```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
data = pd.read_csv("/content/Admission_Predict.csv")
```

```
data.head(8)
```

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76
2	3	316	104	3	3.0	3.5	8.00	1	0.72
3	4	322	110	3	3.5	2.5	8.67	1	0.80
4	5	314	103	2	2.0	3.0	8.21	0	0.65
5	6	330	115	5	4.5	3.0	9.34	1	0.90
				3	3.0	4.0	8.20	1	0.75
				2	3.0	4.0	7.90	0	0.68


```
data.drop(["Serial No."],axis=1,inplace=True)
data.head()
```

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	337	118	4	4.5	4.5	9.65	1	0.92
1	324	107	4	4.0	4.5	8.87	1	0.76
2	316	104	3	3.0	3.5	8.00	1	0.72
3	322	110	3	3.5	2.5	8.67	1	0.80
4	314	103	2	2.0	3.0	8.21	0	0.65

```
data.describe()
```

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Resear
count	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.0000
mean	316.807500	107.410000	3.087500	3.400000	3.452500	8.598925	0.5475
std	11.473646	6.069514	1.143728	1.006869	0.898478	0.596317	0.4983
min	290.000000	92.000000	1.000000	1.000000	1.000000	6.800000	0.0000
25%	308.000000	103.000000	2.000000	2.500000	3.000000	8.170000	0.0000
50%	317.000000	107.000000	3.000000	3.500000	3.500000	8.610000	1.0000
75%	325.000000	112.000000	4.000000	4.000000	4.000000	9.062500	1.0000

```
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().any()
```

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

```
data.corr()
```

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Cha Ac
GRE Score	1.000000	0.835977	0.668976	0.612831	0.557555	0.833060	0.580391	0.802
TOEFL Score	0.835977	1.000000	0.695590	0.657981	0.567721	0.828417	0.489858	0.791
University Rating	0.668976	0.695590	1.000000	0.734523	0.660123	0.746479	0.447783	0.711
				0.734523	1.000000	0.729593	0.718144	0.444029	0.675
				0.660123	0.729593	1.000000	0.670211	0.396859	0.669

Creating a copy...

```
plt.figure(figsize=(10,7))
```

```
sns.heatmap(data.corr(),annot=True,cmap="RdYlGn")
```

```
<Axes: >
```

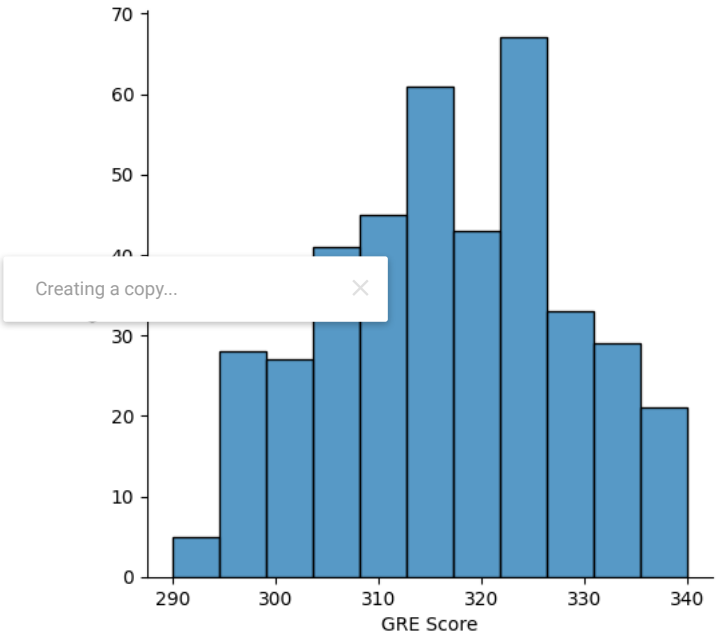

```
data=data.rename(columns = {'Chance of Admit ':'Chance of Admit'})
```

```
data.describe()
```

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Resear
count	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.0000
mean	316.807500	107.410000	3.087500	3.400000	3.452500	8.598925	0.5475
std	11.473646	6.069514	1.143728	1.006869	0.898478	0.596317	0.4983
min	290.000000	92.000000	1.000000	1.000000	1.000000	6.800000	0.0000
25%	308.000000	103.000000	2.000000	2.500000	3.000000	8.170000	0.0000
50%	317.000000	107.000000	3.000000	3.500000	3.500000	8.610000	1.0000
75%	325.000000	112.000000	4.000000	4.000000	4.000000	9.062500	1.0000

