In [2]:

*#Name:Pawar ved balsaheb(T512037)*

**import** numpy **as** np

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

In [4]:

*#Loading data into dataframe*

data **=** pd**.**read\_csv("Admission\_Predict.csv")

In [6]:

data**.**head()

Out[6]:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **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 |

In [8]:

data**.**tail()

Out[8]:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Serial No.** | **GRE**  **Score** | **TOEFL**  **Score** | **University**  **Rating** | **SOP** | **LOR** | **CGPA** | **Research** | **Chance of**  **Admit** |
| **395** | 396 | 324 | 110 | 3 | 3.5 | 3.5 | 9.04 | 1 | 0.82 |
| **396** | 397 | 325 | 107 | 3 | 3.0 | 3.5 | 9.11 | 1 | 0.84 |
| **397** | 398 | 330 | 116 | 4 | 5.0 | 4.5 | 9.45 | 1 | 0.91 |
| **398** | 399 | 312 | 103 | 3 | 3.5 | 4.0 | 8.78 | 0 | 0.67 |
| **399** | 400 | 333 | 117 | 4 | 5.0 | 4.0 | 9.66 | 1 | 0.95 |

In [10]:

data**.**shape

Out[10]:

In [12]:

Out[12]:

In [14]:

In [16]:

data

(400, 9)

data**.**columns

Index(['Serial No.', 'GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR ', 'CGPA', 'Research', 'Chance of Admit '],

dtype='object')

data**.**drop("Serial No.",axis**=**1,inplace**=True**)

Out[16]:

In [18]:

In [20]:

Out[20]:

In [22]:

*#Find missing values*

print("Missing values:\n") data**.**isnull()**.**sum()

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **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 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **395** | 324 | 110 | 3 | 3.5 | 3.5 | 9.04 | 1 | 0.82 |
| **396** | 325 | 107 | 3 | 3.0 | 3.5 | 9.11 | 1 | 0.84 |
| **397** | 330 | 116 | 4 | 5.0 | 4.5 | 9.45 | 1 | 0.91 |
| **398** | 312 | 103 | 3 | 3.5 | 4.0 | 8.78 | 0 | 0.67 |
| **399** | 333 | 117 | 4 | 5.0 | 4.0 | 9.66 | 1 | 0.95 |

400 rows × 8 columns

data["Chance of Admit "]**=**data["Chance of Admit "]**.**apply(**lambda** x: 1 **if** x**>**0.5 **els**

data

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **GRE**  **Score** | **TOEFL**  **Score** | **University**  **Rating** | **SOP** | **LOR** | **CGPA** | **Research** | **Chance of**  **Admit** |
| **0** | 337 | 118 | 4 | 4.5 | 4.5 | 9.65 | 1 | 1 |
| **1** | 324 | 107 | 4 | 4.0 | 4.5 | 8.87 | 1 | 1 |
| **2** | 316 | 104 | 3 | 3.0 | 3.5 | 8.00 | 1 | 1 |
| **3** | 322 | 110 | 3 | 3.5 | 2.5 | 8.67 | 1 | 1 |
| **4** | 314 | 103 | 2 | 2.0 | 3.0 | 8.21 | 0 | 1 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **395** | 324 | 110 | 3 | 3.5 | 3.5 | 9.04 | 1 | 1 |
| **396** | 325 | 107 | 3 | 3.0 | 3.5 | 9.11 | 1 | 1 |
| **397** | 330 | 116 | 4 | 5.0 | 4.5 | 9.45 | 1 | 1 |
| **398** | 312 | 103 | 3 | 3.5 | 4.0 | 8.78 | 0 | 1 |
| **399** | 333 | 117 | 4 | 5.0 | 4.0 | 9.66 | 1 | 1 |

400 rows × 8 columns

Missing values:

Out[22]:

In [24]:

GRE Score 0

TOEFL Score 0

University Rating 0

SOP 0

LOR 0

CGPA 0

Research 0

Chance of Admit 0

dtype: int64

data**.**info()

<class 'pandas.core.frame.DataFrame'> RangeIndex: 400 entries, 0 to 399

Data columns (total 8 columns):

# Column Non-Null Count Dtype

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 |  | GRE Score | 400 | non-null |  | int64 |
| 1 |  | TOEFL Score | 400 | non-null |  | int64 |
| 2 |  | University Rating | 400 | non-null |  | int64 |
| 3 |  | SOP | 400 | non-null |  | float64 |
| 4 |  | LOR | 400 | non-null |  | float64 |
| 5 |  | CGPA | 400 | non-null |  | float64 |
| 6 |  | Research | 400 | non-null |  | int64 |
| 7 |  | Chance of Admit | 400 | non-null |  | int64 |

dtypes: float64(3), int64(5) memory usage: 25.1 KB

data**.**corr()

In [26]:

Out[26]:

In [28]:

