```
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("collegePlace.csv")
df.head()
```

	Age	Gender	Stream	Internships	CGPA	Hostel	HistoryOfBacklogs	Placed
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	
3	21	Male	Information Technology	0	8	0	1	
4	22	Male	Mechanical	0	8	1	0	

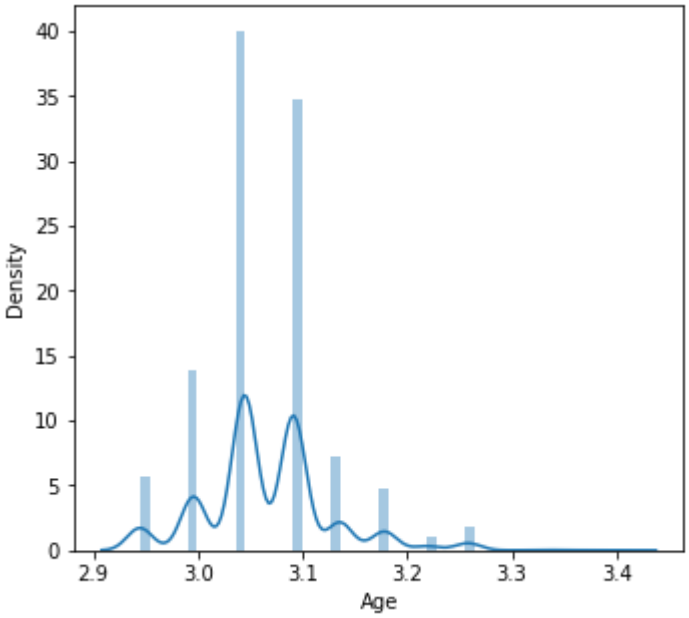
```
df.info()
```

```
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-bb709bf82109>: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)
```

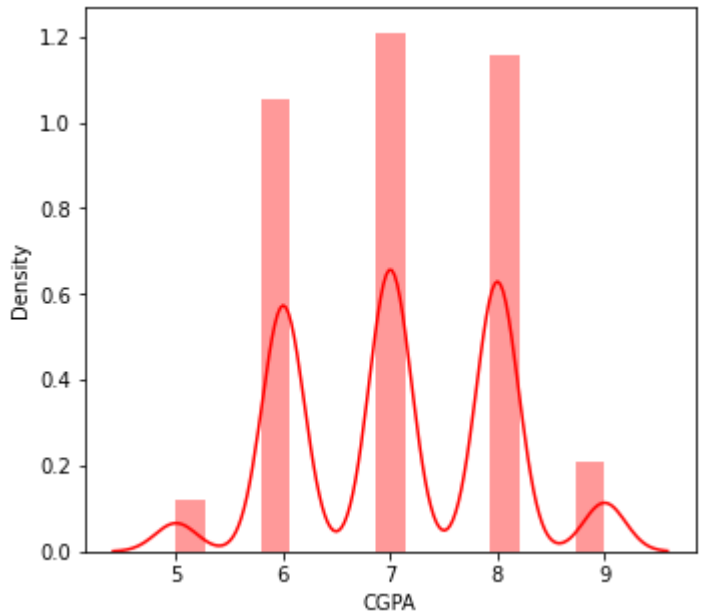



```
df = df.replace(['Male'],[0])
df = df.replace(['Female'],[1])

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

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

df
```

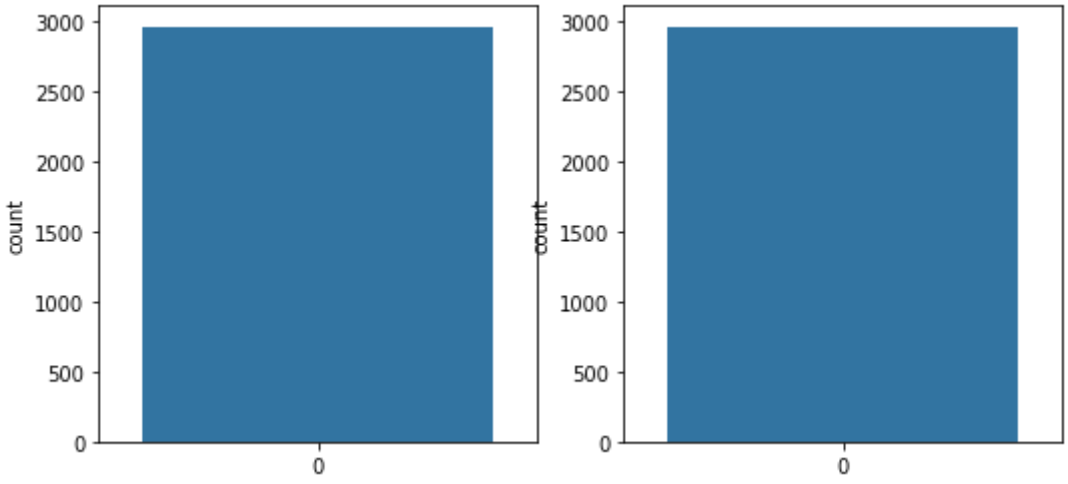
	Age	Gender	Stream	Internships	CGPA	HistoryOfBacklogs	PlacedOrNot
0	22	0	2	1	8	1	1
1	21	1	0	0	7	1	1
2	22	1	1	1	6	0	1
3	21	0	1	0	8	1	1
4	22	0	3	0	8	0	1
...	...	...	...	...	...	...	...
2961	23	0	1	0	7	0	0