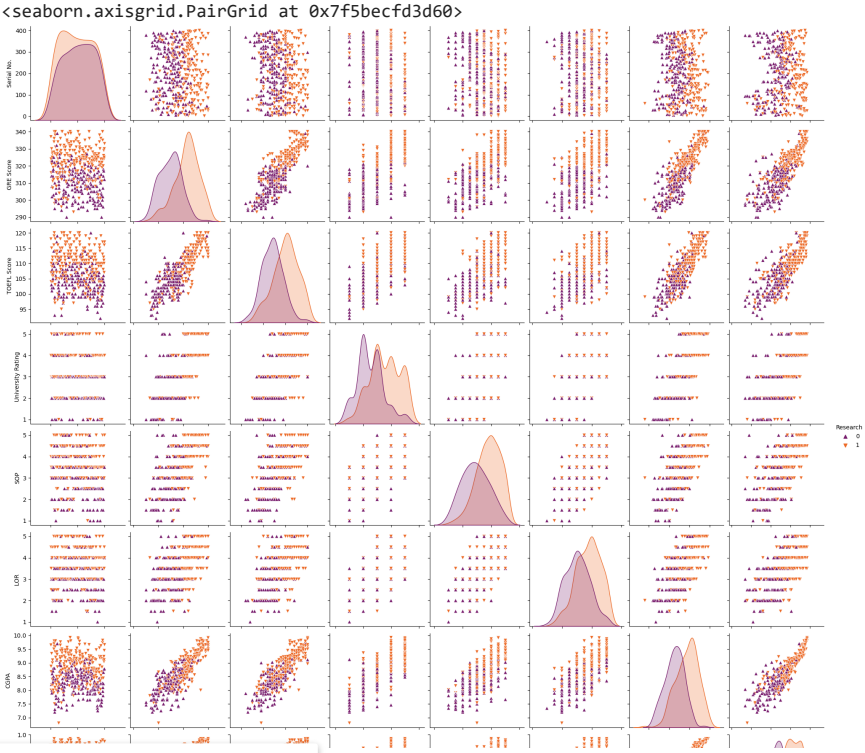
```
sns.displot(data['GRE Score'])
```

```
<seaborn.axisgrid.FacetGrid at 0x7f5bed185be0>
```



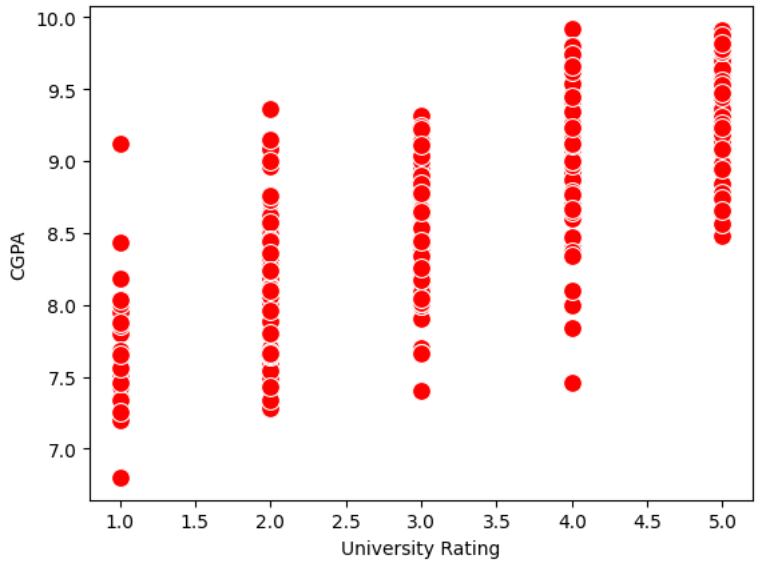

```
sns.pairplot(data=data,hue='Research',markers=["^","v"],palette='inferno')
```

```
<seaborn.axisgrid.PairGrid at 0x7f5becfd3d60>
```



