]])

for x in range(10):

if(x==len(l2019)):

break

self.ltotal.append(d2019[l2019[x]])

for x in range(10):

if(x==len(l2017)):

break

self.ltotal.append(d2017[l2017[x]])

for x in range(10):

if(x==len(l2016)):

break

self.ltotal.append(d2016[l2016[x]])

self.c=0

self.t=0

for x in range(10):

if(x==len(l2019)):

break

self.c=self.c+self.ltotal.count(d2019[l2019[x]])

self.t=self.t+4

for x in range(10):

if(x==len(l2018)):

break

self.c=self.c+self.ltotal.count(d2018[l2018[x]])

self.t=self.t+4

for x in range(10):

if(x==len(l2017)):

break

self.c=self.c+self.ltotal.count(d2017[l2017[x]])

self.t=self.t+4

for x in range(10):

if(x==len(l2016)):

break

self.c=self.c+self.ltotal.count(d2016[l2016[x]])

self.t=self.t+4

with open("eaccuracy.txt","r") as f:

ac=f.readlines()

ac[0]=float(ac[0])

ac[1]=float(ac[1])

ac[0]=((self.c/self.t)\*100+(ac[1]\*ac[0]))/(ac[1]+1)

ac[1]=ac[1]+1

with open("eaccuracy.txt","w") as f:

f.write(str(ac[0])+'\n')

f.write(str(ac[1]))

if(flag==0):

messagebox.showerror("Error","You could not crack EAMCET\nBetter luck next time")

else:

rclg=Tk()

rclg.title("List of colleges")

rclg.geometry("750x600")

if(len(l2018)>=1):

c0=Button(rclg,text=d2018[l2018[0]],padx=300,pady=15)

c0.pack()

if(len(l2018)>=2):

c1=Button(rclg,text=d2018[l2018[1]],padx=300,pady=15)

c1.pack()

if(len(l2018)>=3):

c2=Button(rclg,text=d2018[l2018[2]],padx=300,pady=15)

c2.pack()

if(len(l2018)>=4):

c3=Button(rclg,text=d2018[l2018[3]],padx=300,pady=15)

c3.pack()

if(len(l2018)>=5):

c4=Button(rclg,text=d2018[l2018[4]],padx=300,pady=15)

c4.pack()

if(len(l2018)>=6):

c5=Button(rclg,text=d2018[l2018[5]],padx=300,pady=15)

c5.pack()

if(len(l2018)>=7):

c6=Button(rclg,text=d2018[l2018[6]],padx=300,pady=15)

c6.pack()

if(len(l2018)>=8):

c7=Button(rclg,text=d2018[l2018[7]],padx=300,pady=15)

c7.pack()

if(len(l2018)>=9):

c8=Button(rclg,text=d2018[l2018[8]],padx=300,pady=15)

c8.pack()

if(len(l2018)>=10):

c9=Button(rclg,text=d2018[l2018[9]],padx=300,pady=15)

c9.pack()

l=Label(rclg,text=" ")

l.pack()

button\_acc=Button(rclg,text="click here to check accuracy of the result",fg="white",bg="grey",command=self.my\_accuracy)

button\_acc.pack()

else:

messagebox.showerror("Error","Invalid rank")

return None

def jcalculate(self):

self.ltotal=[]

self.dtotal={}

g=self.r.get()

g1="Female-only (including Supernumerary)"

g2="Gender-Neutral"

i=self.course.get()

flag=0

r=self.er.get()

try:

r=int(r)

except ValueError:

messagebox.showerror("Error","Invalid rank")

if(r>0):

c=self.category.get()

wb=openpyxl.load\_workbook("jee1.xlsx")

s=wb['Sheet1']

l1=[]

d1={}

for a in range(2,9339):

j='G'+str(a)

co=s[j].value

j='B'+str(a)

p=s[j].value

j='D'+str(a)

q=s[j].value

j='E'+str(a)

u=s[j].value

r=int(r)

if (type(co)!=int):

continue

if(co>r and p==i and c==q and g==1):

