```
import numpy as np
import pandas as pd
import os
```

```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import svm
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
from sklearn.model_selection import cross_val_score
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import joblib
from sklearn.metrics import accuracy_score
```

```
df = pd.read_csv("/content/collegePlace.csv")
df.head()
```

	Age	Gender	Stream	Internships	CGPA	Hostel	HistoryOfBacklogs	PlacedOr
0	22	Male	Electronics And Communication	1	8	1	1	
1	21	Female	Computer Science	0	7	1	1	
2	22	Female	Information Technology	1	6	0	0	

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2966 entries, 0 to 2965
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Age                2966 non-null   int64
 1   Gender             2966 non-null   object
 2   Stream             2966 non-null   object
 3   Internships        2966 non-null   int64
 4   CGPA               2966 non-null   int64
 5   Hostel             2966 non-null   int64
 6   HistoryOfBacklogs  2966 non-null   int64
 7   PlacedOrNot        2966 non-null   int64
dtypes: int64(6), object(2)
memory usage: 185.5+ KB
```

```
df.isnull().sum()
```

```
Age                 0
Gender              0
Stream              0
Internships         0
CGPA                0
Hostel              0
HistoryOfBacklogs   0
PlacedOrNot         0
dtype: int64
```

```
def transformationplot(feature):
  plt.figure(figsize=(12,5))
  plt.subplot(1,2,1)
  sns.distplot(feature)
transformationplot(np.log(df['Age']))
```

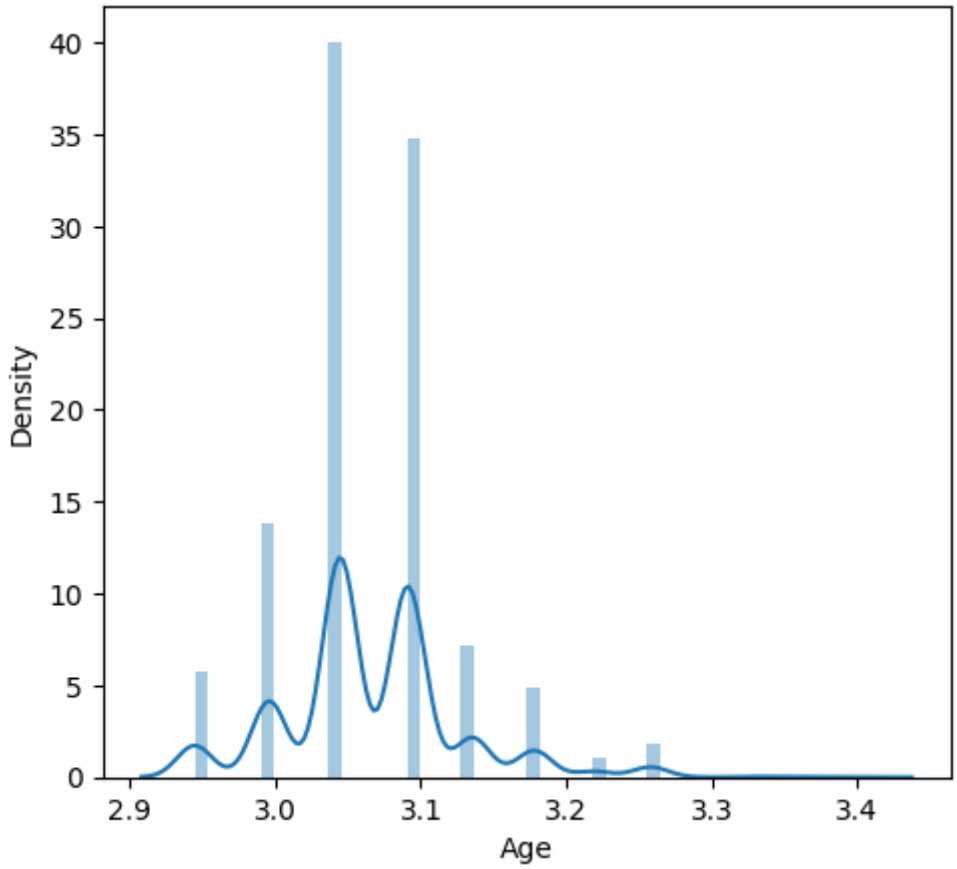
```
<ipython-input-6-1af2f448342d>:4: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(feature)
```



