```
import pandas as pd

data = pd.read_csv('Placement.csv')

import warnings

warnings.filterwarnings('ignore')
```

1. Display Top 5 Rows of The Dataset

```
data.head()

   sl_no  gender  ssc_p    ssc_b  hsc_p    hsc_b     hsc_s
degree_p  \
0      1       0  67.00   Others  91.00   Others  Commerce     58.00

1      2       0  79.33  Central  78.33   Others   Science     77.48

2      3       0  65.00  Central  68.00  Central      Arts     64.00

3      4       0  56.00  Central  52.00  Central   Science     52.00

4      5       0  85.80  Central  73.60  Central  Commerce     73.30


    degree_t workex  etest_p specialisation  mba_p      status
salary
0   Sci&Tech     No     55.0         Mkt&HR  58.80      Placed
270000.0
1   Sci&Tech    Yes     86.5        Mkt&Fin  66.28      Placed
200000.0
2  Comm&Mgmt     No     75.0        Mkt&Fin  57.80      Placed
250000.0
3   Sci&Tech     No     66.0         Mkt&HR  59.43  Not Placed
NaN
4  Comm&Mgmt     No     96.8        Mkt&Fin  55.50      Placed
425000.0
```

2. Check Last 5 Rows of The Dataset

```
data.tail()

     sl_no  gender  ssc_p    ssc_b  hsc_p    hsc_b     hsc_s  degree_p
\
210    211       0   80.6   Others   82.0   Others  Commerce      77.6

211    212       0   58.0   Others   60.0   Others   Science      72.0

212    213       0   67.0   Others   67.0   Others  Commerce      73.0

213    214       1   74.0   Others   66.0   Others  Commerce      58.0
```

```
214    215        0   62.0   Central    58.0   Others    Science      53.0


      degree_t workex  etest_p specialisation  mba_p      status
salary
210   Comm&Mgmt      No     91.0        Mkt&Fin  74.49      Placed
400000.0
211    Sci&Tech      No     74.0        Mkt&Fin  53.62      Placed
275000.0
212   Comm&Mgmt     Yes     59.0        Mkt&Fin  69.72      Placed
295000.0
213   Comm&Mgmt      No     70.0         Mkt&HR  60.23      Placed
204000.0
214   Comm&Mgmt      No     89.0         Mkt&HR  60.22  Not Placed
NaN
```

3. Find Shape of Our Dataset (Number of Rows And Number of Columns)

```
data.shape

(215, 15)

print("Number of Rows",data.shape[0])
print("Number of Columns",data.shape[1])

Number of Rows 215
Number of Columns 15
```

4. Get Information About Our Dataset Like the Total Number of Rows, Total Number of Columns, Datatypes of Each Column And Memory Requirement

```
data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 215 entries, 0 to 214
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   sl_no           215 non-null    int64
 1   gender          215 non-null    int64
 2   ssc_p           215 non-null    float64
 3   ssc_b           215 non-null    object
 4   hsc_p           215 non-null    float64
 5   hsc_b           215 non-null    object
 6   hsc_s           215 non-null    object
 7   degree_p        215 non-null    float64
 8   degree_t        215 non-null    object 
 9   workex          215 non-null    object 
 10  etest_p         215 non-null    float64
 11  specialisation  215 non-null    object 
 12  mba_p           215 non-null    float64
 13  status          215 non-null    object 
 14  salary          148 non-null    float64
dtypes: float64(6), int64(2), object(7)
memory usage: 25.3+ KB
```

5. Check Null Values In The Dataset

```
data.isnull().sum()

sl_no              0
gender             0
ssc_p              0
ssc_b              0
hsc_p              0
hsc_b              0
hsc_s              0
degree_p           0
degree_t           0
workex             0
etest_p            0
specialisation     0
mba_p              0
status             0
salary            67
dtype: int64
```