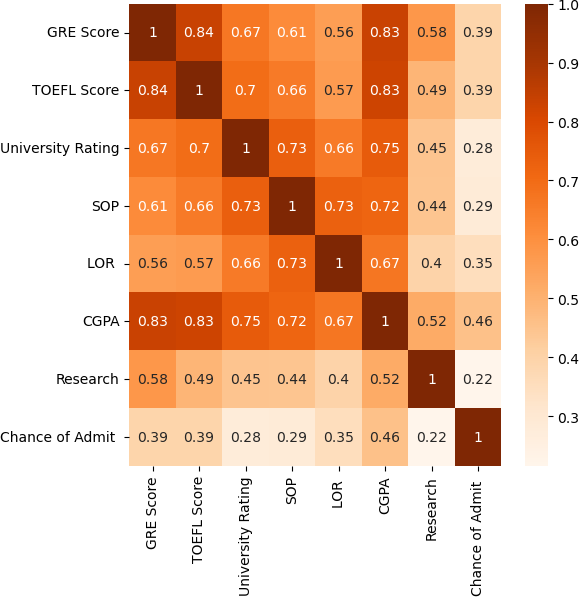
plt**.**figure(figsize**=**(6,6))

sns**.**heatmap(data**.**corr(), annot**=True**, cmap**=**'Oranges') plt**.**show()

**Score Score Rating**

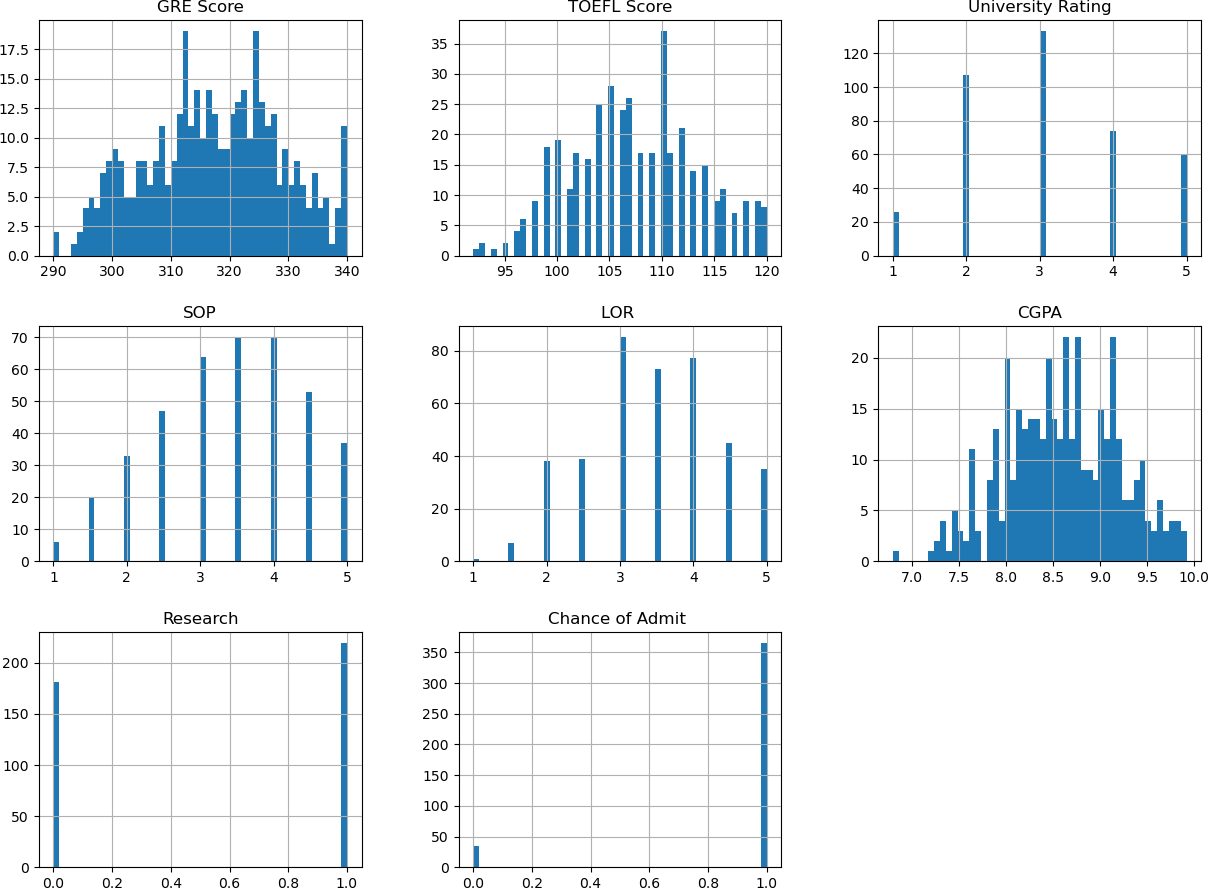
**Admit**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **GRE**  **Score** | **TOEFL**  **Score** | **University**  **Rating** | **SOP** | **LOR** | **CGPA** | **Research** | **Cha**  **Ad** |
| **GRE** 1.000000 | 0.835977 | 0.668976 | 0.612831 | 0.557555 | 0.833060 | 0.580391 | 0.390 |
| **TOEFL** 0.835977 | 1.000000 | 0.695590 | 0.657981 | 0.567721 | 0.828417 | 0.489858 | 0.393 |
| **University** 0.668976 | 0.695590 | 1.000000 | 0.734523 | 0.660123 | 0.746479 | 0.447783 | 0.279 |
| **SOP** 0.612831 | 0.657981 | 0.734523 | 1.000000 | 0.729593 | 0.718144 | 0.444029 | 0.285 |
| **LOR** 0.557555 | 0.567721 | 0.660123 | 0.729593 | 1.000000 | 0.670211 | 0.396859 | 0.353 |
| **CGPA** 0.833060 | 0.828417 | 0.746479 | 0.718144 | 0.670211 | 1.000000 | 0.521654 | 0.455 |
| **Research** 0.580391 | 0.489858 | 0.447783 | 0.444029 | 0.396859 | 0.521654 | 1.000000 | 0.216 |
| **Chance of** 0.390875 | 0.393121 | 0.279316 | 0.285939 | 0.353341 | 0.455949 | 0.216193 | 1.000 |