if(u==g2 or u==g1):

k='A'+str(a)

n=s[k].value

d1[co]=n

l1.append(co)

flag=1

else:

if(co>r and p==i and c==q and g==2):

if(u==g2):

k='A'+str(a)

n=s[k].value

d1[co]=n

l1.append(co)

flag=1

l1.sort()

if(r>0):

c=self.category.get()

wb=openpyxl.load\_workbook("jee2.xlsx")

s=wb['Sheet1']

l2=[]

d2={}

for a in range(2,9339):

j='G'+str(a)

co=s[j].value

j='B'+str(a)

p=s[j].value

j='D'+str(a)

q=s[j].value

j='E'+str(a)

u=s[j].value

r=int(r)

if (type(co)!=int):

continue

if(co>r and p==i and c==q and g==1):

if(u==g2 or u==g1):

k='A'+str(a)

n=s[k].value

d2[co]=n

l2.append(co)

flag=1

else:

if(co>r and p==i and c==q and g==2):

if(u==g2):

k='A'+str(a)

n=s[k].value

d2[co]=n

l2.append(co)

flag=1

l2.sort()

if(r>0):

c=self.category.get()

wb=openpyxl.load\_workbook("jee3.xlsx")

s=wb['Sheet1']

l3=[]

d3={}

for a in range(2,9339):

j='G'+str(a)

co=s[j].value

j='B'+str(a)

p=s[j].value

j='D'+str(a)

q=s[j].value

j='E'+str(a)

u=s[j].value

r=int(r)

if (type(co)!=int):

continue

if(co>r and p==i and c==q and g==1):

if(u==g2 or u==g1):

k='A'+str(a)

n=s[k].value

d3[co]=n

l3.append(co)

flag=1

else:

if(co>r and p==i and c==q and g==2):

if(u==g2):

k='A'+str(a)

n=s[k].value

d3[co]=n

l3.append(co)

flag=1

l3.sort()

if(r>0):

c=self.category.get()

wb=openpyxl.load\_workbook("jee4.xlsx")

s=wb['Sheet1']

l4=[]

d4={}

for a in range(2,9339):

j='G'+str(a)

co=s[j].value

j='B'+str(a)

p=s[j].value

j='D'+str(a)

q=s[j].value

j='E'+str(a)

u=s[j].value

r=int(r)

if (type(co)!=int):

continue

if(co>r and p==i and c==q and g==1):

if(u==g2 or u==g1):

k='A'+str(a)

n=s[k].value

d4[co]=n

l4.append(co)

flag=1

else:

if(co>r and p==i and c==q and g==2):

if(u==g2):

k='A'+str(a)

n=s[k].value

d4[co]=n

l4.append(co)

flag=1

if(flag==0):

messagebox.showerror("Error","You could not crack JEE\nBetter luck next time")

l4.sort()

for x in range(10):

if(x==len(l1)):

break

self.ltotal.append(d1[l1[x]])

for x in range(10):

if(x==len(l2)):

break

self.ltotal.append(d2[l2[x]])

for x in range(10):

if(x==len(l3)):

break

self.ltotal.append(d3[l3[x]])

for x in range(10):

if(x==len(l4)):

break

self.ltotal.append(d4[l4[x]])

self.c=0

self.t=0

for x in range(10):

if(x==len(l1)):

break

self.c=self.c+self.ltotal.count(d1[l1[x]])

self.t=self.t+4

for x in range(10):

if(x==len(l2)):

break

self.c=self.c+self.ltotal.count(d2[l2[x]])

self.t=self.t+4

for x in range(10):

if(x==len(l3)):

break

self.c=self.c+self.ltotal.count(d3[l3[x]])

self.t=self.t+4

for x in range(10):

if(x==len(l4)):

break

self.c=self.c+self.ltotal.count(d4[l4[x]])

self.t=self.t+4

with open("jaccuracy.txt","r") as f:

ac=f.readlines()

ac[0]=float(ac[0])

ac[1]=float(ac[1])