6. Get Overall Statistics About The Dataset

```
data.describe()

            sl_no      gender       ssc_p       hsc_p    degree_p  
etest_p  \
count  215.000000  215.000000  215.000000  215.000000  215.000000  
215.000000   
mean   108.000000    0.353488   67.303395   66.333163   66.370186   
72.100558   
std     62.209324    0.479168   10.827205   10.897509    7.358743   
13.275956   
min      1.000000    0.000000   40.890000   37.000000   50.000000   
50.000000   
25%     54.500000    0.000000   60.600000   60.900000   61.000000   
60.000000   
50%    108.000000    0.000000   67.000000   65.000000   66.000000   
71.000000   
75%    161.500000    1.000000   75.700000   73.000000   72.000000   
```

```
83.500000
max     215.000000    1.000000   89.400000   97.700000   91.000000
98.000000

            mba_p         salary
count  215.000000     148.000000
mean    62.278186  288655.405405
std      5.833385   93457.452420
min     51.210000  200000.000000
25%     57.945000  240000.000000
50%     62.000000  265000.000000
75%     66.255000  300000.000000
max     77.890000  940000.000000
```

7. EDA

```
data.columns

Index(['sl_no', 'gender', 'ssc_p', 'ssc_b', 'hsc_p', 'hsc_b', 'hsc_s',
       'degree_p', 'degree_t', 'workex', 'etest_p', 'specialisation',
'mba_p',
       'status', 'salary'],
      dtype='object')
```

How Many Students Got Placed?

```
data['status'].unique()

array(['Placed', 'Not Placed'], dtype=object)

data['status'].value_counts()

Placed        148
Not Placed     67
Name: status, dtype: int64
```

Could you display the top 5 sci&tech students placed according to their salary?

```
data.columns

Index(['sl_no', 'gender', 'ssc_p', 'ssc_b', 'hsc_p', 'hsc_b', 'hsc_s',
       'degree_p', 'degree_t', 'workex', 'etest_p', 'specialisation',
'mba_p',
       'status', 'salary'],
      dtype='object')

data[(data['degree_t']=="Sci&Tech") &
(data['status']=="Placed")].sort_values(by="salary",ascending=False).h
ead()
```

```
     sl_no  gender  ssc_p    ssc_b  hsc_p    hsc_b    hsc_s  degree_p  
\
150    151       0  71.00  Central  58.66  Central  Science     58.00

77      78       0  64.00   Others  80.00   Others  Science     65.00

163    164       0  63.00   Others  67.00   Others  Science     64.00

174    175       0  73.24   Others  50.83   Others  Science     64.27

53      54       0  80.00   Others  70.00   Others  Science     72.00


     degree_t workex  etest_p specialisation  mba_p  status    salary

150  Sci&Tech    Yes     56.0        Mkt&Fin  61.30  Placed  690000.0

77   Sci&Tech    Yes     69.0        Mkt&Fin  57.65  Placed  500000.0

163  Sci&Tech     No     75.0        Mkt&Fin  66.46  Placed  500000.0

174  Sci&Tech    Yes     64.0        Mkt&Fin  66.23  Placed  500000.0

53   Sci&Tech     No     87.0         Mkt&HR  71.04  Placed  450000.0
```

8. Data Preprocessing

```
data.head()
```

```
   sl_no  gender  ssc_p    ssc_b  hsc_p    hsc_b     hsc_s  
degree_p  \
0      1       0  67.00   Others  91.00   Others  Commerce     58.00

1      2       0  79.33  Central  78.33   Others   Science     77.48

2      3       0  65.00  Central  68.00  Central      Arts     64.00

3      4       0  56.00  Central  52.00  Central   Science     52.00

4      5       0  85.80  Central  73.60  Central  Commerce     73.30


    degree_t workex  etest_p specialisation  mba_p      status  
salary
0   Sci&Tech     No     55.0         Mkt&HR  58.80      Placed  
270000.0
1   Sci&Tech    Yes     86.5        Mkt&Fin  66.28      Placed  
200000.0
2  Comm&Mgmt     No     75.0        Mkt&Fin  57.80      Placed  
250000.0
3   Sci&Tech     No     66.0         Mkt&HR  59.43  Not Placed  
```

```
NaN
4   Comm&Mgmt     No      96.8        Mkt&Fin  55.50      Placed
425000.0

data = data.drop(['sl_no','salary'],axis=1)

data.head(1)

   gender  ssc_p   ssc_b  hsc_p   hsc_b     hsc_s  degree_p  degree_t
workex  \
0       0   67.0  Others   91.0  Others  Commerce      58.0  Sci&Tech
No

   etest_p specialisation  mba_p  status
0     55.0         Mkt&HR   58.8  Placed
```