In [30]:

data**.**hist(bins **=** 50,figsize **=** (15,11));



In [32]:

data\_admit **=** data[data['Chance of Admit ']**==**1]

data\_non\_admit **=** data[data['Chance of Admit ']**==**0]

print("Admitted count : " ,data\_admit**.**shape[0])

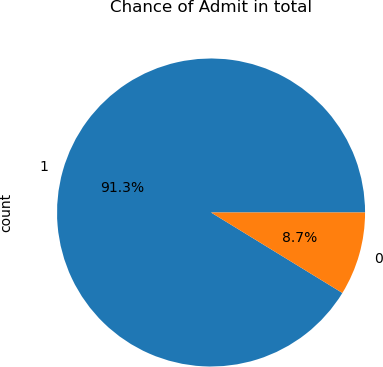
print("Non - Admitted count : " ,data\_non\_admit**.**shape[0])

Admitted count : 365 Non - Admitted count : 35

In [34]:

data['Chance of Admit ']**.**value\_counts()**.**plot(kind**=**'pie',figsize**=**(5,5),autopct**=**'% plt**.**title("Chance of Admit in total")

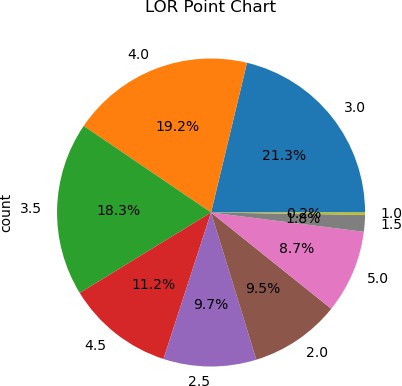
plt**.**show()



In [36]:

data['LOR ']**.**value\_counts()**.**plot(kind**=**'pie',figsize**=**(5,5),autopct**=**'%1.1f%%') plt**.**title("LOR Point Chart")

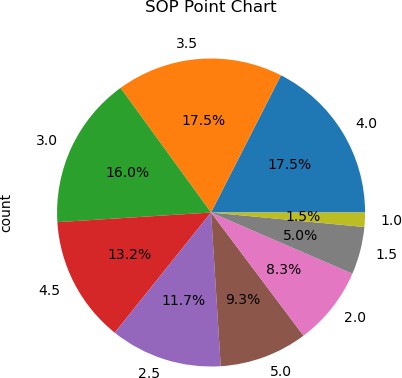
plt**.**show()



In [38]:

data['SOP']**.**value\_counts()**.**plot(kind**=**'pie',figsize**=**(5,5),autopct**=**'%1.1f%%') plt**.**title("SOP Point Chart")

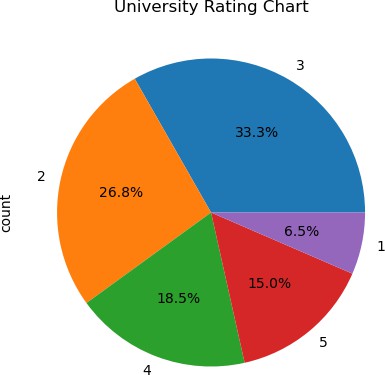
plt**.**show()



In [40]:

data["University Rating"]**.**value\_counts()**.**plot(kind**=**'pie',figsize**=**(5,5),autopct**=**' plt**.**title("University Rating Chart")

plt**.**show()



In [42]:

*#highest GRE score*

print("maximum GRE Score : ",data['GRE Score']**.**max())