ac[0]=((self.c/self.t)\*100+(ac[1]\*ac[0]))/(ac[1]+1)

ac[1]=ac[1]+1

with open("jaccuracy.txt","w") as f:

f.write(str(ac[0])+'\n')

f.write(str(ac[1]))

if(flag==0):

messagebox.showerror("Error","You could not crack JEE\nBetter luck next time")

rclg=Tk()

rclg.title("List of colleges")

rclg.geometry("750x600")

if(len(l1)>=1):

c0=Button(rclg,text=d1[l1[0]],padx=300,pady=15)

c0.pack()

if(len(l1)>=2):

c1=Button(rclg,text=d1[l1[1]],padx=300,pady=15)

c1.pack()

if(len(l1)>=3):

c2=Button(rclg,text=d1[l1[2]],padx=300,pady=15)

c2.pack()

if(len(l1)>=4):

c3=Button(rclg,text=d1[l1[3]],padx=300,pady=15)

c3.pack()

if(len(l1)>=5):

c4=Button(rclg,text=d1[l1[4]],padx=300,pady=15)

c4.pack()

if(len(l1)>=6):

c5=Button(rclg,text=d1[l1[5]],padx=300,pady=15)

c5.pack()

if(len(l1)>=7):

c6=Button(rclg,text=d1[l1[6]],padx=300,pady=15)

c6.pack()

if(len(l1)>=8):

c7=Button(rclg,text=d1[l1[7]],padx=300,pady=15)

c7.pack()

if(len(l1)>=9):

c8=Button(rclg,text=d1[l1[8]],padx=300,pady=15)

c8.pack()

if(len(l1)>=10):

c9=Button(rclg,text=d1[l1[9]],padx=300,pady=15)

c9.pack()

l=Label(rclg,text=" ")

l.pack()

button\_acc=Button(rclg,text="click here to check accuracy of the result",padx=80,pady=20,fg="white",bg="grey",font=("bold"),command=self.my\_accuracy)

button\_acc.pack()

else:

messagebox.showerror("Error","Invalid rank")

return None

def jee(self):

self.root.destroy()

self.root=Tk()

self.root.geometry("+-10+0")

self.root.geometry("1700x800")

self.frame1=Frame(self.root,bg="white",width=1700,height=775)

self.frame1.place(relx=0,rely=0.05)

self.root.title("JEE Predictions")

self.l1=Label(self.root,text="Giving your exam rank will help us recommend you better colleges. If you don't have actual rank, then enter expected rank.",bg="white",font=("arial",12,"bold"))

self.l2=Label(self.root,text="Enter your rank",bg="white",font=("arial",13,"bold"))

self.l3=Label(self.root,text="Enter your category",bg="white",font=("arial",13,"bold"))

self.l4=Label(self.root,text="Enter your course",bg="white",font=("arial",13,"bold"))

self.l5=Label(self.root,text="Enter your gender",bg="white",font=("arial",13,"bold"))

self.l6=Label(self.root,text="Are you specially abled",bg="white",font=("arial",13,"bold"))

self.l7=Label(self.root,text=" ")

self.l8=Label(self.root,text=" ")

self.l9=Label(self.root,text=" ")

self.l10=Label(self.root,text=" ")

self.l11=Label(self.root,text=" ")

self.l12=Label(self.root,text=" ")

self.button\_e=Button(self.root,text="EAMCET",padx=160,pady=20,bg="white",font=("arial",15,"bold"),command= lambda:self.eamcet())

self.button\_j=Button(self.root,text="JEE MAINS",padx=160,pady=20,fg="white",bg="#008080",font=("arial",15,"bold"),command= lambda:self.jee())

self.button\_e.place(relx=0.2,rely=0.05)

self.button\_j.place(relx=0.5,rely=0.05)

self.l1.place(relx=0.19,rely=0.2)

self.l2.place(relx=0.3,rely=0.3)

self.er=Entry(self.root,width=40,borderwidth=1)

self.er.place(relx=0.45,rely=0.3)

