In [63]:
```
import pandas as pd
import numpy as np
```

In [42]:
```
df=pd.read_csv("Admission_Predict_Ver1.1.csv")
col_names=df.columns.tolist()
print("Column names:")
print(col_names)
print("\nSample Data:")
print(df.head())
```

```
Column names:
['Serial No.', 'GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR ', 'CGP
A', 'Research', 'Chance of Admit ']

Sample Data:
   Serial No.  GRE Score  TOEFL Score  University Rating  SOP  LOR   CGPA  \
0           1        337          118                  4  4.5   4.5  9.65
1           2        324          107                  4  4.0   4.5  8.87
2           3        316          104                  3  3.0   3.5  8.00
3           4        322          110                  3  3.5   2.5  8.67
4           5        314          103                  2  2.0   3.0  8.21

   Research  Chance of Admit
0         1              0.92
1         1              0.76
2         1              0.72
3         1              0.80
4         0              0.65
```

In [43]:
```
df=df.rename(columns={'Serial No.':'no','GRE Score':'gre','TOEFL Score':'toefl','Unive
                      'CGPA':'gpa','Research':'research','Chance of Admit ':'char
```

In [44]:
```
df.dtypes
```

Out[44]:
```
no             int64
gre            int64
toefl          int64
rating         int64
sop          float64
lor          float64
gpa          float64
research       int64
chance       float64
dtype: object
```

In [45]:
```
df.shape
```

Out[45]:
```
(500, 9)
```

In [46]:
```
df.isnull().sum()
```

Out[46]:
```
no          0
gre         0
toefl       0
rating      0
sop         0
lor         0
gpa         0
research    0
chance      0
dtype: int64
```

In [47]:
```
df.describe()
```

Out[47]:

	no	gre	toefl	rating	sop	lor	gpa	research
count	500.000000	500.000000	500.000000	500.000000	500.000000	500.00000	500.000000	500.000000
mean	250.500000	316.472000	107.192000	3.114000	3.374000	3.48400	8.576440	0.560000
std	144.481833	11.295148	6.081868	1.143512	0.991004	0.92545	0.604813	0.496884
min	1.000000	290.000000	92.000000	1.000000	1.000000	1.00000	6.800000	0.000000
25%	125.750000	308.000000	103.000000	2.000000	2.500000	3.00000	8.127500	0.000000
50%	250.500000	317.000000	107.000000	3.000000	3.500000	3.50000	8.560000	1.000000
75%	375.250000	325.000000	112.000000	4.000000	4.000000	4.00000	9.040000	1.000000
max	500.000000	340.000000	120.000000	5.000000	5.000000	5.00000	9.920000	1.000000

In [48]:
```
df.groupby('rating').mean()
```

Out[48]:

rating	no	gre	toefl	sop	lor	gpa	research	chance
1	281.558824	304.911765	100.205882	1.941176	2.426471	7.798529	0.294118	0.562059
2	249.555556	309.134921	103.444444	2.682540	2.956349	8.177778	0.293651	0.626111
3	247.574074	315.030864	106.314815	3.308642	3.401235	8.500123	0.537037	0.702901
4	275.809524	323.304762	110.961905	4.000000	3.947619	8.936667	0.780952	0.801619
5	207.753425	327.890411	113.438356	4.479452	4.404110	9.278082	0.876712	0.888082

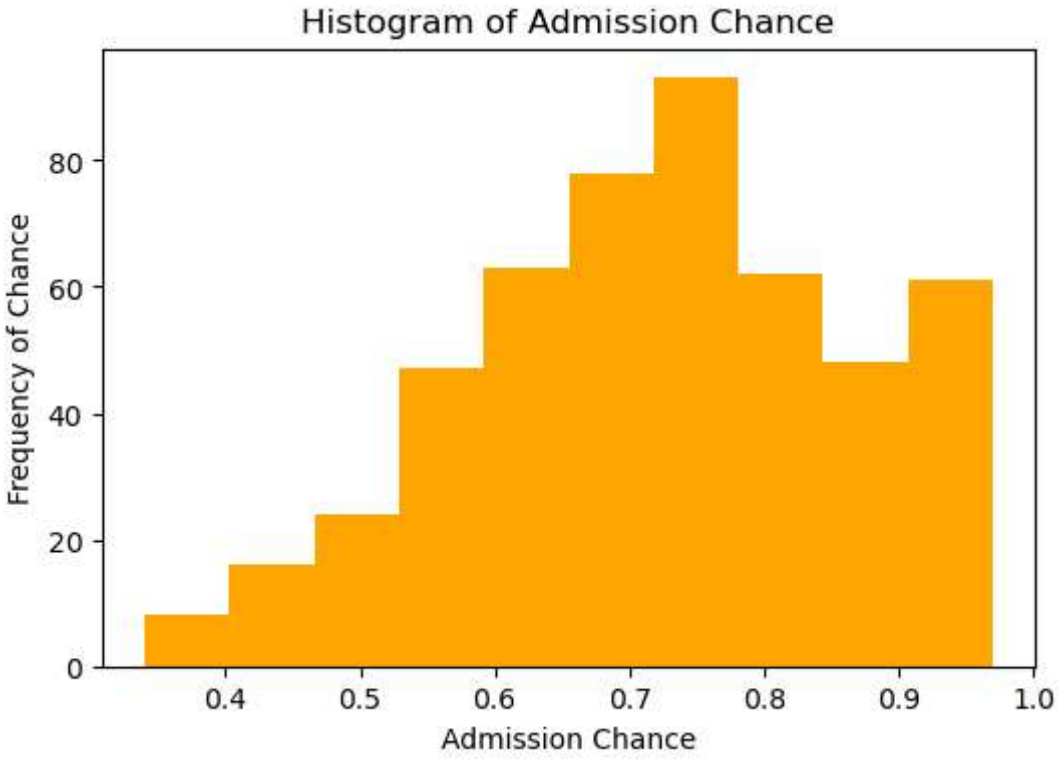
In [49]:
```
df[df['chance']>0.82].groupby('chance').mean()
```