```
df = df.replace(['male'],[0])
df = df.replace(['Female'],[1])
```

```
df = df.replace(['Computer Science','Information Technology','Electronics And Communicatio
```

```
df = df.drop(['Hostel'],axis=1)
```

```
df
```

	Age	Gender	Stream	Internships	CGPA	HistoryOfBacklogs	PlacedOrNot
0	22	Male	2	1	8	1	1
1	21	1	0	0	7	1	1
2	22	1	1	1	6	0	1
3	21	Male	1	0	8	1	1
4	22	Male	3	0	8	0	1
...	...	...	...	...	...	...	...
2961	23	Male	1	0	7	0	0
2962	23	Male	3	1	7	0	0
2963	22	Male	1	1	7	0	0
2964	22	Male	0	1	7	0	0
2965	23	Male	5	0	8	0	1

2966 rows × 7 columns

```
plt.figure(figsize=(12,5))
plt.subplot(121)
sns.distplot(df['CGPA'],color='r')
```

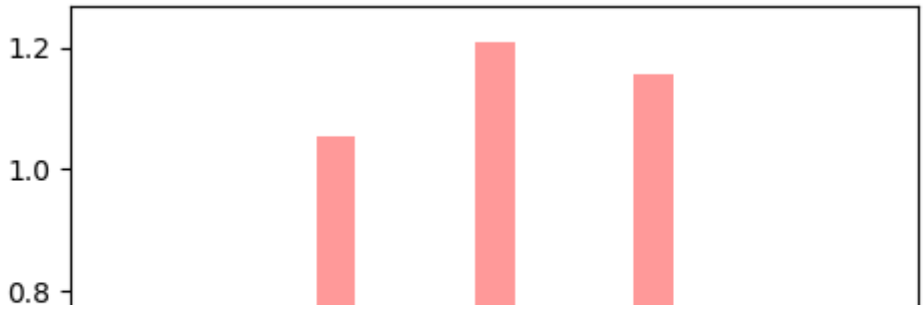
```
<ipython-input-11-f92659182652>:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['CGPA'],color='r')
<Axes: xlabel='CGPA', ylabel='Density'>
```



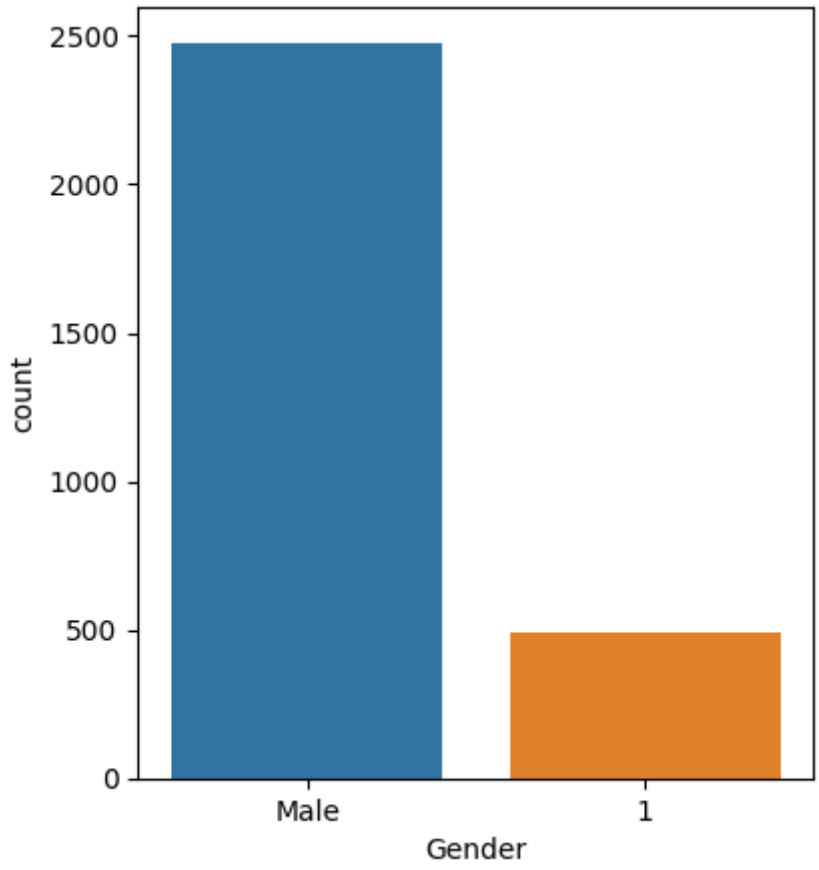