*#lowest GRE score*

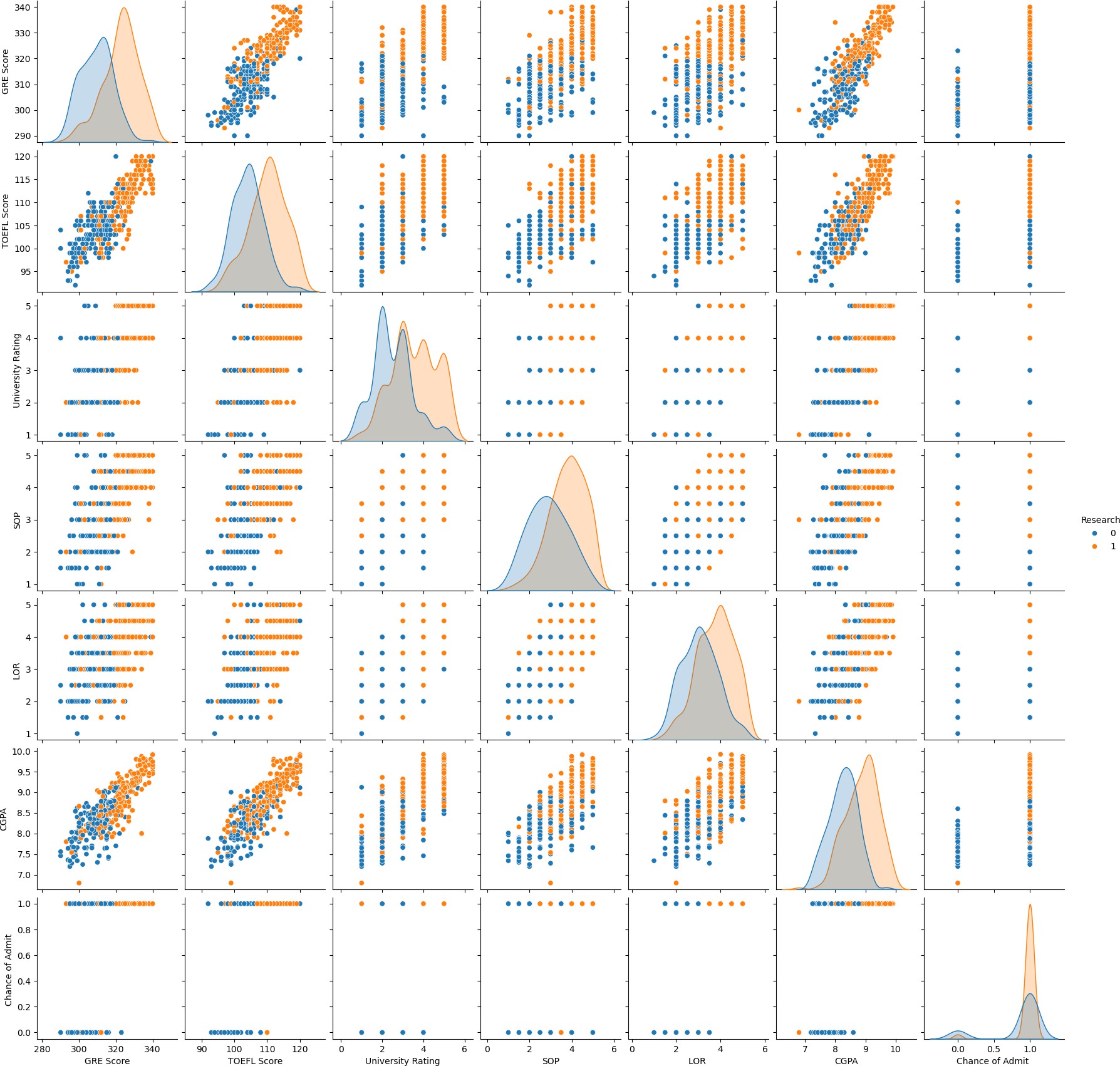
print("minimum GRE Score : ",data['GRE Score']**.**min())

maximum GRE Score : 340 minimum GRE Score : 290

In [44]:

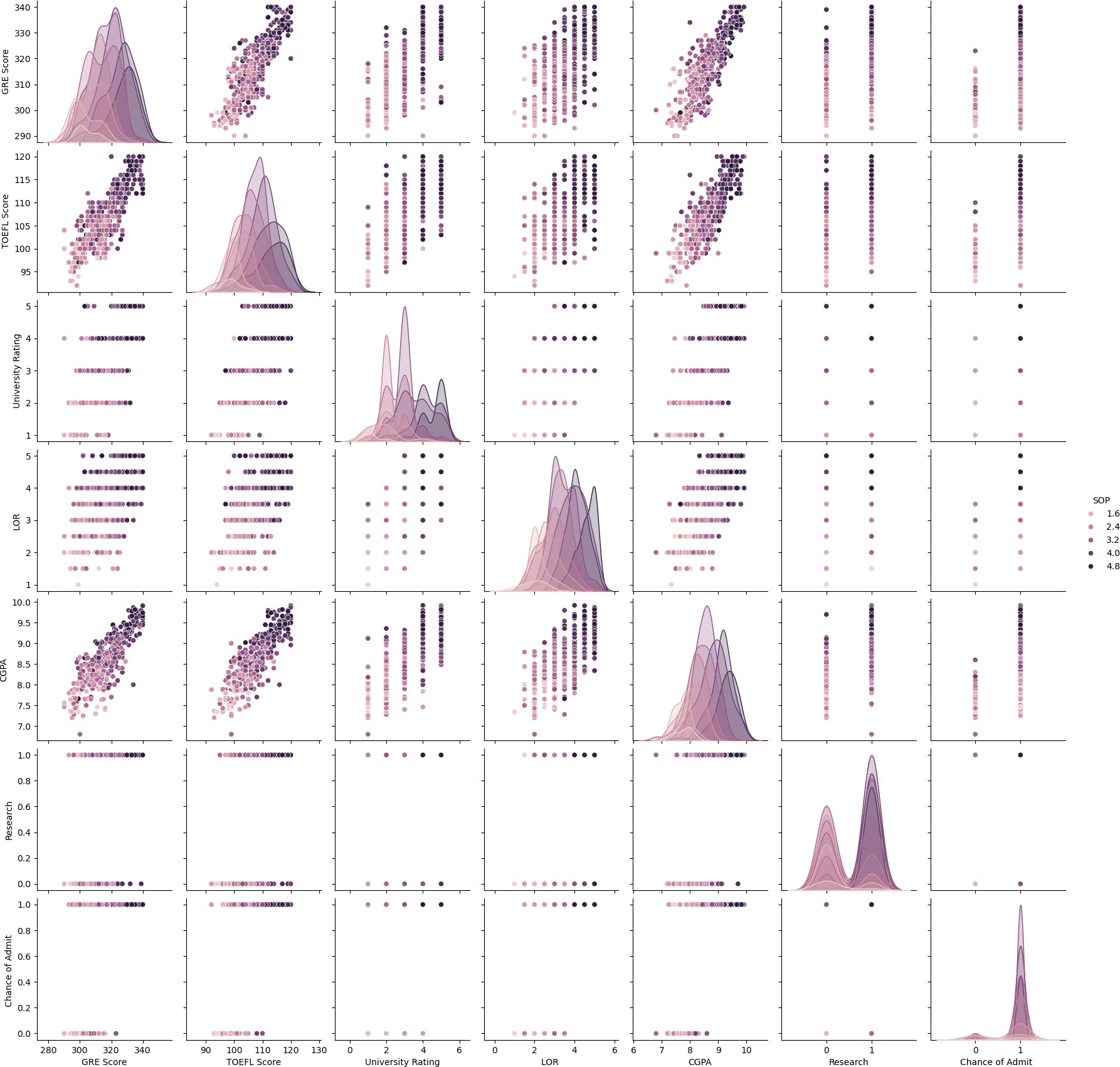
sns**.**pairplot(data,hue **=** "Research")