Out[49]:

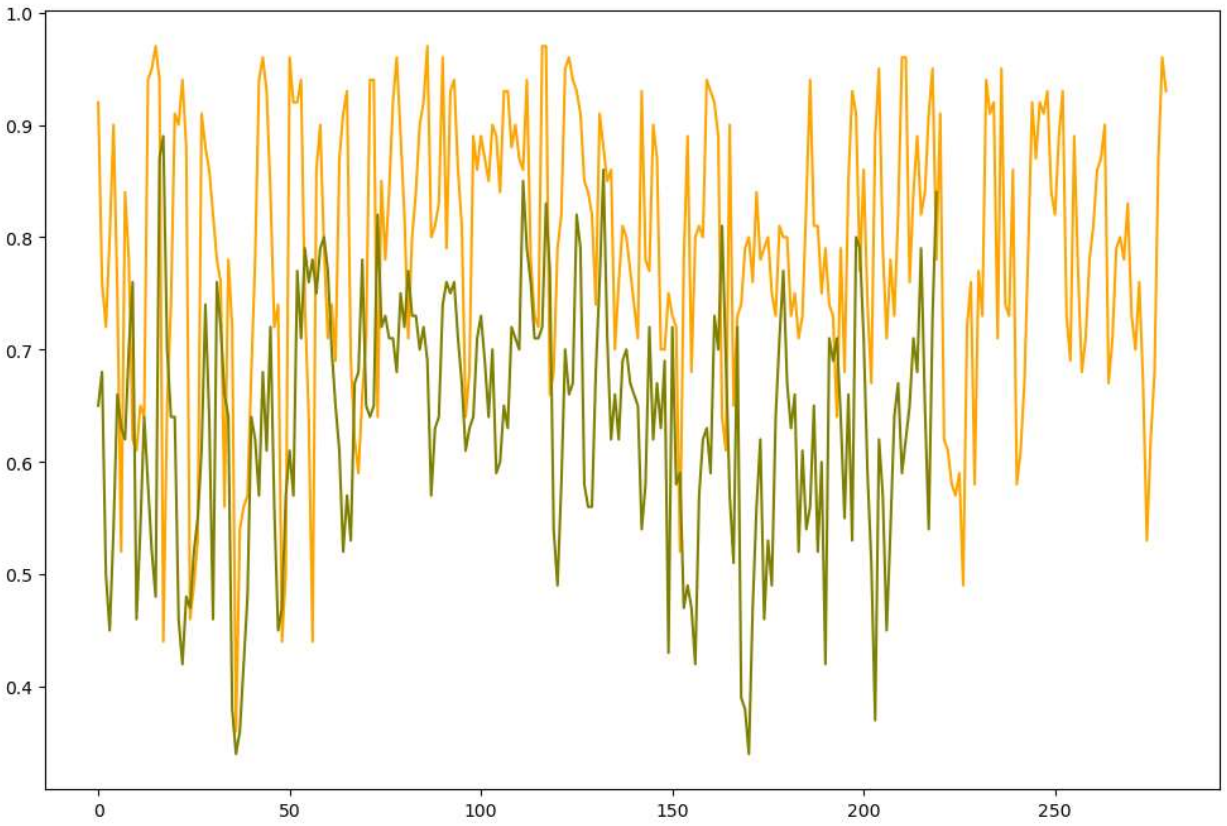
chance	no	gre	toefl	rating	sop	lor	gpa	research
0.83	309.000000	326.500000	112.750000	3.750000	3.875000	3.750000	9.032500	0.750000
0.84	255.636364	323.909091	109.636364	3.454545	3.818182	3.772727	9.032727	0.909091
0.85	229.000000	322.000000	111.500000	3.666667	4.083333	4.166667	9.041667	0.833333
0.86	246.900000	325.400000	114.400000	4.200000	4.300000	4.300000	9.124000	0.900000
0.87	273.750000	325.625000	111.125000	4.625000	4.375000	4.187500	9.101250	0.875000
0.88	127.000000	323.000000	110.750000	5.000000	4.875000	4.500000	9.152500	1.000000
0.89	269.000000	328.636364	113.545455	4.363636	4.318182	4.136364	9.270909	0.909091
0.90	186.777778	330.555556	116.111111	4.000000	4.500000	4.111111	9.324444	1.000000
0.91	267.400000	330.500000	115.000000	4.500000	4.250000	4.450000	9.328000	1.000000
0.92	226.888889	328.555556	114.888889	4.777778	4.388889	4.500000	9.417778	1.000000
0.93	268.833333	330.583333	115.916667	4.583333	4.583333	4.250000	9.477500	1.000000
0.94	160.846154	334.230769	116.692308	4.846154	4.692308	4.846154	9.533077	1.000000
0.95	288.000000	336.200000	118.000000	4.400000	4.900000	4.400000	9.540000	1.000000
0.96	239.500000	337.375000	116.000000	4.625000	4.750000	4.687500	9.753750	1.000000
0.97	144.000000	337.500000	119.750000	4.750000	4.250000	4.250000	9.875000	1.000000