Encoding the Categorical Columns

```
data['ssc_b'].unique()

array(['Others', 'Central'], dtype=object)

data['ssc_b'] = data['ssc_b'].map({'Central':1,'Others':0})

data.head(2)

   gender  ssc_p  ssc_b  hsc_p   hsc_b     hsc_s  degree_p  degree_t
workex  \
0       0  67.00      0  91.00  Others  Commerce     58.00  Sci&Tech
No
1       0  79.33      1  78.33  Others   Science     77.48  Sci&Tech
Yes

   etest_p specialisation  mba_p  status
0     55.0         Mkt&HR  58.80  Placed
1     86.5        Mkt&Fin  66.28  Placed

data['hsc_b'].unique()

array(['Others', 'Central'], dtype=object)

data['hsc_b'] = data['hsc_b'].map({'Central':1,'Others':0})

data.head(2)

   gender  ssc_p  ssc_b  hsc_p  hsc_b     hsc_s  degree_p  degree_t
workex  \
0       0  67.00      0  91.00      0  Commerce     58.00  Sci&Tech
No
1       0  79.33      1  78.33      0   Science     77.48  Sci&Tech
Yes
```

```
   etest_p specialisation  mba_p  status
0     55.0          Mkt&HR  58.80  Placed
1     86.5         Mkt&Fin  66.28  Placed
```

```
data['hsc_s'].unique()
```

```
array(['Commerce', 'Science', 'Arts'], dtype=object)
```

```
data['hsc_s'] = data['hsc_s'].map({'Science':2,'Commerce':1,'Arts':0})
```

```
data.head()
```

```
   gender  ssc_p  ssc_b  hsc_p  hsc_b  hsc_s  degree_p   degree_t
workex  \
0       0  67.00      0  91.00      0      1     58.00   Sci&Tech
No
1       0  79.33      1  78.33      0      2     77.48   Sci&Tech
Yes
2       0  65.00      1  68.00      1      0     64.00  Comm&Mgmt
No
3       0  56.00      1  52.00      1      2     52.00   Sci&Tech
No
4       0  85.80      1  73.60      1      1     73.30  Comm&Mgmt
No

   etest_p specialisation  mba_p      status
0     55.0          Mkt&HR  58.80      Placed
1     86.5         Mkt&Fin  66.28      Placed
2     75.0         Mkt&Fin  57.80      Placed
3     66.0          Mkt&HR  59.43  Not Placed
4     96.8         Mkt&Fin  55.50      Placed
```

```
data['degree_t'].unique()
```

```
array(['Sci&Tech', 'Comm&Mgmt', 'Others'], dtype=object)
```

```
data['degree_t'] =
data['degree_t'].map({'Sci&Tech':2,'Comm&Mgmt':1,'Others':0})
```

```
data.head(2)
```

```
   gender  ssc_p  ssc_b  hsc_p  hsc_b  hsc_s  degree_p  degree_t
workex  \
0       0  67.00      0  91.00      0      1     58.00         2
No
1       0  79.33      1  78.33      0      2     77.48         2
Yes

   etest_p specialisation  mba_p  status
0     55.0          Mkt&HR  58.80  Placed
1     86.5         Mkt&Fin  66.28  Placed
```

```
data['specialisation'].unique()

array(['Mkt&HR', 'Mkt&Fin'], dtype=object)

data['specialisation']
=data['specialisation'].map({'Mkt&HR':1,'Mkt&Fin':0})

data.head(2)

   gender  ssc_p  ssc_b  hsc_p  hsc_b  hsc_s  degree_p  degree_t
workex  \
0       0  67.00      0  91.00      0      1     58.00         2
No
1       0  79.33      1  78.33      0      2     77.48         2
Yes

   etest_p  specialisation  mba_p  status
0     55.0               1  58.80  Placed
1     86.5               0  66.28  Placed

data['workex'].unique()

array(['No', 'Yes'], dtype=object)

data['workex'] = data['workex'].map({'Yes':1,'No':0})

data.head(2)

   gender  ssc_p  ssc_b  hsc_p  hsc_b  hsc_s  degree_p  degree_t
workex  \
0       0  67.00      0  91.00      0      1     58.00         2
0
1       0  79.33      1  78.33      0      2     77.48         2
1

   etest_p  specialisation  mba_p  status
0     55.0               1  58.80  Placed
1     86.5               0  66.28  Placed

data['status'].unique()

array(['Placed', 'Not Placed'], dtype=object)

data['status'] = data['status'].map({'Placed':1,'Not Placed':0})

data.head()

   gender  ssc_p  ssc_b  hsc_p  hsc_b  hsc_s  degree_p  degree_t
workex  \
0       0  67.00      0  91.00      0      1     58.00         2
0
1       0  79.33      1  78.33      0      2     77.48         2
1
```

```
2       0  65.00       1  68.00       1       0      64.00          1
0
3       0  56.00       1  52.00       1       2      52.00          2
0
4       0  85.80       1  73.60       1       1      73.30          1
0

   etest_p  specialisation  mba_p  status
0     55.0               1  58.80       1
1     86.5               0  66.28       1
2     75.0               0  57.80       1
3     66.0               1  59.43       0
4     96.8               0  55.50       1
```