Out[44]: <seaborn.axisgrid.PairGrid at 0x24e2de89940>



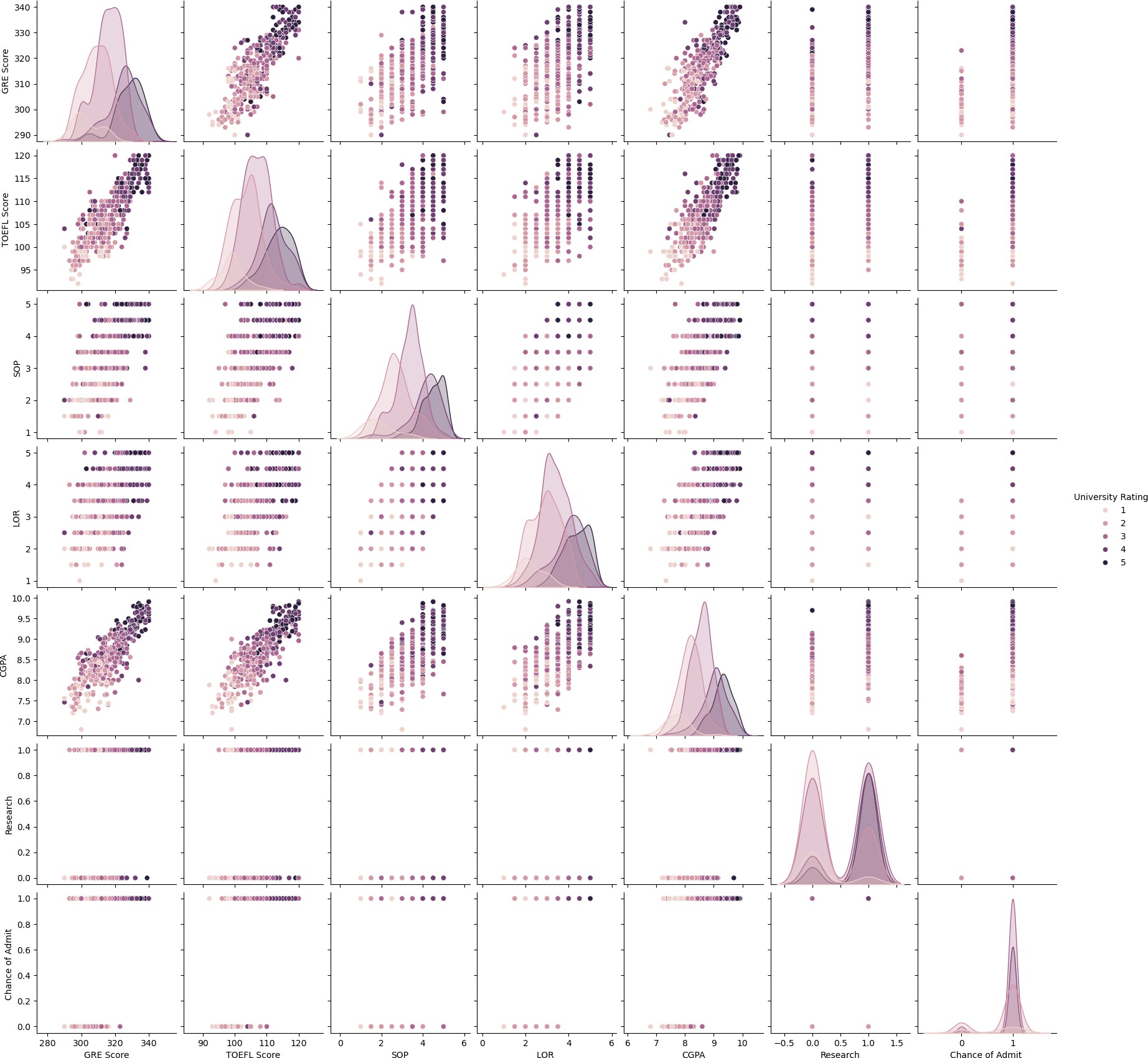
In [46]:

sns**.**pairplot(data,hue **=** "SOP");