```
plt.figure(figsize=(12,5))
plt.subplot(121)
sns.distplot(df['PlacedOrNot'],color='r')
```

```
<ipython-input-12-5e468beb8a0d>:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
```

```
#plotting the count plot
plt.figure(figsize=(20,5))
plt.subplot(1,4,1)
sns.countplot(x=df["Gender"])
plt.show()
```

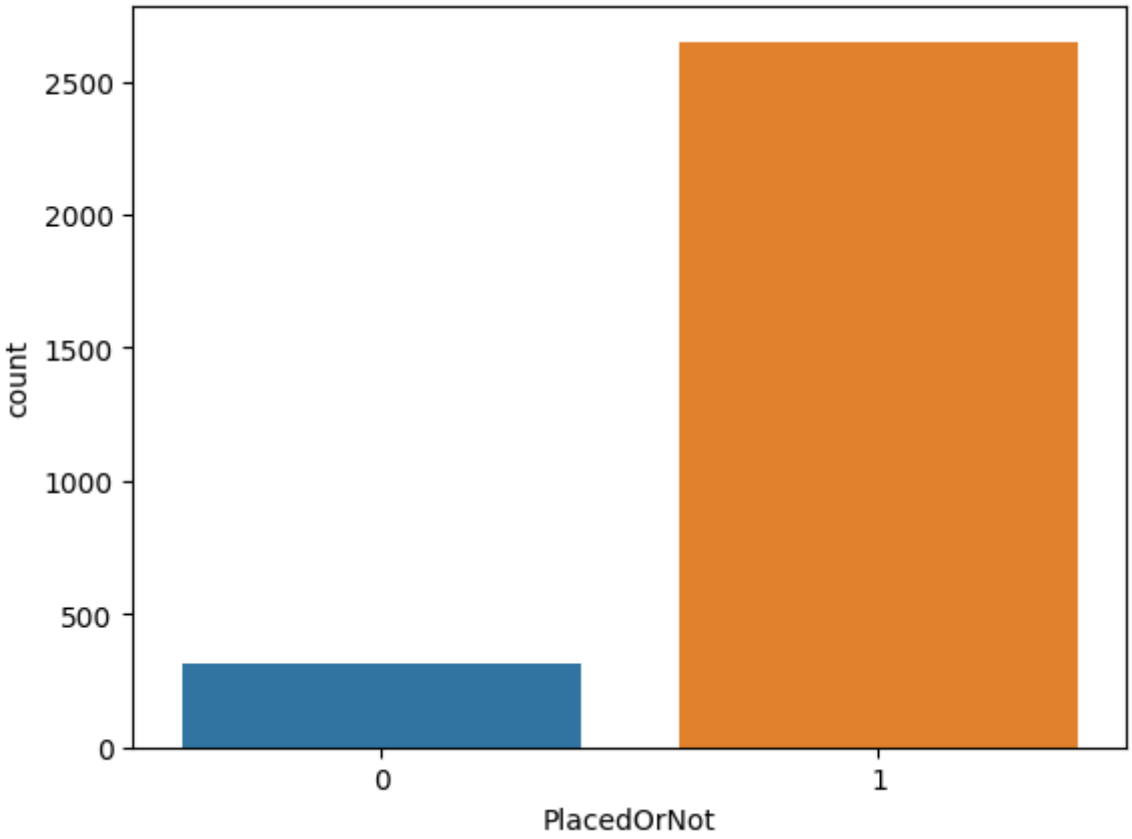



```
plt.figure(figsize=(20,5))
plt.subplot(131)
sns.countplot(df['PlacedOrNot'],x=df['CGPA'])
```