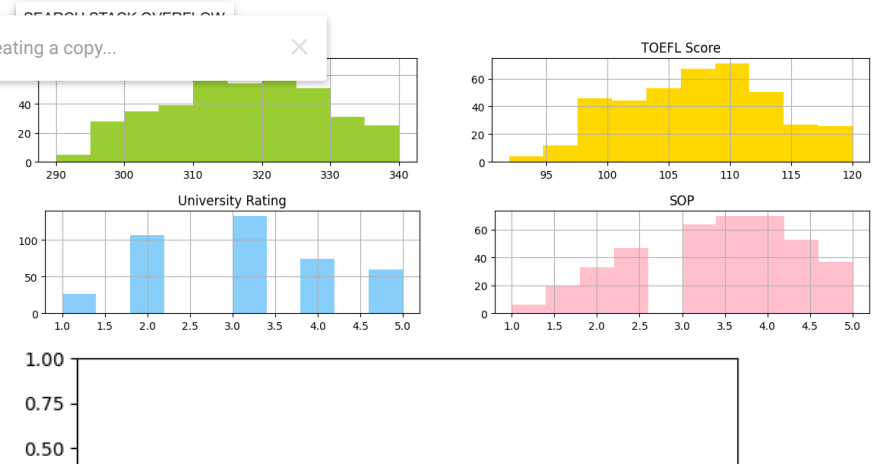
Creating a copy...

```
',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()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/usr/local/lib/python3.9/dist-packages/pandas/core/indexes/base.py in get_loc(self,
key, method, tolerance)
   3801             try:
-> 3802                 return self._engine.get_loc(casted_key)
   3803             except KeyError as err:

                                 4 frames

pandas/_libs/hashtable_class_helper.pxi in
pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in
pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'LOR'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
/usr/local/lib/python3.9/dist-packages/pandas/core/indexes/base.py in get_loc(self,
key, method, tolerance)
   3802                 return self._engine.get_loc(casted_key)
   3803             except KeyError as err:
-> 3804                 raise KeyError(key) from err
   3805             except TypeError:
   3806                 # If we have a listlike key, _check_indexing_error will
raise

KeyError: 'LOR'
```


```
print('Mean CGPA Score is :',int(data['CGPA'].mean()))
print('Mean GRE Score is :',int(data['GRE Score'].mean()))
print('Mean TOEFL Score is :',int(data['TOEFL Score'].mean()))
```

```
Mean CGPA Score is : 8
Mean GRE Score is : 316
Mean TOEFL Score is : 107
```

```
data.head()
```

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	337	118	4	4.5	4.5	9.65	1	0.92
1	324	107	4	4.0	4.5	8.87	1	0.76
2	316	104	3	3.0	3.5	8.00	1	0.72
3	322	110	3	3.5	2.5	8.67	1	0.80
4	314	103	2	2.0	3.0	8.21	0	0.65

```
x=data.iloc[:,0:-1].values
```

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

```
array([[0.94       , 0.92857143, 0.75       , ..., 0.875      , 0.91346154,
        1.        ],
       [0.68       , 0.53571429, 0.75       , ..., 0.875      , 0.66346154,
        1.        ],
```

```
       [0.52      , 0.42857143, 0.5       , ..., 0.625     , 0.38461538,
        1.        ],
       ...,
       [0.8       , 0.85714286, 0.75      , ..., 0.875     , 0.84935897,
        1.        ],
       [0.44      , 0.39285714, 0.5       , ..., 0.75      , 0.63461538,
        0.        ],
       [0.86      , 0.89285714, 0.75      , ..., 0.75      , 0.91666667,
        1.        ]])
```

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

```
array([[337.  , 118.  ,   4.  , ...,   4.5 ,   9.65,   1.  ],
       [324.  , 107.  ,   4.  , ...,   4.5 ,   8.87,   1.  ],
       [316.  , 104.  ,   3.  , ...,   3.5 ,   8.  ,   1.  ],
       ...,
       [330.  , 116.  ,   4.  , ...,   4.5 ,   9.45,   1.  ],
       [312.  , 103.  ,   3.  , ...,   4.  ,   8.78,   0.  ],
       [333.  , 117.  ,   4.  , ...,   4.  ,   9.66,   1.  ]])
```