In [48]:

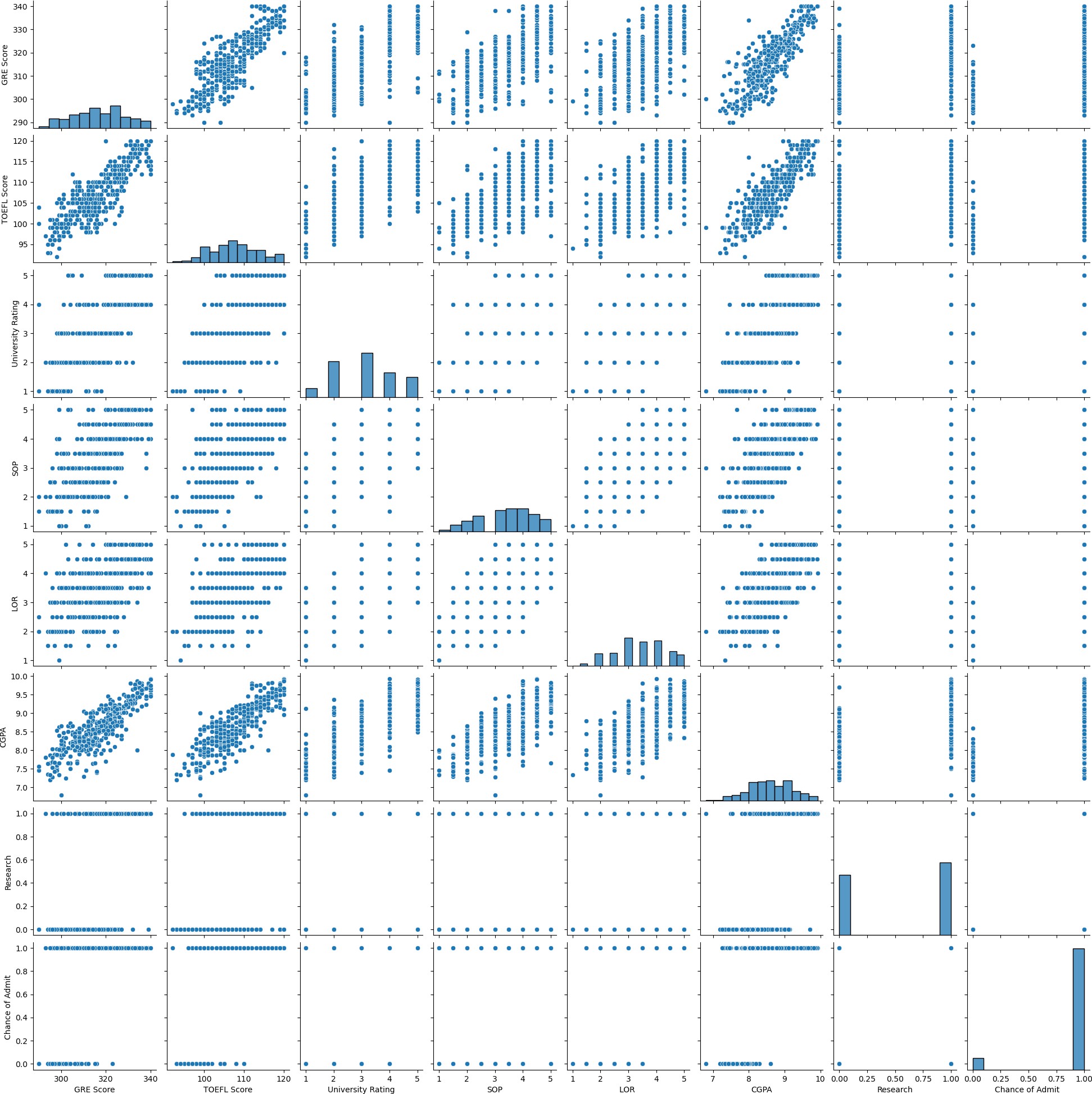
sns**.**pairplot(data,hue **=** "University Rating");



In [50]:

sns**.**pairplot(data)

Out[50]: <seaborn.axisgrid.PairGrid at 0x24e375efe00>



In [52]:

X**=** data**.**drop("Chance of Admit ",axis **=**1 ) y**=** data["Chance of Admit "]

In [54]:

X**.**nunique()

|  |  |  |
| --- | --- | --- |
| Out[54]: GRE Score |  | 49 |
| TOEFL Score |  | 29 |
| University | Rating | 5 |
| SOP |  | 9 |
| LOR |  | 9 |
| CGPA |  | 168 |
| Research |  | 2 |

dtype: int64

In [56]:

**from** sklearn.model\_selection **import** train\_test\_split

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X, y,test\_size **=** 0.2, random

*# Shape of train Test Split*

print(X\_train**.**shape,y\_train**.**shape) print(X\_test**.**shape,y\_test**.**shape)

(320, 7) (320,)

(80, 7) (80,)

In [58]:

**from** sklearn.tree **import** DecisionTreeClassifier

*# instantiate the model*

tree **=** DecisionTreeClassifier()

*# fit the model*

tree**.**fit(X\_train, y\_train)

Out[58]:



DecisionTreeClassifier()