```
<Axes: xlabel='PlacedOrNot', ylabel='count'>
```


```
sns.countplot(df['PlacedOrNot'],x=df['Stream'])
```

```
<Axes: xlabel='PlacedOrNot', ylabel='count'>
```




```
# performing feature scaling operation using standard scalar as x part of the dataset
# there different type of values in the columns
sc=StandardScaler()


x = df['PlacedOrNot']
y = df['CGPA']


x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.4,stratify = x,random_s


classifier = Sequential()
#add input layer and first hidden layer
classifier.add(keras.layers.Dense (6, activation = 'relu', input_dim = 6))
classifier.add(keras.layers.Dropout (0.50))
#add 2nd hidden layer
```

```
classifier.add(keras.layers.Dense (6, activation = 'relu'))
classifier.add(keras.layers.Dropout (0.50))
#final or output layer
classifier.add(keras.layers.Dense (1, activation = 'sigmoid'))


 #Compiling the model

loss_1 = tf. keras.losses.BinaryCrossentropy()

classifier.compile(optimizer = "Adam", loss = loss_1, metrics = ['accuracy'])


#fitting the model

classifier.fit(x_train, Y_train, batch_size = 20, epochs = 100)


classifier = svm.SVC(kernel='linear')


classifier.fit(x_train,y_train)


x_train_prediction = classifier.predict(x_train)
training_data_accuracy = accuracy_score(x_train_prediction,x_train)


print('Accuracy score of the training data:', training_data_accuracy)


best_k = {"Regular":0}
best_score= {"Regular":0}
for k in range(3, 50, 2):


## Using Regular training set
knn_temp = KNeighborsClassifier(n_neighbors=k)                          #Instantiate the
knn_temp.fit(X_train, Y_train)                                          # Fit the model t
knn_temp_pred = knn_temp.predict(X_test)                                # Predict on the
score = metrics.accuracy_score (Y_test, knn_temp_pred)*100              # Get accuracy
if score >= best_score["Regular"] and score < 100:                      # Store best para
  best_score["Regular"] = score
   best_k["Regular"] = k


## Instantiate the models
knn = KNeighborsClassifier(n_neighbors=best_k["Regular"])
## Fit the model to the training set
knn.fit(X_train, Y_train)
knn_pred = knn.predict(X_test)
testd = accuracy_score (knn_pred, Y_test)


import tensorflow as tf
from tensorflow import keras
```

```
from keras.models import Sequential
from tensorflow.keras import layers


classifier = Sequential()


#add input layer and first hidden layer
classifier.add(keras.layers.Dense (6, activation = 'relu', input_dim = 6))
classifier.add(keras.layers.Dropout (0.50))
#add 2nd hidden layer
classifier.add(keras.layers.Dense (6, activation = 'relu'))
classifier.add(keras.layers.Dropout (0.50))
#final or output layer
classifier.add(keras.layers.Dense (1, activation = 'sigmoid'))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-9-b6b12966572d> in <cell line: 1>()
----> 1 classifier = Sequential()
      2
      3
      4 #add input layer and first hidden layer
      5 classifier.add(keras.layers.Dense (6, activation = 'relu', input_dim = 6))

NameError: name 'Sequential' is not defined
```

SEARCH STACK OVERFLOW

```
 #Compiling the model

loss_1 = tf. keras.losses.BinaryCrossentropy()

classifier.compile(optimizer = "Adam", loss = loss_1, metrics = ['accuracy'])
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-5-052ca7f92f47> in <cell line: 3>()
      1 #Compiling the model
      2
----> 3 loss_1 = tf. keras.losses.BinaryCrossentropy()
      4
      5 classifier.compile(optimizer = "Adam", loss = loss_1, metrics =
['accuracy'])

NameError: name 'tf' is not defined
```

SEARCH STACK OVERFLOW

```
#fitting the model
classifier.fit(x_train, y_train, batch_size = 20, epochs = 100)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-4-7d03ebd94e82> in <cell line: 2>()
      1 #fitting the model
----> 2 classifier.fit(x_train, y_train, batch_size = 20, epochs = 100)
```

```
import pickle

pickle.dump(knn, open("placement.pk1", 'wb'))
model = pickle.load(open('placement.pkl', 'rb'))
```

0s completed at 8:59 PM