9. Store Feature Matrix In X and Response(Target) In Vector y

```
data.columns
```

```
Index(['gender', 'ssc_p', 'ssc_b', 'hsc_p', 'hsc_b', 'hsc_s',
'degree_p',
       'degree_t', 'workex', 'etest_p', 'specialisation', 'mba_p',
'status'],
      dtype='object')
```

```
X = data.drop('status',axis=1)
y= data['status']
```

```
y
```

```
0      1
1      1
2      1
3      0
4      1
      ..
210    1
211    1
212    1
213    1
214    0
Name: status, Length: 215, dtype: int64
```

10. Splitting The Dataset Into The Training Set And Test Set

```
from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.20,rand
om_state=42)
```

11. Import The models

```
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn import svm
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
```

12. Model Training

```
lr = LogisticRegression()
lr.fit(X_train,y_train)

svm = svm.SVC()
svm.fit(X_train,y_train)

knn=KNeighborsClassifier()
knn.fit(X_train,y_train)

dt=DecisionTreeClassifier()
dt.fit(X_train,y_train)

rf=RandomForestClassifier()
rf.fit(X_train,y_train)

gb=GradientBoostingClassifier()
gb.fit(X_train,y_train)
```

```
GradientBoostingClassifier()
```

13. Prediction on Test Data

```
y_pred1 = lr.predict(X_test)
y_pred2 = svm.predict(X_test)
y_pred3 = knn.predict(X_test)
y_pred4 = dt.predict(X_test)
y_pred5 = rf.predict(X_test)
y_pred6 = gb.predict(X_test)
```

14. Evaluating the Algorithms

```
from sklearn.metrics import accuracy_score

score1=accuracy_score(y_test,y_pred1)
score2=accuracy_score(y_test,y_pred2)
score3=accuracy_score(y_test,y_pred3)
score4=accuracy_score(y_test,y_pred4)
score5=accuracy_score(y_test,y_pred5)
score6=accuracy_score(y_test,y_pred6)
```

```
print(score1,score2,score3,score4,score5,score6)
```

```
0.8837209302325582 0.7674418604651163 0.7906976744186046
0.8372093023255814 0.7906976744186046 0.813953488372093
```

```
final_data = pd.DataFrame({'Models':['LR','SVC','KNN','DT','RF','GB'],
            'ACC':[score1*100,
                  score2*100,
                  score3*100,
                  score4*100,
                  score5*100,score6*100]})
```