self.button\_abtacc=Button(self.root,text="About",padx=20,pady=8,font=("arial",10),command= lambda: self.jabout())

self.button\_abtacc.place(relx=0.055,rely=0)

self.button\_home=Button(self.root,text="Home",padx=20,pady=8,font=("arial",10),command= lambda: self.home())

self.button\_home.place(relx=0,rely=0)

self.category= StringVar()

self.category.set("OPEN")

self.dc=OptionMenu(self.root,self.category,'OPEN','OBC-NCL','EWS','ST','SC')

self.dc.place(relx=0.45,rely=0.4)

self.l3.place(relx=0.3,rely=0.4)

self.course= StringVar()

self.course.set("Computer Science and Engineering (4 Years, Bachelor of Technology)")

self.di=OptionMenu(self.root,self.course,'Computer Science and Engineering (4 Years, Bachelor of Technology)','Mechanical Engineering (4 Years, Bachelor of Technology)','Civil Engineering (4 Years, Bachelor of Technology)','Electronics and Electrical Communication Engineering (4 Years, Bachelor of Technology)','Electrical and Electronics Engineering (4 Years, Bachelor of Technology)','Artificial Intelligence (4 Years, Bachelor of Technology)','Data Science and Engineering (4 Years, Bachelor of Technology)','Information Technology (4 Years, Bachelor of Technology)','Chemical Engineering (4 Years, Bachelor of Technology)','Metallurgical Engineering (4 Years, Bachelor of Technology)','Mining Engineering (4 Years, Bachelor of Technology)','Aerospace Engineering (4 Years, Bachelor of Technology)','Bio Engineering (4 Years, Bachelor of Technology)',)

self.di.place(relx=0.45,rely=0.5)

self.l4.place(relx=0.3,rely=0.5)

self.r=IntVar()

self.l5.place(relx=0.3,rely=0.6)

Radiobutton(self.root,text="Female",bg="white",font=(10),variable=self.r,value=1).place(relx=0.45,rely=0.6)

Radiobutton(self.root,text="Male",bg="white",font=(10),variable=self.r,value=2).place(relx=0.55,rely=0.6)

self.r1=IntVar()

self.l6.place(relx=0.3,rely=0.7)

Radiobutton(self.root,text="Yes",bg="white",font=(10),variable=self.r1,value=1).place(relx=0.45,rely=0.7)

Radiobutton(self.root,text="No",bg="white",font=(10),variable=self.r1,value=2).place(relx=0.55,rely=0.7)

self.button\_calc=Button(self.root,text="Predict Results",padx=90,pady=25,fg="white",bg="orange",font=("arial",15,"bold"),command=lambda:self.jcalculate())

self.button\_calc.place(relx=0.4,rely=0.8)

def eamcet(self):

self.root.destroy()

self.root=Tk()

self.root.geometry("+-10+0")

self.root.geometry("1700x800")

self.frame1=Frame(self.root,bg="white",width=1700,height=775)

self.frame1.place(relx=0,rely=0.05)

self.root.title("EAMCET Predictions")

self.l1=Label(self.root,text="Giving your exam rank will help us recommend you better colleges. If you don't have actual rank, then enter expected rank.",bg="white",font=("arial",12,"bold"))

self.l2=Label(self.root,text="Enter your rank",bg="white",font=("arial",13,"bold"))

self.l3=Label(self.root,text="Enter your category",bg="white",font=("arial",13,"bold"))

self.l4=Label(self.root,text="Enter your course",bg="white",font=("arial",13,"bold"))

self.l5=Label(self.root,text="Enter your gender",bg="white",font=("arial",13,"bold"))

self.l6=Label(self.root,text="Are you specially abled",bg="white",font=("arial",13,"bold"))

self.l7=Label(self.root,text=" ")

self.l8=Label(self.root,text=" ")

self.l9=Label(self.root,text=" ")

self.l10=Label(self.root,text=" ")

self.l11=Label(self.root,text=" ")

self.l12=Label(self.root,text=" ")