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

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

Creating a copy...

```
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],
       [ True],
       [ True],
       [False],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [False],
       [False],
       [ True],
       [ True],
       [ True],
       [False],
       [ True],
       [ True],
       [ True],
       [ True],
```



```
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [False],
       [ True],
```

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

```
def logreg(x_train,x_test,y_train,y_test):
    lr = LogisticRegression(random_state=0)
    lr.fit(x_train,y_train)
    y_lr_tr = lr.predict(x_train))
    print(accuracy_score(y_lr_tr,y_train))
    ypred_lr = lr.predict(x_test)
    ptint(accuracy_score(ypred_lr,y_test))
    print("***Logistic Regression***")
    print("Confusion_Matrix")
    print(confusion_matrix(y_test,ypred_lr))
    print("classification Report")
    print(classification_report(y_test,ypred_lr))
```

```
  File "<ipython-input-51-4b56dac921c6>", line 4
```

```
from sklearn.linear_model.logistic import LogisticRegression
cls =LogisticRegression(random_state =0)

lr=cls.fit(x_train, y_train)

y_pred =lr.predict(x_test)
y_pred
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-50-65477ef1674c> in <cell line: 1>()
----> 1 from sklearn.linear_model.logistic import LogisticRegression
      2 cls =LogisticRegression(random_state =0)
      3
      4 lr=cls.fit(x_train, y_train)
      5

ModuleNotFoundError: No module named 'sklearn.linear_model.logistic'

---------------------------------------------------------------------------
NOTE: If your import is failing due to a missing package, you can
manually install dependencies using either !pip or !apt.

To view examples of installing some common dependencies, click the
"Open Examples" button below.
---------------------------------------------------------------------------
```

SEARCH STACK OVERFLOW

Creating a copy...

```
from tensorflow.keras.layers import Dense, Activation, Dropout
from tensorflow.keras.optimizers import Adam


model=keras.Sequential()


model.add(Dense(7,activation ='relu',input_dim=7))


model.add(Dense(7,activation='relu'))


model.add(Dense(1,activation='linear'))


model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 7)                 56

 dense_1 (Dense)             (None, 7)                 56

 dense_2 (Dense)             (None, 1)                 8

=================================================================
Total params: 120
Trainable params: 120
Non-trainable params: 0
_________________________________________________________________
```

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

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-59-657a427e00f1> in <cell line: 1>()
----> 1 model.fit(x_train, y_train, batch_size = 20, Epochs = 100)

                                 1 frames
/usr/local/lib/python3.9/dist-packages/keras/utils/traceback_utils.py in error_handler(*args, **kwargs)
     63         filtered_tb = None
     64         try:
---> 65             return fn(*args, **kwargs)
     66         except Exception as e:
     67             filtered_tb = _process_traceback_frames(e.__traceback__)

TypeError: fit() got an unexpected keyword argument 'Epochs'
```

SEARCH STACK OVERFLOW

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


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

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-61-9e483767323c> in <cell line: 1>()
----> 1 model.fit(x_train, y_train, bath_size = 20, epochs = 100)
      2

 1 frames
/usr/local/lib/python3.9/dist-packages/keras/utils/traceback_utils.py in
error_handler(*args, **kwargs)
     63         filtered_tb = None
     64         try:
---> 65             return fn(*args, **kwargs)
     66         except Exception as e:
     67             filtered_tb = _process_traceback_frames(e.__traceback__)

TypeError: fit() got an unexpected keyword argument 'bath_size'

SEARCH STACK OVERFLOW
```

```
from sklearn.metrics import accuracy_score


train_predictions = model.predict(x_train)

print(train_predictions)
```