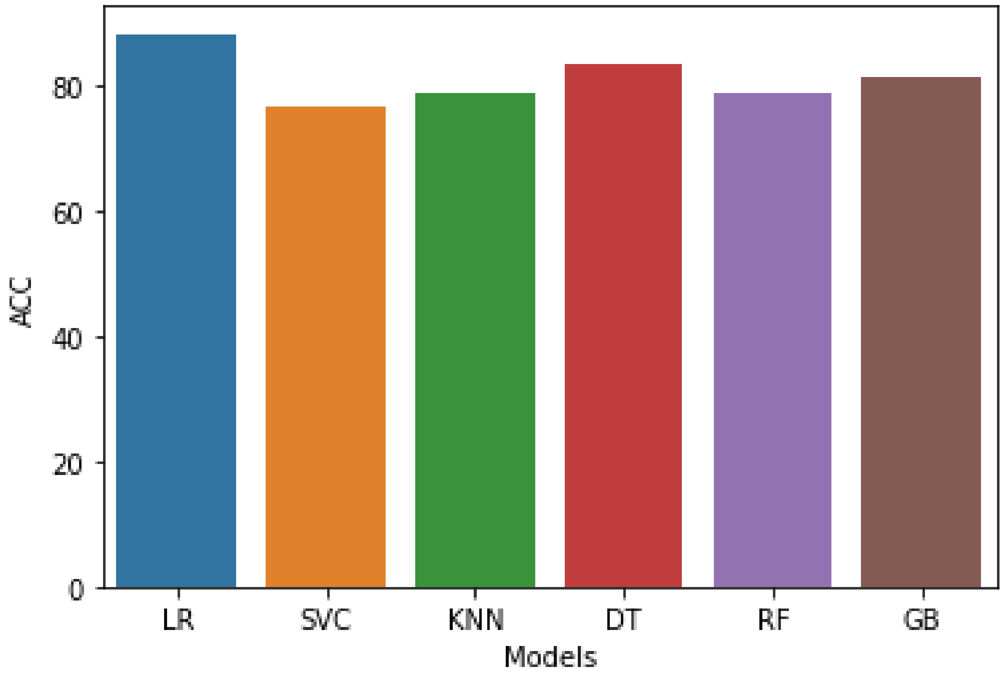
```
final_data
```

```
  Models        ACC
0     LR  88.372093
1    SVC  76.744186
2    KNN  79.069767
3     DT  83.720930
4     RF  79.069767
5     GB  81.395349
```

```
import seaborn as sns
```

```
sns.barplot(final_data['Models'],final_data['ACC'])
```

```
<AxesSubplot:xlabel='Models', ylabel='ACC'>
```



15. Prediction on New Data

```
new_data = pd.DataFrame({
    'gender':0,
    'ssc_p':67.0,
    'ssc_b':0,
    'hsc_p':91.0,
    'hsc_b':0,
    'hsc_s':1,
    'degree_p':58.0,
    'degree_t':2,
    'workex':0,
    'etest_p':55.0,
     'specialisation':1,
    'mba_p':58.8,
},index=[0])
```

```
lr= LogisticRegression()
lr.fit(X,y)
```

```
LogisticRegression()
```

```
p=lr.predict(new_data)
prob=lr.predict_proba(new_data)
if p==1:
    print('Placed')
    print(f"You will be placed with probability of {prob[0][1]:.2f}")
else:
    print("Not-placed")
```

```
Placed
You will be placed with probability of 0.96
```

```
prob
```

```
array([[0.04186191, 0.95813809]])
```

16. Save Model Using Joblib

```
import joblib
```

```
joblib.dump(lr,'model_campus_placement')
```

```
['model_campus_placement']
```

```
model = joblib.load('model_campus_placement')
```

```
model.predict(new_data)
```

```
array([1], dtype=int64)
```