self.button\_e=Button(self.root,text="EAMCET",padx=160,pady=20,fg="white",bg="#008080",font=("arial",15,"bold"),command= lambda:self.eamcet())

self.button\_j=Button(self.root,text="JEE MAINS",padx=160,pady=20,bg="white",font=("arial",15,"bold"),command= lambda:self.jee())

self.button\_e.place(relx=0.2,rely=0.05)

self.button\_j.place(relx=0.5,rely=0.05)

self.er=Entry(self.root,width=40,borderwidth=1)

self.button\_abtacc=Button(self.root,text="About",padx=20,pady=8,font=("arial",10),command= lambda: self.eabout())

self.button\_abtacc.place(relx=0.055,rely=0)

self.button\_home=Button(self.root,text="Home",padx=20,pady=8,font=("arial",10),command= lambda: self.home())

self.button\_home.place(relx=0,rely=0)

self.l1.place(relx=0.19,rely=0.2)

self.l2.place(relx=0.3,rely=0.3)

self.er.place(relx=0.45,rely=0.3)

self.l3.place(relx=0.3,rely=0.4)

self.category = StringVar()

self.category.set("OC")

self.dc=OptionMenu(self.root,self.category,'OC','BC-A','BC-B','BC-C','BC-D','BC-E','SC','ST')

self.dc.place(relx=0.45,rely=0.4)

self.l4.place(relx=0.3,rely=0.5)

self.course= StringVar()

self.course.set("CSE")

self.di=OptionMenu(self.root,self.course,'CSE', 'MEC', 'ECE', 'EEE', 'MET','CIV', 'CHE','TEX', 'MIN','AGR', 'BIO')

self.di.place(relx=0.45,rely=0.5)

self.r=IntVar()

self.l5.place(relx=0.3,rely=0.6)

Radiobutton(self.root,text="Female",bg="white",font=(10),variable=self.r,value=1).place(relx=0.45,rely=0.6)

Radiobutton(self.root,text="Male",bg="white",font=(10),variable=self.r,value=2).place(relx=0.55,rely=0.6)

self.button\_calc=Button(self.root,text="Predict Results",padx=90,pady=20,fg="white",bg="orange",font=("arial",15,"bold"),command=lambda:self.ecalculate())

self.button\_calc.place(relx=0.4,rely=0.75)

def home(self):

self.root.destroy()

self.root=Tk()

self.root.title("College Predictor")

self.root.geometry("+-10+0")

self.root.geometry("1700x800")

self.button\_e=Button(self.root,text="EAMCET",padx=160,pady=20,bg="white",font=("arial",15,"bold"),command= lambda:self.eamcet())

self.button\_j=Button(self.root,text="JEE MAINS",padx=160,pady=20,bg="white",font=("arial",15,"bold"),command= lambda:self.jee())

self.button\_e.place(relx=0.2,rely=0.4)

self.button\_j.place(relx=0.5,rely=0.4)

self.ld=Label(self.root,text="Select the examination",font=("arial",25,"bold"))

self.ld.place(relx=0.37,rely=0.3)

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

self.root=Tk()

self.root.title("College Predictor")

self.root.geometry("+-10+0")

self.root.geometry("1700x800")

self.button\_e=Button(self.root,text="EAMCET",padx=160,pady=20,bg="white",font=("arial",15,"bold"),command= lambda:self.eamcet())

self.button\_j=Button(self.root,text="JEE MAINS",padx=160,pady=20,bg="white",font=("arial",15,"bold"),command= lambda:self.jee())

self.button\_e.place(relx=0.2,rely=0.4)