[?](https://scikit-learn.org/1.4/modules/generated/sklearn.tree.DecisionTreeClassifier.html)

lassifier i

▾ DecisionTreeC

In [60]:

y\_train\_tree **=** tree**.**predict(X\_train) y\_test\_tree **=** tree**.**predict(X\_test)

In [62]:

**from** sklearn.metrics **import** accuracy\_score

*#computing the accuracy of the model performance*

acc\_train\_tree **=** accuracy\_score(y\_train,y\_train\_tree) acc\_test\_tree **=** accuracy\_score(y\_test,y\_test\_tree)

print("Decision Tree : Accuracy on training Data: {:.3f}"**.**format(acc\_train\_tree) print("Decision Tree : Accuracy on test Data: {:.3f}"**.**format(acc\_test\_tree))

Decision Tree : Accuracy on training Data: 1.000 Decision Tree : Accuracy on test Data: 0.863

In [64]:

**from** sklearn.metrics **import** classification\_report

*#computing the classification report of the model*

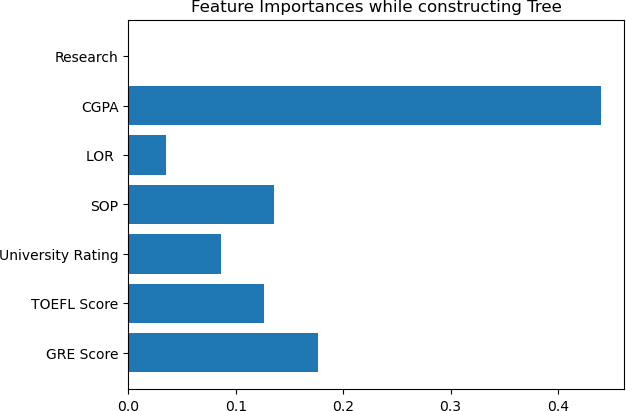
print(classification\_report(y\_test, y\_test\_tree))

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| 0 | 0.44 | 0.40 | 0.42 | 10 |
| 1 | 0.92 | 0.93 | 0.92 | 70 |
| accuracy |  |  | 0.86 | 80 |
| macro avg | 0.68 | 0.66 | 0.67 | 80 |
| weighted avg | 0.86 | 0.86 | 0.86 | 80 |

In [66]:

plt**.**barh(X**.**columns,tree**.**feature\_importances\_)

plt**.**title("Feature Importances while constructing Tree") plt**.**show()



In [68]:

*#visualization of Confusion Matrix*

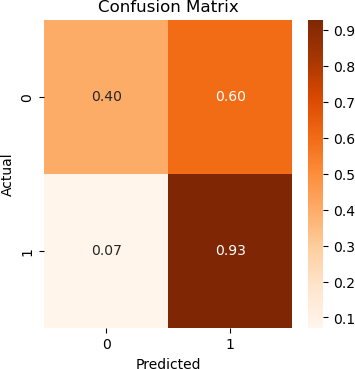
**from** sklearn.metrics **import** confusion\_matrix cm**=**confusion\_matrix(y\_test,y\_test\_tree)

cmn **=** cm**.**astype('float') **/** cm**.**sum(axis**=**1)[:, np**.**newaxis] fig, ax **=** plt**.**subplots(figsize**=**(4,4))

sns**.**heatmap(cmn, annot**=True**, fmt**=**'.2f',cmap**=**'Oranges') plt**.**title("Confusion Matrix")

plt**.**ylabel('Actual')

plt**.**xlabel('Predicted') plt**.**show(block**=False**);



In [70]:

training\_accuracy **=** [] test\_accuracy **=** []

*# try max\_depth from 1 to 15*

depth **=** range(1,16)

**for** n **in** depth:

tree\_test **=** DecisionTreeClassifier(max\_depth**=**n) tree\_test**.**fit(X\_train, y\_train)

*# record training set accuracy*

training\_accuracy**.**append(tree\_test**.**score(X\_train, y\_train))

*# record generalization accuracy*

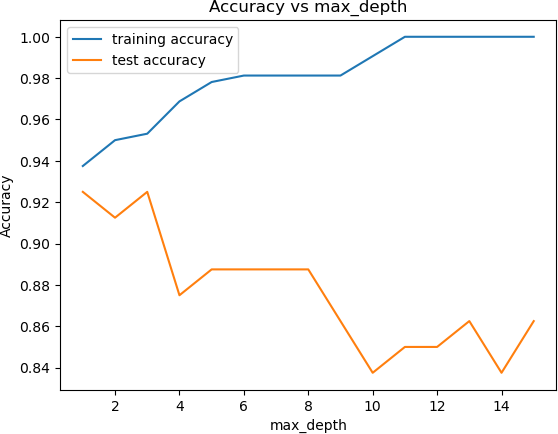
test\_accuracy**.**append(tree\_test**.**score(X\_test, y\_test))

*#plotting the training & testing accuracy for max\_depth from 1 to 15*

plt**.**plot(depth, training\_accuracy, label**=**"training accuracy") plt**.**plot(depth, test\_accuracy, label**=**"test accuracy")

plt**.**title("Accuracy vs max\_depth") plt**.**ylabel("Accuracy")

plt**.**xlabel("max\_depth") plt**.**legend();



In [72]:

**from** sklearn.tree **import** export\_text

**from** sklearn.tree **import** DecisionTreeClassifier

*# instantiate the model*

tree **=** DecisionTreeClassifier(max\_depth**=**3)

*# fit the model*

tree**.**fit(X\_train, y\_train)

text\_representation **=** export\_text(tree) print(text\_representation)

|--- feature\_5 <= 7.66

| |--- feature\_0 <= 301.00

| | |--- feature\_4 <= 2.75

| | | |--- class: 0

| | |--- feature\_4 > 2.75

| | | |--- class: 0

| |--- feature\_0 > 301.00

| | |--- feature\_0 <= 315.00

| | | |--- class: 1

| | |--- feature\_0 > 315.00

| | | |--- class: 0

|--- feature\_5 > 7.66

| |--- feature\_5 <= 8.20

| | |--- feature\_5 <= 8.19

| | | |--- class: 1

| | |--- feature\_5 > 8.19

| | | |--- class: 1

| |--- feature\_5 > 8.20

| | |--- feature\_4 <= 2.25

| | | |--- class: 1

| | |--- feature\_4 > 2.25

| | | |--- class: 1

In [74]:

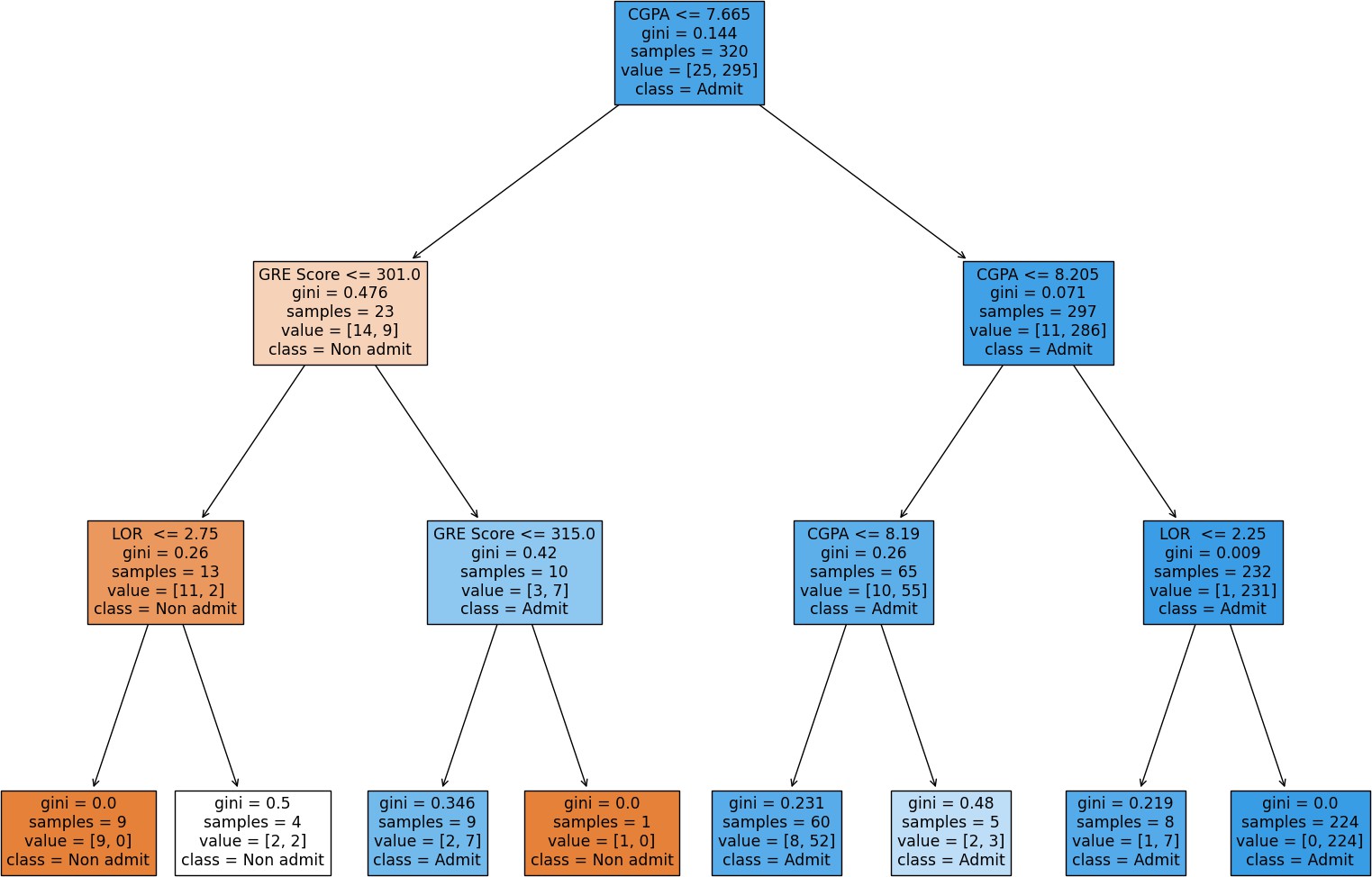
**import** sklearn.tree **as** tr

fig **=** plt**.**figure(figsize**=**(20,15))

\_ **=** tr**.**plot\_tree(tree,

feature\_names**=**X**.**columns,

class\_names**=**np**.**array(["Non admit","Admit"]), filled**=True**)



In [ ]: