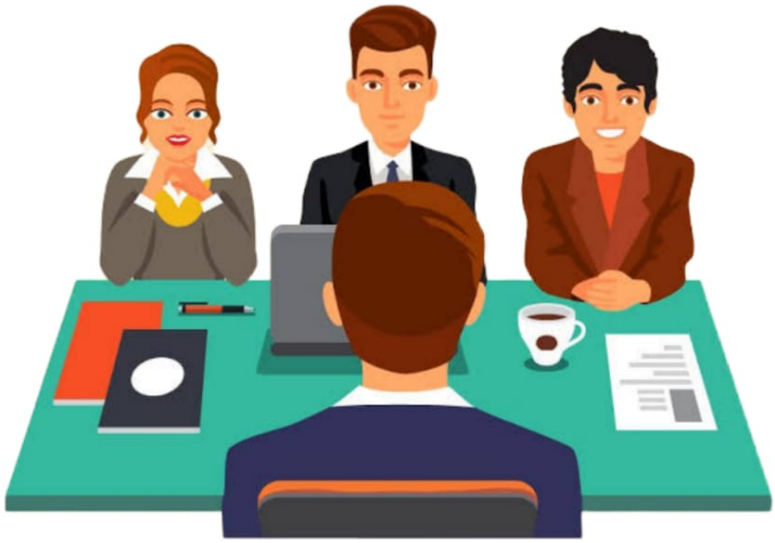

In [68]:
```
import pandas as pd

from sklearn.model_selection import train_test_split

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
from xgboost import XGBClassifier

from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV

from sklearn.metrics import accuracy_score,r2_score,precision_score,recall_sc

import pickle
import joblib
```

In [2]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
dataset = pd.read_csv('Placement.csv')
dataset.shape
```

Out[3]:
```
(215, 15)
```

In [4]:
```
dataset.head()
```

Out[4]:

| | sl_no | gender | ssc_p | ssc_b | hsc_p | hsc_b | hsc_s | degree_p | degree_t | workex | etes |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 67.00 | Others | 91.00 | Others | Commerce | 58.00 | Sci&Tech | No | 5 |
| **1** | 2 | 0 | 79.33 | Central | 78.33 | Others | Science | 77.48 | Sci&Tech | Yes | 8 |
| **2** | 3 | 0 | 65.00 | Central | 68.00 | Central | Arts | 64.00 | Comm&Mgmt | No | 7 |
| **3** | 4 | 0 | 56.00 | Central | 52.00 | Central | Science | 52.00 | Sci&Tech | No | 6 |
| **4** | 5 | 0 | 85.80 | Central | 73.60 | Central | Commerce | 73.30 | Comm&Mgmt | No | 9 |

In [5]:

```
dataset.tail()
```

Out[5]:

| | sl_no | gender | ssc_p | ssc_b | hsc_p | hsc_b | hsc_s | degree_p | degree_t | workex | et |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **210** | 211 | 0 | 80.6 | Others | 82.0 | Others | Commerce | 77.6 | Comm&Mgmt | No | |
| **211** | 212 | 0 | 58.0 | Others | 60.0 | Others | Science | 72.0 | Sci&Tech | No | |
| **212** | 213 | 0 | 67.0 | Others | 67.0 | Others | Commerce | 73.0 | Comm&Mgmt | Yes | |
| **213** | 214 | 1 | 74.0 | Others | 66.0 | Others | Commerce | 58.0 | Comm&Mgmt | No | |
| **214** | 215 | 0 | 62.0 | Central | 58.0 | Others | Science | 53.0 | Comm&Mgmt | No | |

In [6]:

```
print('Number of Rows:',dataset.shape[0])
print('Number of Columns:',dataset.shape[1])
```

```
Number of Rows: 215
Number of Columns: 15
```

In [7]:

```
dataset.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
```

In [8]:

```
dataset.isnull().sum()
```

Out[8]:
```
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
```

In [9]:
```
dataset.isna().sum()
```

Out[9]:
```
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
```

In [10]:
```
dataset.duplicated().sum()
```

Out[10]:
```
0
```

In [11]:
```
dataset.describe()
```

Out[11]:

| | sl_no | gender | ssc_p | hsc_p | degree_p | etest_p | mba_p | |
|---|---|---|---|---|---|---|---|---|
| **count** | 215.000000 | 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 | 62.278186 | 288 |
| **std** | 62.209324 | 0.479168 | 10.827205 | 10.897509 | 7.358743 | 13.275956 | 5.833385 | 93 |
| **min** | 1.000000 | 0.000000 | 40.890000 | 37.000000 | 50.000000 | 50.000000 | 51.210000 | 200 |
| **25%** | 54.500000 | 0.000000 | 60.600000 | 60.900000 | 61.000000 | 60.000000 | 57.945000 | 240 |
| **50%** | 108.000000 | 0.000000 | 67.000000 | 65.000000 | 66.000000 | 71.000000 | 62.000000 | 265 |
| **75%** | 161.500000 | 1.000000 | 75.700000 | 73.000000 | 72.000000 | 83.500000 | 66.255000 | 300 |
| **max** | 215.000000 | 1.000000 | 89.400000 | 97.700000 | 91.000000 | 98.000000 | 77.890000 | 940 |

In [12]:
```
dataset.corr()
```

Out[12]:

| | sl_no | gender | ssc_p | hsc_p | degree_p | etest_p | mba_p | salary |
|---|---|---|---|---|---|---|---|---|
| **sl_no** | 1.000000 | -0.074306 | -0.078155 | -0.085711 | -0.088281 | 0.063636 | 0.022327 | 0.063764 |
| **gender** | -0.074306 | 1.000000 | 0.068969 | 0.021334 | 0.173217 | -0.084294 | 0.300531 | -0.158912 |
| **ssc_p** | -0.078155 | 0.068969 | 1.000000 | 0.511472 | 0.538404 | 0.261993 | 0.388478 | 0.035330 |
| **hsc_p** | -0.085711 | 0.021334 | 0.511472 | 1.000000 | 0.434206 | 0.245113 | 0.354823 | 0.076819 |
| **degree_p** | -0.088281 | 0.173217 | 0.538404 | 0.434206 | 1.000000 | 0.224470 | 0.402364 | -0.019272 |
| **etest_p** | 0.063636 | -0.084294 | 0.261993 | 0.245113 | 0.224470 | 1.000000 | 0.218055 | 0.178307 |
| **mba_p** | 0.022327 | 0.300531 | 0.388478 | 0.354823 | 0.402364 | 0.218055 | 1.000000 | 0.175013 |
| **salary** | 0.063764 | -0.158912 | 0.035330 | 0.076819 | -0.019272 | 0.178307 | 0.175013 | 1.000000 |

In [13]:
```
dataset.columns
```

Out[13]:
```
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')
```

In [14]:
```
dataset['status'].unique()
```

Out[14]:
```
array(['Placed', 'Not Placed'], dtype=object)
```

In [15]:
```
dataset['status'].value_counts()
```

Out[15]:
```
Placed        148
Not Placed     67
Name: status, dtype: int64
```

In [16]:
```
dataset[(dataset['degree_t']=='Sci&Tech') & (dataset['status']=='Placed')].sc
```

Out[16]:

| | sl_no | gender | ssc_p | ssc_b | hsc_p | hsc_b | hsc_s | degree_p | degree_t | workex | etest_p |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **150** | 151 | 0 | 71.00 | Central | 58.66 | Central | Science | 58.00 | Sci&Tech | Yes | 56.00 |
| **77** | 78 | 0 | 64.00 | Others | 80.00 | Others | Science | 65.00 | Sci&Tech | Yes | 69.00 |
| **163** | 164 | 0 | 63.00 | Others | 67.00 | Others | Science | 64.00 | Sci&Tech | No | 75.00 |
| **174** | 175 | 0 | 73.24 | Others | 50.83 | Others | Science | 64.27 | Sci&Tech | Yes | 64.00 |