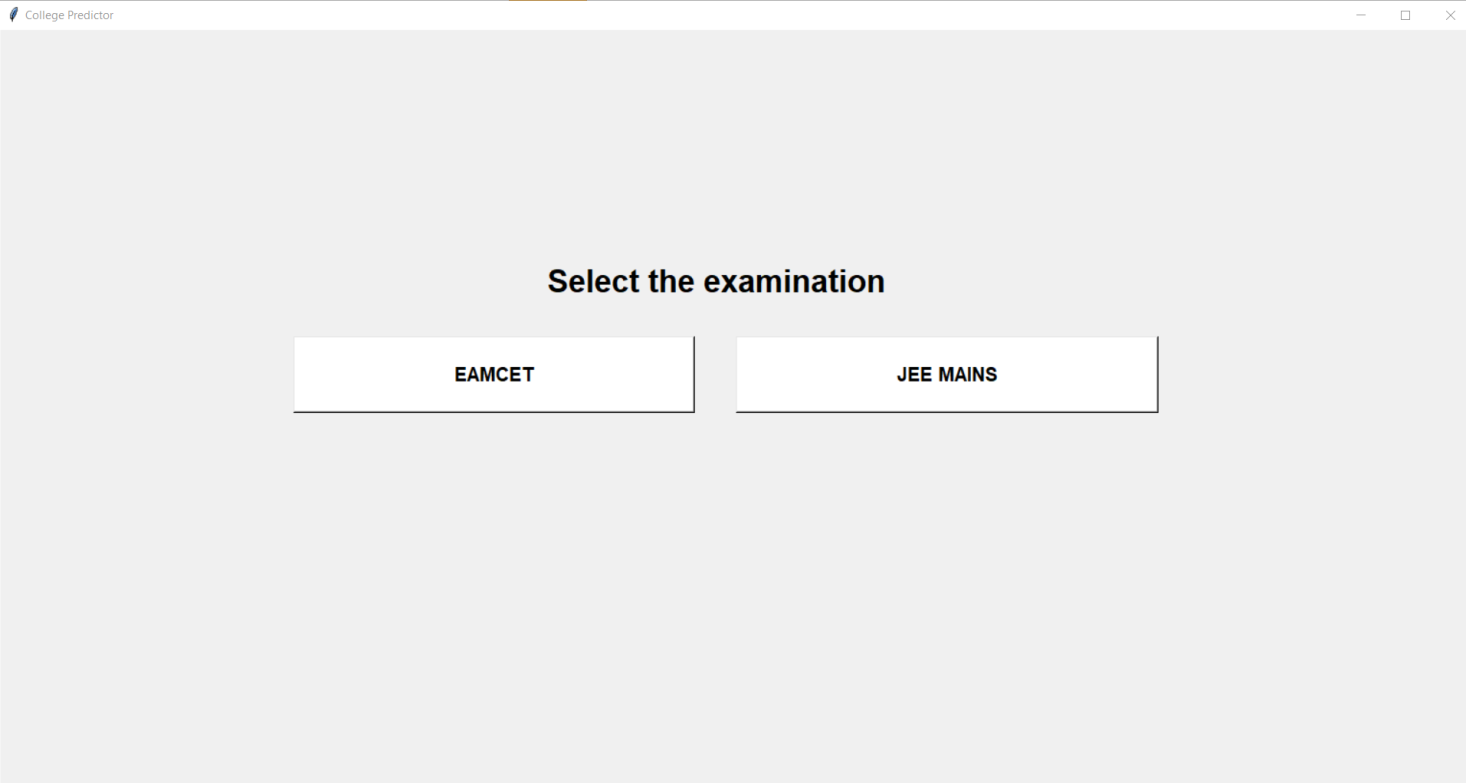
self.button\_j.place(relx=0.5,rely=0.4)

