```
import numpy as np
import pandas as pd
import matplotlib.pyplot as pd
import seaborn as sns
import csv
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving Admission_Predict.csv to Admission_Predict.csv
```

Double-click (or enter) to edit

Double-click (or enter) to edit

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving Admission_Predict.csv to Admission_Predict (2).csv
```

```
data = pd.read_csv('Admission_Predict.csv')
```

```
data.info()
```

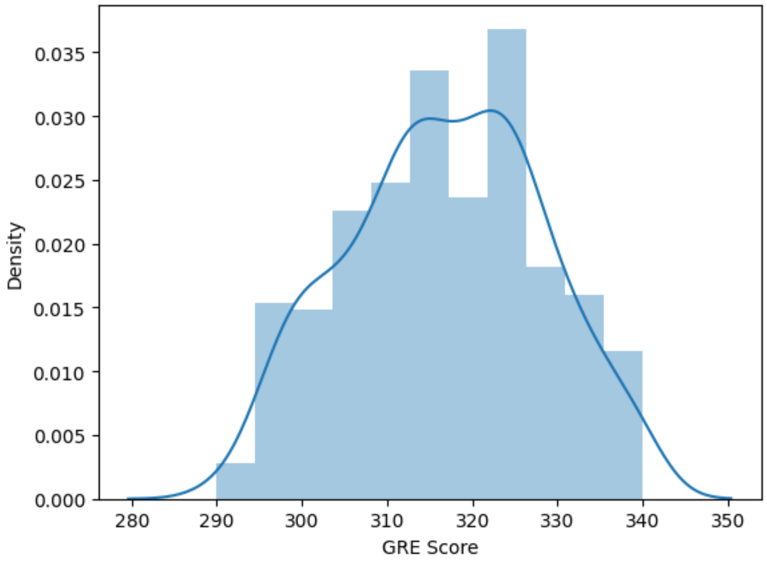
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Serial No.         400 non-null    int64
 1   GRE Score          400 non-null    int64
 2   TOEFL Score        400 non-null    int64
 3   University Rating  400 non-null    int64
 4   SOP                400 non-null    float64
 5   LOR                400 non-null    float64
 6   CGPA               400 non-null    float64
 7   Research           400 non-null    int64
 8   Chance of Admit    400 non-null    float64
dtypes: float64(4), int64(5)
memory usage: 28.2 KB
```

```
data.isnull()
```

```
   Serial    GRE    TOEFL   University   SOP   LOR   CGPA  Research     Chance
      No.  Score    Score       Rating                                  of Admit
```

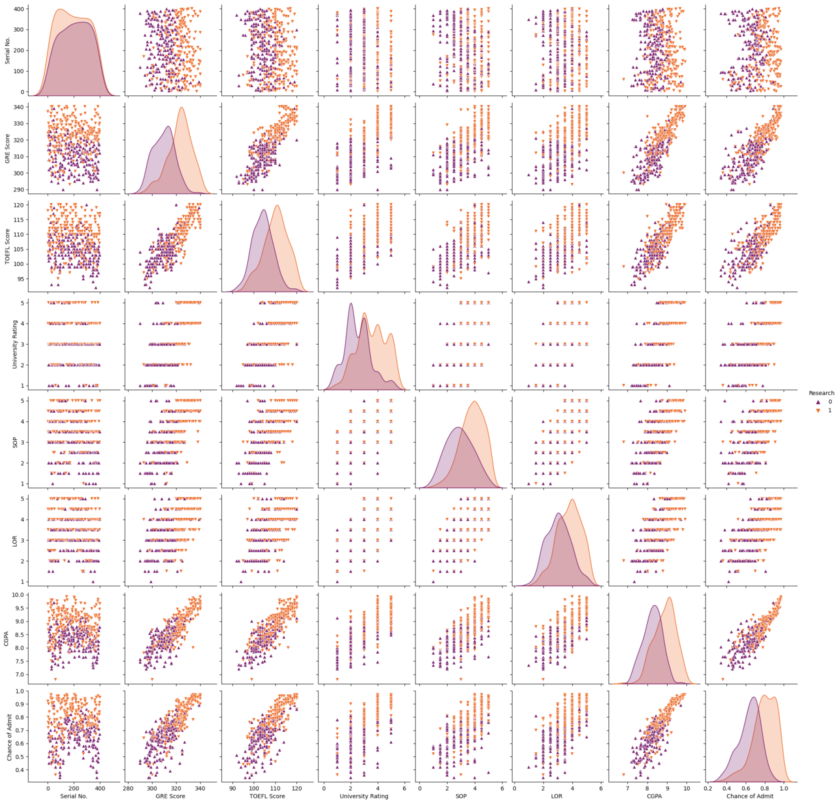
```
sns.distplot(data['GRE Score'])
```

```
<ipython-input-6-64e93544a305>:1: 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(data['GRE Score'])
<Axes: xlabel='GRE Score', ylabel='Density'>
```



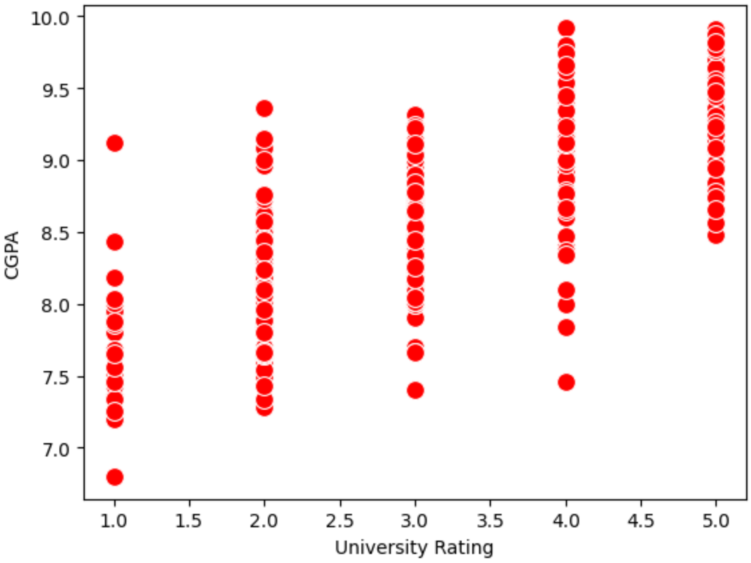
```
sns.pairplot(data=data,hue='Research',markers=["^","v"],palette='inferno')
```

```
<seaborn.axisgrid.PairGrid at 0x7f2b31912790>
```

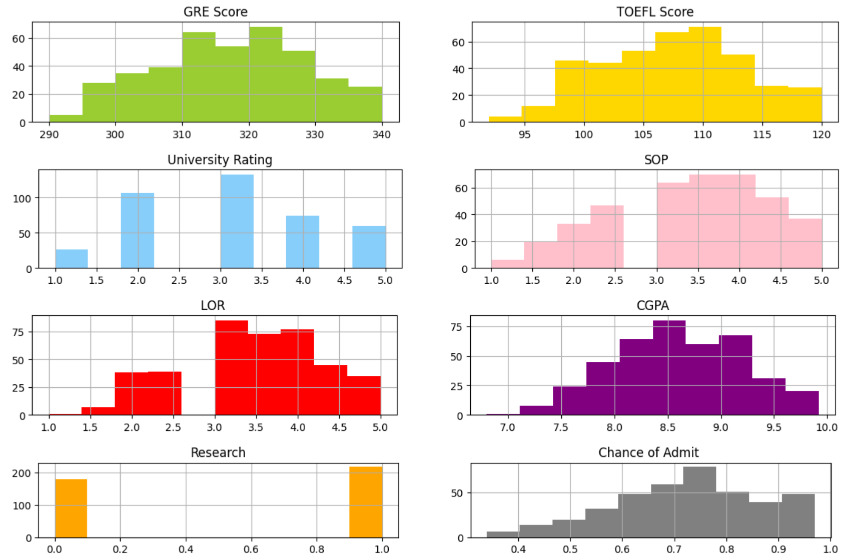


```
sns.scatterplot(x='University Rating',y='CGPA',data=data,color='Red',s=100)
```

```
<Axes: xlabel='University Rating', ylabel='CGPA'>
```



```
category = ['GRE Score','TOEFL Score','University Rating','SOP','LOR ','CGPA','Research','Chance of Admit ']
color = ['Yellowgreen','gold','lightskyblue','pink','red','purple','orange','gray']
start = True
for i in np.arange(4):
  fig = plt.figure(figsize=(14,8))
  plt.subplot2grid((4,2),(i,0))
  data[category[2*i]]. hist(color=color[2*i],bins=10)
  plt.title(category[2*i])
  plt.subplot2grid((4,2),(i,1))
  data[category[2*i+1]].hist(color=color[2*i+1],bins=10)
  plt.title(category[2*i+1])
plt.subplots_adjust(hspace = 0.7, wspace = 0.2)
plt.show()
```



```
from sklearn.preprocessing import MinMaxScaler
sc=MinMaxScaler()
x=sc.fit_transform(x)
x
```

```
x=data.iloc[:,0:7].values
x
```

```
array([[  1.  , 337.  , 118.  , ...,   4.5 ,   4.5 ,   9.65],
       [  2.  , 324.  , 107.  , ...,   4.  ,   4.5 ,   8.87],
       [  3.  , 316.  , 104.  , ...,   3.  ,   3.5 ,   8.  ],
       ...,
       [398.  , 330.  , 116.  , ...,   5.  ,   4.5 ,   9.45],
       [399.  , 312.  , 103.  , ...,   3.5 ,   4.  ,   8.78],
       [400.  , 333.  , 117.  , ...,   5.  ,   4.  ,   9.66]])
```

```
y=data.iloc[:,7:].values
y
```

```
array([[1.  , 0.92],
       [1.  , 0.76],
       [1.  , 0.72],
       [1.  , 0.8 ],
       [0.  , 0.65],
       [1.  , 0.9 ],
       [1.  , 0.75],
       [0.  , 0.68],
       [0.  , 0.5 ],
       [0.  , 0.45],
       [1.  , 0.52],
       [1.  , 0.84],
       [1.  , 0.78],
       [1.  , 0.62],
       [1.  , 0.61],
       [0.  , 0.54],
       [0.  , 0.66],
       [1.  , 0.65],
       [0.  , 0.63],
       [0.  , 0.62],
       [1.  , 0.64],
       [0.  , 0.7 ],
       [1.  , 0.94],
       [1.  , 0.95],
       [1.  , 0.97],
       [1.  , 0.94],
       [0.  , 0.76],
       [1.  , 0.44],
       [0.  , 0.46],
       [0.  , 0.54],
       [1.  , 0.65],
       [1.  , 0.74],
       [1.  , 0.91],
       [1.  , 0.9 ],
       [1.  , 0.94],
       [1.  , 0.88],
       [0.  , 0.64],
       [0.  , 0.58],
       [0.  , 0.52],
       [0.  , 0.48],
       [1.  , 0.46],
       [1.  , 0.49],
       [1.  , 0.53],
       [0.  , 0.87],
       [1.  , 0.91],
       [1.  , 0.88],
       [1.  , 0.86],
       [0.  , 0.89],
       [1.  , 0.82],
       [1.  , 0.78],
       [1.  , 0.76],
       [1.  , 0.56],
       [1.  , 0.78],
       [1.  , 0.72],
       [0.  , 0.7 ],
       [0.  , 0.64],
       [0.  , 0.64],
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y, test_size=0.30,random_state=101)
```

```
y_train=(y_train>0.5)
y_train
```

```
    array([[ True,  True],
           [False,  True],
           [ True,  True],
           [ True,  True],
           [ True, False],
           [ True,  True],
           [ True,  True],
           [ True,  True],
           [False,  True],
           [ True,  True],
           [ True,  True],
           [False,  True],
           [ True,  True],
           [ True,  True],
           [False,  True],
           [False, False],
           [ True, False],
           [False,  True],
           [ True,  True],
           [False,  True],
           [False, False],
           [False,  True],
           [False,  True],
           [ True,  True],
           [ True,  True],
           [ True,  True],
           [ True,  True],
           [ True,  True],
           [False,  True],
           [False,  True],
           [ True,  True],
           [False,  True],
           [False,  True],
           [False,  True],
           [False,  True],
           [ True,  True],
           [ True,  True],
           [False,  True],
           [ True,  True],
           [False,  True],
           [False,  True],
           [ True,  True],
           [ True,  True],
           [False,  True],
           [ True,  True],
           [ True,  True],
           [False,  True],
           [ True,  True],
           [ True,  True],
           [False,  True],
           [ True,  True],
           [ True,  True],
           [False,  True],
           [ True,  True],
           [False,  True],
           [ True,  True],
           [False, False],
           [False,  True]
```

```
y_test=(y_test>0.5)
```

```
from sklearn.linear_model import LogisticRegression
cls=LogisticRegression(randam_state=0)
Ir=cls.fit(x_train,y_train)
y_pred=Ir.predict(x_test)
y_pred
```