| **53** | 54 | 0 | 80.00 | Others | 70.00 | Others | Science | 72.00 | Sci&Tech | No | 87.00 |
| **39** | 40 | 0 | 81.00 | Others | 68.00 | Others | Science | 64.00 | Sci&Tech | No | 93.00 |
| **145** | 146 | 0 | 89.40 | Others | 65.66 | Others | Science | 71.25 | Sci&Tech | No | 72.00 |
| **128** | 129 | 0 | 80.40 | Central | 73.40 | Central | Science | 77.72 | Sci&Tech | Yes | 81.20 |
| **24** | 25 | 0 | 76.50 | Others | 97.70 | Others | Science | 78.86 | Sci&Tech | No | 97.40 |
| **70** | 71 | 0 | 82.00 | Others | 61.00 | Others | Science | 62.00 | Sci&Tech | No | 89.00 |
| **22** | 23 | 1 | 69.80 | Others | 60.80 | Others | Science | 72.23 | Sci&Tech | No | 55.53 |

In [17]:
```
dataset = dataset.drop(['sl_no','salary'],axis=1)
```

In [18]:
```
dataset['ssc_b'].unique()
```
Out[18]:
```
array(['Others', 'Central'], dtype=object)
```

In [19]:
```
dataset['ssc_b'] = dataset['ssc_b'].map({'Central':1,'Others':0})
```

In [20]:
```
dataset['hsc_b'].unique()
```
Out[20]:
```
array(['Others', 'Central'], dtype=object)
```

In [21]:
```
dataset['hsc_b'] = dataset['hsc_b'].map({'Central':1,'Others':0})
```

In [22]:
```
dataset['hsc_s'].unique()
```
Out[22]:
```
array(['Commerce', 'Science', 'Arts'], dtype=object)
```

In [23]:
```
dataset['hsc_s'] = dataset['hsc_s'].map({'Science':2,'Commerce':1,'Arts':0})
```

In [24]:
```
dataset['degree_t'].unique()
```
Out[24]:
```
array(['Sci&Tech', 'Comm&Mgmt', 'Others'], dtype=object)
```

In [25]:
```
dataset['degree_t'] = dataset['degree_t'].map({'Sci&Tech':2,'Comm&Mgmt':1,'Ot
```

In [26]:
```
dataset['specialisation'].unique()
```
Out[26]:
```
array(['Mkt&HR', 'Mkt&Fin'], dtype=object)
```

In [27]:
```
dataset['specialisation'] = dataset['specialisation'].map({'Mkt&HR':1,'Mkt&F
```

In [28]:
```
dataset['workex'].unique()
```
Out[28]:
```
array(['No', 'Yes'], dtype=object)
```

In [29]:
```
dataset['workex'] = dataset['workex'].map({'Yes':1,'No':0})
```

In [30]:
```
dataset['status'].unique()
```
Out[30]:
```
array(['Placed', 'Not Placed'], dtype=object)
```

In [31]:
```
dataset['status'] = dataset['status'].map({'Placed':1,'Not Placed':0})
```

In [32]:
```
dataset.head(11)
```

Out[32]:

| | gender | ssc_p | ssc_b | hsc_p | hsc_b | hsc_s | degree_p | degree_t | workex | etest_p | specialisat |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 67.00 | 0 | 91.00 | 0 | 1 | 58.00 | 2 | 0 | 55.00 | |
| **1** | 0 | 79.33 | 1 | 78.33 | 0 | 2 | 77.48 | 2 | 1 | 86.50 | |
| **2** | 0 | 65.00 | 1 | 68.00 | 1 | 0 | 64.00 | 1 | 0 | 75.00 | |
| **3** | 0 | 56.00 | 1 | 52.00 | 1 | 2 | 52.00 | 2 | 0 | 66.00 | |
| **4** | 0 | 85.80 | 1 | 73.60 | 1 | 1 | 73.30 | 1 | 0 | 96.80 | |
| **5** | 0 | 55.00 | 0 | 49.80 | 0 | 2 | 67.25 | 2 | 1 | 55.00 | |
| **6** | 1 | 46.00 | 0 | 49.20 | 0 | 1 | 79.00 | 1 | 0 | 74.28 | |
| **7** | 0 | 82.00 | 1 | 64.00 | 1 | 2 | 66.00 | 2 | 1 | 67.00 | |
| **8** | 0 | 73.00 | 1 | 79.00 | 1 | 1 | 72.00 | 1 | 0 | 91.34 | |
| **9** | 0 | 58.00 | 1 | 70.00 | 1 | 1 | 61.00 | 1 | 0 | 54.00 | |
| **10** | 0 | 58.00 | 1 | 61.00 | 1 | 1 | 60.00 | 1 | 1 | 62.00 | |

In [33]:

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

In [34]:

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.20,random_st
```