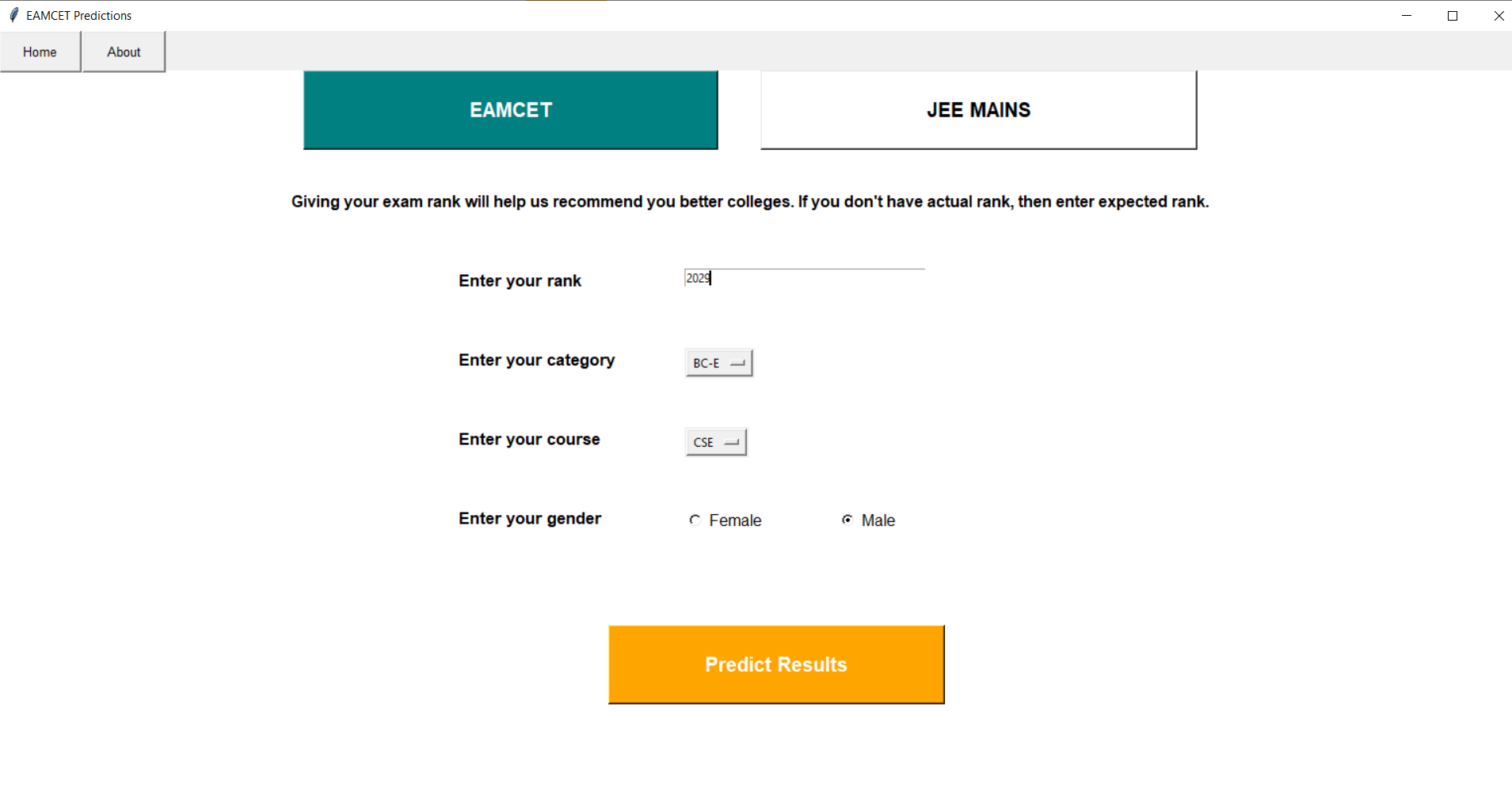
self.ld=Label(self.root,text="Select the examination",font=("arial",25,"bold"))