GUI

```
from tkinter import *
import joblib
import numpy as np
from sklearn import *
import tkinter.font as font
import pandas as pd

def show_entry_fields():
    text = clicked.get()
    if text == "Male":
        p1=1
        print(p1)
    else:
        p1=0
        print(p1)
    p2=float(e2.get())
    text = clicked1.get()
    if text == "Central":
        p3=1
        print(p3)
    else:
        p3=0
        print(p3)
    p4=float(e4.get())
    text = clicked6.get()
    if text == "Central":
        p5=1
        print(p3)
    else:
        p5=0
        print(p3)
    text = clicked2.get()
    if text == "Science":
        p6=2
        print(p6)
    elif text == "Commerce":
        p6=1
        print(p6)
    else:
        p6=0
        print(p6)
    p7=float(e7.get())
    text = clicked3.get()
    if text == "Sci&Tech":
        p8=2
        print(p8)
    elif text=="Comm&Mgmt":
        p8=1
        print(p8)
    else:
        p8=0
        print(p8)
    text = clicked4.get()
    if text == "Yes":
        p9=1
        print(p3)
    else:
        p9=0
        print(p3)
    p10=float(e10.get())
    text = clicked5.get()
    if text == "Mkt&HR":
        p11=1
        print(p11)
    else:
        p11=0
        print(p11)
    p12=float(e12.get())

    model = joblib.load('model_campus_placement')
    new_data = pd.DataFrame({
    'gender':p1,
    'ssc_p':p2,
    'ssc_b':p3,
    'hsc_p':p4,
    'hsc_b':p5,
    'hsc_s':p6,
    'degree_p':p7,
    'degree_t':p8,
    'workex':p9,
    'etest_p':p10,
     'specialisation':p11,
    'mba_p':p12,
},index=[0])
    result=model.predict(new_data)
    result1=model.predict_proba(new_data)

    if result[0] == 0:
        Label(master, text="Can't Placed").grid(row=31)
    else:
        Label(master, text="Student Will be Placed With Probability of",font=("Arial", 15)).grid(row=31)
        Label(master, text=round(result1[0][1],2)*100,font=("Arial", 15)).grid(row=33)
        Label(master, text="Percent",font=("Arial", 15)).grid(row=34)

master = Tk()
```

```
master.title("Campus Placement Prediction System")


label = Label(master, text = "Campus Placement Prediction System"
                          , bg = "green", fg = "white",font=("Arial",
20)) \
                               .grid(row=0,columnspan=2)


Label(master, text="Gender",font=("Arial", 15)).grid(row=1)
Label(master, text="Secondary Education percentage- 10th
Grade",font=("Arial", 15)).grid(row=2)
Label(master, text="Board of Education",font=("Arial",
15)).grid(row=3)
Label(master, text="Higher Secondary Education percentage- 12th
Grade",font=("Arial", 15)).grid(row=4)
Label(master, text="Board of Education",font=("Arial",
15)).grid(row=5)
Label(master, text="Specialization in Higher Secondary
Education",font=("Arial", 15)).grid(row=6)
Label(master, text="Degree Percentage",font=("Arial", 15)).grid(row=7)
Label(master, text="Under Graduation(Degree type)- Field of degree
education",font=("Arial", 15)).grid(row=8)
Label(master, text="Work Experience",font=("Arial", 15)).grid(row=9)
Label(master, text="Enter test percentage",font=("Arial",
15)).grid(row=10)
Label(master, text="branch specialization",font=("Arial",
15)).grid(row=11)
Label(master, text="MBA percentage",font=("Arial", 15)).grid(row=12)
clicked = StringVar()
options = ["Male","Female"]

clicked1 = StringVar()
options1 = ["Central","Others"]

clicked2 = StringVar()
options2 = ["Science","Commerce","Arts"]

clicked3 = StringVar()
options3 = ["Sci&Tech","Comm&Mgmt","Others"]

clicked4 = StringVar()
options4 = ["Yes","No"]

clicked5 = StringVar()
options5 = ["Mkt&HR","Mky&Fin"]

clicked6 = StringVar()
options6 = ["Central","Others"]
e1 = OptionMenu(master , clicked , *options )
e1.configure(width=13)
```

```
e2 = Entry(master)
e3 = OptionMenu(master , clicked1 , *options1 )
e3.configure(width=13)
e4 = Entry(master)
e5 = OptionMenu(master , clicked6 , *options6)
e5.configure(width=13)
e6 = OptionMenu(master , clicked2 , *options2)
e6.configure(width=13)
e7 = Entry(master)
e8 = OptionMenu(master , clicked3 , *options3)
e8.configure(width=13)
e9 = OptionMenu(master , clicked4 , *options4)
e9.configure(width=13)
e10 = Entry(master)
e11 = OptionMenu(master , clicked5 , *options5)
e11.configure(width=13)
e12 = Entry(master)


e1.grid(row=1, column=1)
e2.grid(row=2, column=1)
e3.grid(row=3, column=1)
e4.grid(row=4, column=1)
e5.grid(row=5, column=1)
e6.grid(row=6, column=1)
e7.grid(row=7, column=1)
e8.grid(row=8, column=1)
e9.grid(row=9, column=1)
e10.grid(row=10, column=1)
e11.grid(row=11, column=1)
e12.grid(row=12, column=1)
buttonFont = font.Font(family='Helvetica', size=16, weight='bold')
Button(master, text='Predict',height= 1,
width=8,activebackground='#00ff00',font=buttonFont,bg='black',
fg='white',command=show_entry_fields).grid()

mainloop()
```