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

```
<ipython-input-8-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'>
```

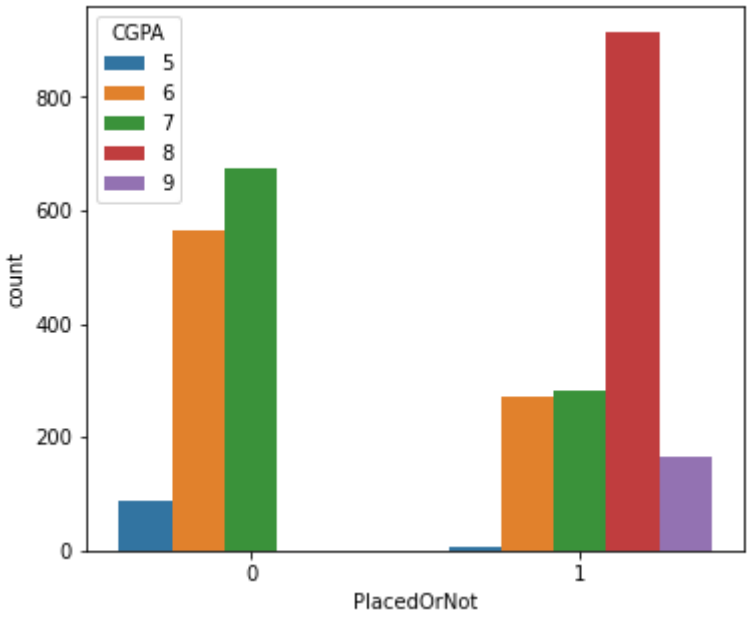



```
plt.figure(figsize=(18,4))
plt.subplot(1,4,1)
sns.countplot(df['Gender'])
plt.subplot(1,4,2)
sns.countplot(df['Stream'])
plt.show()
```




```
plt.figure(figsize=(20,5))
plt.subplot(131)
sns.countplot(data=df, x="PlacedOrNot", hue="CGPA")
```

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




```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x = sc.fit_transform(df)
x
x = pd.DataFrame(x,columns=df.columns)
```

Double-click (or enter) to edit

```
X = df.drop(['PlacedOrNot'], axis =1)
Y = df['PlacedOrNot']

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size = 0.2, stratify=Y, random_s
```

```
classifier = svm.SVC(kernel='linear')
classifier.fit(X_train, Y_train)
```

```
SVC
SVC(kernel='linear')
```

```
X_train_prediction = classifier.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)

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

```
Accuracy score of the training data: 0.7685497470489039
```

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

    knn_temp = KNeighborsClassifier(n_neighbors=k)
    knn_temp.fit(X_train, Y_train)
    knn_temp_pred = knn_temp.predict(X_test)
    score = metrics.accuracy_score(Y_test, knn_temp_pred)*100
    if score >= best_score["Regular"] and score < 100:
       best_score["Regular"] = score
       best_k["Regular"] = k

print("---Results---\nk: {}\nScore: {}".format(best_k, best_score))
knn = KNeighborsClassifier(n_neighbors=best_k["Regular"])
knn.fit(X_train, Y_train)
knn_pred = knn.predict(X_test)
testd = accuracy_score(knn_pred, Y_test)
```

```
---Results---
k: {'Regular': 21}
Score: {'Regular': 87.7104377104377}
```

```
from google.colab import drive
drive.mount('/content/drive')
```