Creating a copy...

```
====] - 0s 2ms/step
[[12.924982 ]
 [12.831165 ]
 [12.793592 ]
 [13.070606 ]
 [12.947409 ]
 [13.362659 ]
 [13.016581 ]
 [13.414518 ]
 [12.538475 ]
 [13.044149 ]
 [12.895784 ]
 [12.343576 ]
 [12.875975 ]
 [12.790312 ]
 [12.593692 ]
 [11.823761 ]
 [12.244467 ]
 [12.0178585]
 [13.014093 ]
 [12.839886 ]
 [12.635277 ]
 [12.388715 ]
 [11.705034 ]
 [13.090451 ]
 [13.309011 ]
 [13.165588 ]
 [13.073106 ]
 [12.473944 ]
 [12.387314 ]
 [12.34285  ]
 [12.858695 ]
 [12.979944 ]
 [12.387562 ]
 [13.027916 ]
 [12.401426 ]
 [13.638262 ]
 [13.151957 ]
 [12.450689 ]
 [12.961393 ]
 [12.352225 ]
 [12.980994 ]
 [13.208031 ]
 [13.335403 ]
 [12.143257 ]
 [12.711775 ]
 [13.421017 ]
 [12.454525 ]
 [12.886136 ]
 [11.945795 ]
 [12.81345  ]
 [13.130873 ]
 [12.457009 ]
 [12.524707 ]
 [12.580418 ]
 [12.007471 ]
```

```
 [13.149942 ]
 [12.254769 ]
```

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

print(train_acc)
```

```
0.9178571701049805
```

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

print(test_acc)
```

```
0.8999999761581421
```

```
print(classification report(y_test,pred))
```

```
  File "<ipython-input-65-7b39d62fdf57>", line 1
    print(classification report(y_test,pred))
                                ^
SyntaxError: invalid syntax
```

SEARCH STACK OVERFLOW

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



```
====] - 0s 3ms/step
array([[ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
```

```
            [ True],
            [ True],
```

```
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("Recall 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))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-78-b6c30a1fb899> in <cell line: 2>()
      1 from sklearn.metrics import accuracy_score,recall_score,roc_auc_score,confusion_matrix
----> 2 print("\nAccuracy score: %f" %(accuracy_score(y_test,y_pred) * 100))
      3 print("Recall score : %f" %(recall_score(y_test,y_pred) * 100))
      4 print("ROC score : %f\n" %(roc_auc_score(y_test,y_pred) * 100))
      5 print(confusion_matrix(y_test,y_pred))

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

SEARCH STACK OVERFLOW

```
from sklearn.metrics import accuracy_score,recall_score,roc_auc_score,confusion_matrix

print(classification_report(y_train,pred))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-82-b9f5ff78e2ab> in <cell line: 3>()
      1 from sklearn.metrics import accuracy_score,recall_score,roc_auc_score,confusion_matrix

----> 3 print(classification_report(y_train,pred))

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

Creating a copy...

SEARCH STACK OVERFLOW

```
from sklearn.metrics import accuracy_score,recall_score,roc_auc_score,confusion_matrix

print(classification_report(y_test,pred))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-83-d324e7a1d5dc> in <cell line: 3>()
      1 from sklearn.metrics import accuracy_score,recall_score,roc_auc_score,confusion_matrix
      2
----> 3 print(classification_report(y_test,pred))

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

SEARCH STACK OVERFLOW

```
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 load_model
```

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

```
@app.route('/')
def home():
    return render_template('Demo2.html')
```

```
@app.route('/')
def home():
    return render_template('Demo2.html')

@app.route('/y_predict',methods=['POST'])
def y_predict():
```

```
    min1=[290.0, 92.0, 1.0, 1.0, 1.0, 6.8, 0.0]
    max1=[340.0, 120.0, 5.0, 5.0, 5.0, 9.92, 1.0]
    k= [float(x) for x in request.from.values()]
    p=[]
    for i in range(7):
        l=(k[i]-min1[i])/(max1[i]-min1[i])
        p.append(1)
    prediction = model.predict([p])
    print(prediction)
    output=prediction[0]
    if(output==False):
        return render_template('noChance.html', prediction_text='You Dont havea chance of getting adims')
    else:
        return render_template('noChance.html', prediction_text='You Dont havea chance of getting adims')
if __name__== "__main__":
    app.run(debug=False)
```

```
  File "<ipython-input-76-97b20ac113df>", line 12
    k= [float(x) for x in request.from.values()]
                                      ^
SyntaxError: invalid syntax
```

SEARCH STACK OVERFLOW

```
pickle.dump(lr,open('university.pkl','wb'))
```

```
---------------------------------------------------------------------------
                                          Traceback (most recent call last)
                                         in <cell line: 1>()
                                         iversity.pkl','wb'))
NameError: name 'pickle' is not defined
```

SEARCH STACK OVERFLOW

Creating a copy...

Double-click (or enter) to edit

Double-click (or enter) to edit