In [35]:

```
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)

xg = XGBClassifier()
xg.fit(X_train,y_train)
```

Out[35]:

```
▾ XGBClassifier
              eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwis
e',
              importance_type=None, interaction_constraints='',
              learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=
4,
              max_delta_step=0, max_depth=6, max_leaves=0, min_child_weig
ht=1,
              missing=nan, monotone_constraints='()', n_estimators=100,
              n_jobs=0, num_parallel_tree=1, predictor='auto', random_sta
te=0,
              reg_alpha=0, reg_lambda=1, ...)
```

In [36]:

```
y_pred_1 = lr.predict(X_test)
y_pred_2 = svm.predict(X_test)
y_pred_3 = knn.predict(X_test)
y_pred_4 = dt.predict(X_test)
y_pred_5 = rf.predict(X_test)
y_pred_6 = gb.predict(X_test)
y_pred_7 = xg.predict(X_test)
```

In [37]:

```
score_1 = accuracy_score(y_test,y_pred_1)
score_2 = accuracy_score(y_test,y_pred_2)
score_3 = accuracy_score(y_test,y_pred_3)
score_4 = accuracy_score(y_test,y_pred_4)
score_5 = accuracy_score(y_test,y_pred_5)
score_6 = accuracy_score(y_test,y_pred_6)
score_7 = accuracy_score(y_test,y_pred_7)
```

In [38]:

```
print(score_1,score_2,score_3,score_4,score_5,score_6,score_7)
```

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

In [39]:

```
r_score_1 = r2_score(y_test,y_pred_1)
r_score_2 = r2_score(y_test,y_pred_2)
r_score_3 = r2_score(y_test,y_pred_3)
r_score_4 = r2_score(y_test,y_pred_4)
r_score_5 = r2_score(y_test,y_pred_5)
r_score_6 = r2_score(y_test,y_pred_6)
r_score_7 = r2_score(y_test,y_pred_7)
```

In [40]:

```
print(r_score_1,r_score_2,r_score_3,r_score_4,r_score_5,r_score_6,r_score_7)
```

```
0.422043010752688 -0.15591397849462396 -0.04032258064516148 0.19086021505376
327 -0.04032258064516148 0.19086021505376327 0.19086021505376327
```

In [41]:

```
p_score_1 = precision_score(y_test,y_pred_1)
p_score_2 = precision_score(y_test,y_pred_2)
p_score_3 = precision_score(y_test,y_pred_3)
p_score_4 = precision_score(y_test,y_pred_4)
p_score_5 = precision_score(y_test,y_pred_5)
p_score_6 = precision_score(y_test,y_pred_6)
p_score_7 = precision_score(y_test,y_pred_7)
```

In [42]:
```
print(p_score_1,p_score_2,p_score_3,p_score_4,p_score_5,p_score_6,p_score_7)
```

```
0.90625 0.7837837837837838 0.7894736842105263 0.9 0.8055555555555556 0.85294
11764705882 0.8529411764705882
```

In [43]:
```
rc_score_1 = recall_score(y_test,y_pred_1)
rc_score_2 = recall_score(y_test,y_pred_2)
rc_score_3 = recall_score(y_test,y_pred_3)
rc_score_4 = recall_score(y_test,y_pred_4)
rc_score_5 = recall_score(y_test,y_pred_5)
rc_score_6 = recall_score(y_test,y_pred_6)
rc_score_7 = recall_score(y_test,y_pred_7)
```

In [44]:
```
print(rc_score_1,rc_score_2,rc_score_3,rc_score_4,rc_score_5,rc_score_6,rc_sc
```

```
0.9354838709677419 0.9354838709677419 0.967741935483871 0.8709677419354839
0.9354838709677419 0.9354838709677419 0.9354838709677419
```

In [47]:
```
decision_tree = DecisionTreeClassifier(random_state=42)

params = {
    'criterion':('gini','entropy'),
    'splitter':('best','random'),
    'max_depth':(list(range(1,20))),
    'min_samples_split':[2,3,4],
    'min_samples_leaf':list(range(1,20))}

tree_cv = GridSearchCV(decision_tree,params,scoring='accuracy',n_jobs=-1,verb

tree_cv.fit(X_train,y_train)
```

```
Fitting 3 folds for each of 4332 candidates, totalling 12996 fits
```

Out[47]:
```
▸ GridSearchCV
▸ estimator: DecisionTreeClassifier
  ▸ DecisionTreeClassifier
```

In [53]:
```
dt_y_pred = tree_cv.predict(X_test)
```

In [54]:
```
accuracy_score(y_test,dt_y_pred)
```

Out[54]:
```
0.6976744186046512
```

In [55]:
```
precision_score(y_test,dt_y_pred)
```

Out[55]:
```
0.7647058823529411
```

In [56]:
```
r2_score(y_test,dt_y_pred)
```

Out[56]:
```
-0.5026881720430112
```

In [57]:
```
recall_score(y_test,dt_y_pred)
```

Out[57]:
```
0.8387096774193549
```

In [58]:
```
rf_clf_best_params = RandomForestClassifier(n_estimators=200,
                                            min_samples_split=5,
                                            min_samples_leaf=8,
                                            max_features='auto',
                                            max_depth=None,
                                            bootstrap=False)

rf_clf_best_params.fit(X_train,y_train)
```

Out[58]:
```
RandomForestClassifier
RandomForestClassifier(bootstrap=False, max_features='auto', min_samples_
leaf=8,
                       min_samples_split=5, n_estimators=200)
```

In [59]:
```
rf_y_pred = rf_clf_best_params.predict(X_test)
```

In [60]:
```
r2_score(y_test,rf_y_pred)
```

Out[60]:
```
0.07526881720430079
```

In [61]:
```
accuracy_score(y_test,rf_y_pred)
```

Out[61]:
```
0.813953488372093
```

In [62]:
```
precision_score(y_test,rf_y_pred)
```

Out[62]:
```
0.8285714285714286
```

In [63]:
```
param_grid = {'gamma': [0,0.1,0.2,0.4,0.8,1.6,3.2,6.4,12.8,25.6,51.2,102.4, 2
              'learning_rate': [0.01, 0.03, 0.06, 0.1, 0.15, 0.2, 0.25, 0.300
              'max_depth': [5,6,7,8,9,10,11,12,13,14],
              'n_estimators': [50,65,80,100,115,130,150],
              'reg_alpha': [0,0.1,0.2,0.4,0.8,1.6,3.2,6.4,12.8,25.6,51.2,102
              'reg_lambda': [0,0.1,0.2,0.4,0.8,1.6,3.2,6.4,12.8,25.6,51.2,102

xgb = XGBClassifier(random_state=15,verbosity=0,silent=0)
rcv = RandomizedSearchCV(estimator=xgb, scoring='accuracy',param_distributio
                         verbose=1, random_state=15, n_jobs=-1)

rcv.fit(X_train,y_train)
```

```
Fitting 3 folds for each of 100 candidates, totalling 300 fits
```

Out[63]:
```
RandomizedSearchCV
estimator: XGBClassifier
XGBClassifier
```

In [64]:
```
rcv_y_pred = rcv.predict(X_test)
```

In [65]:
```
r2_score(y_test,rcv_y_pred)
```

Out[65]: -0.04032258064516148

In [66]:
```
accuracy_score(y_test,rcv_y_pred)
```

Out[66]: 0.7906976744186046

In [67]:
```
precision_score(y_test,rcv_y_pred)
```

Out[67]: 0.8055555555555556

In [74]:
```
lr = LogisticRegression()
lr.fit(X,y)
```

Out[74]:
```
▾ LogisticRegression
LogisticRegression()
```

In [75]:
```
pickle.dump(lr,open('model_campus_placement.pkl','wb'))
```

In [76]:
```
model = pickle.load(open('model_campus_placement.pkl','rb'))
```

In [77]:
```
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])
```

In [79]:
```
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
```

In [81]:
```
model.predict(new_data)
```

Out[81]: array([1], dtype=int64)

In [82]:
```
joblib.dump(lr,'model_campus_placement.job')
```

Out[82]: ['model_campus_placement.job']

In [84]:
```
model_job = joblib.load('model_campus_placement.job')
```

In [85]:
```
model_job.predict(new_data)
```

Out[85]:
```
array([1], dtype=int64)
```

In [92]:
```
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.job')
    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=
        Label(master, text=round(result1[0][1],2)*100,font=("Arial", 15)).gri
        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 = "pink", 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"
Label(master, text="Board of Education",font=("Arial", 15)).grid(row=3)
Label(master, text="Higher Secondary Education percentage- 12th Grade",font=(
Label(master, text="Board of Education",font=("Arial", 15)).grid(row=5)
Label(master, text="Specialization in Higher Secondary Education",font=("Aria
Label(master, text="Degree Percentage",font=("Arial", 15)).grid(row=7)
Label(master, text="Under Graduation(Degree type)- Field of degree education"
```

```
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',f
```

```
mainloop()
```

```
1
1
1
2
2
1
1
```