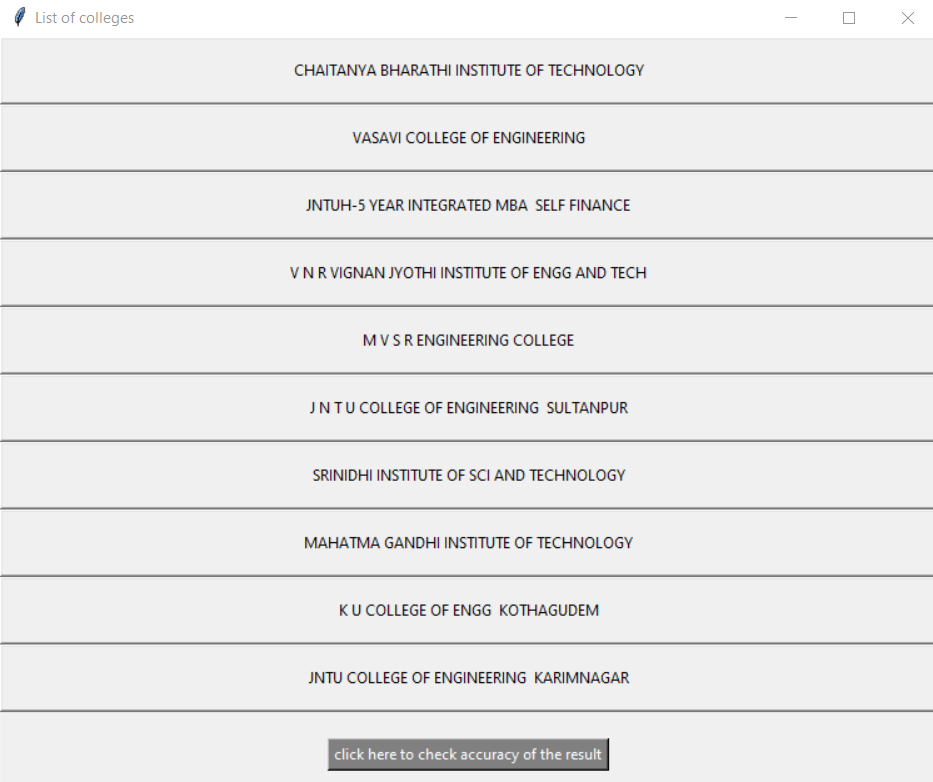
self.ld.place(relx=0.37,rely=0.3)

obj=predictor()

obj.root.mainloop()







**Future scope:**

In future more exam results can be predicted by this predictor like the results of AP EAMCET, NEET, AIMS, POLYTECHNIC, GATE and many more along with the existing exams that are , JEE MAINS, JEE ADVANCED and TSEAMCET.

Furthermore improvement in UI is possible using place option instead of pack and grid and also by using better colour combinations.

The algorithm used for calculating accuracy of the predictor can be made more subtle so that we can get a better result accuracy which can furthermore improve the accuracy of the accuracy predictions.

Features like the details of the college in the result can be displayed using selenium module in python, which can help the user to get the college info easily without any extra effort so that comparing colleges becomes easier for the user.

Instead of using python by using html, css and javascript this can be made into a site that will enable the user to use this predictor from any place any time without having to install python and its modules and the files related to the data of the colleges and there cut offs.

**References:**

1. **Codemy – tkinter course**

[1]**<https://www.youtube.com/watch?v=YXPyB4XeYLA>**

1. **Geeksforgeeks for widgets of tkinter**

[2]**[https://www.geeksforgeeks.org/python-gui-tkinter/#:~:text=Out%20of%20all%20the%20GUI,tkinter%20is%20an%20easy%20task](https://www.geeksforgeeks.org/python-gui-tkinter/" \l ":~:text=Out%20of%20all%20the%20GUI,tkinter%20is%20an%20easy%20task).**

1. **tutorialpoint for placing of the widgets**

[3]**<https://www.tutorialspoint.com/python/python_gui_programming.htm>**

1. **tseamcet cutoff data of previous 4 years data**

[4]**<https://engineering.careers360.com/articles/ts-eamcet-cutoff>**

1. **JEE cutoff data for 4 rounds**

[5][**https://engineering.careers360.com/articles/jee-main-cutoff-marks-and-ranks-faqs#:~:text=General%20Category%20%E2%80%93%2081%20to%20113,ST%20%E2%80%93%2027%20to%2048**](https://engineering.careers360.com/articles/jee-main-cutoff-marks-and-ranks-faqs#:~:text=General%20Category%20%E2%80%93%2081%20to%20113,ST%20%E2%80%93%2027%20to%2048)