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.layers import Dence, Activation,Dropout
from tensorflow.Keras.optimizers import Adam
```

```
 mode1=keras.sequential()
 mode1.add(Dense(7,activation ='relu',input_dim=7))
 mode1.add(Dense(7,activation='relu'))
 mode1.add(Dense(1,activation='linear'))
 mode1.summary()
 model:"sequential"
```

```
model.summary()
model: "sequential"
```

```
model.fit(x_train,y_train,batch_size=20,epochs=100)
```

```
model.compile(loss= 'binary_crossentropy',optimizer= 'adam',metrics=['accuracy'])
```

```
model.fit(x_train,y_train, batch_size=20, epochs=100)
```

```
from sklearn.metrics import accuracy_score
train_predictions= model.predict(x_train)
print(train_predictions)
```

```
train_acc= model.evaluate(x_train, y_train, verbose=0)[1]
print(train_acc)
```

```
test_acc = model.evaluate(x_test,y_test,verbose=0)[1]
print(test_acc)
```

```
print(classification report(v test.pred))
```

```
  File "<ipython-input-27-b549722b754f>", line 1
    print(classification report(v test.pred))
                               ^
SyntaxError: invalid syntax
```

SEARCH STACK OVERFLOW

```
pred=model.predict(x_test)
pred = (pred>0.5)
pred
```

```
from sklearn.metrics import accuracy_score,recall_score,roc_auc_score,confusion_matrix
print("\nAccuracy score : %f" %(accuracy_score(y_test,y_pred)*100))
print("\nRecall score : %f" %(recall_score(y_test,y_pred)*100))
print("ROC score : %f\n" %(roc_auc_score(y_test,y_pred)*100))
print(confusion_matrix(y_test,y__pred))
```

```
, classification_report
from sklearn.metrics import accuracy_score,recall_score,roc_auc_score,confusion_matr:
print(classification_report(y_test,pred))
```

```
model.save('model.h5')
```

```
import numpy as np
from flask import Flask,request,jsonify,render_template
import pickle
app = Flask(_name_)
from tensorflow.keras.models import linear_model
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-44-21bc3785ce0e> in <cell line: 4>()
      2 from flask import Flask,request,jsonify,render_template
      3 import pickle
----> 4 app = Flask(_name_)
      5 from tensorflow.keras.models import linear_model
      6

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

SEARCH STACK OVERFLOW

```
model = load_model('model.h5')
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-45-46901180ca61> in <cell line: 1>()
----> 1 model=load

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

SEARCH STACK OVERFLOW

```
def home():
  return render_template('Dem02.html')
```

```
def home():
  return render_template('Demo2.html')
def y_predict():
 min1=[290.0,92.0,1.0,1.0,6.8,0.0]
 max1=[340,120.0,5.0,5.0,5.0,9.92,1.0]
 k=[float(x)for x in request.form.values()]
 p=[]
 for i in range(7):
   L=(k[i]-min[i])/(max1[i]-min1[i])
   P.append(L)
 prediction=model.predict([p])
 print(prediction)
 output=prediction[0]
 if(output==False):
   return render_template('noChance.html',prediction_text='You Dont have a chance of getting')
 else:
   return render_template('chance.html',prediction_text='You have a chance of getting admission')
if __name__=="__main__":
  app.run(debug=False)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-92-7b18bce7f2f6> in <cell line: 18>()
     17    return render_template('chance.html',prediction_text='You have a
chance of getting admission')
     18 if __name__=="__main__":
---> 19   app.run(debug=False)
     20

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

SEARCH STACK OVERFLOW

```
      return render_template()
  else:
      return render_template()
if __name__=="__main__":
  app.run(debug=False)
```