In [50]:

```
import matplotlib.pyplot as plt
plt.figure(figsize=(6,4))
plt.hist(df['chance'],bins=10,color="orange")
plt.title('Histogram of Admission Chance')
plt.xlabel('Admission Chance')
plt.ylabel('Frequency of Chance')
plt.show()
```

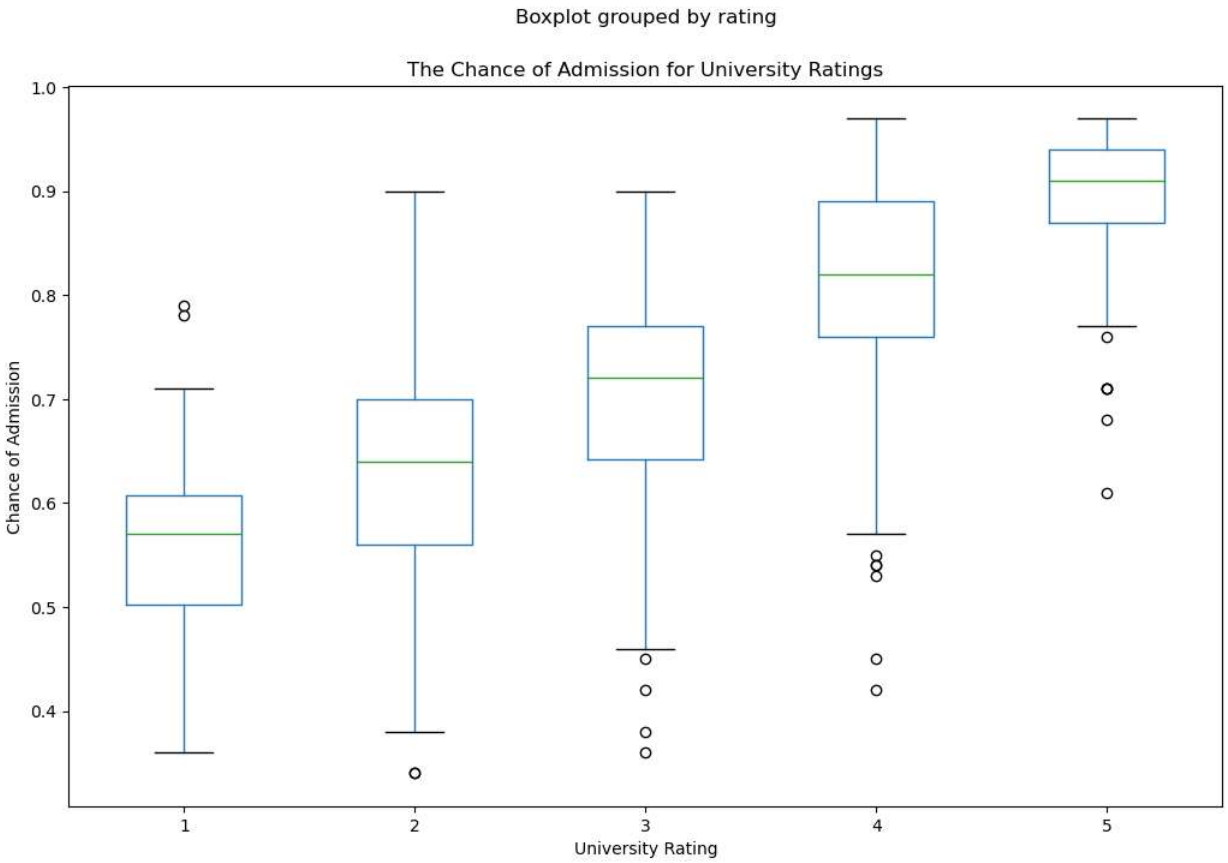


In [51]:
```
plt.figure(figsize=(12,8))
plt.plot(range(len(df[df['research']==1])), df[df['research']==1]['chance'], color='or
plt.plot(range(len(df[df['research']==0])), df[df['research']==0]['chance'], color='ol
plt.show()
```

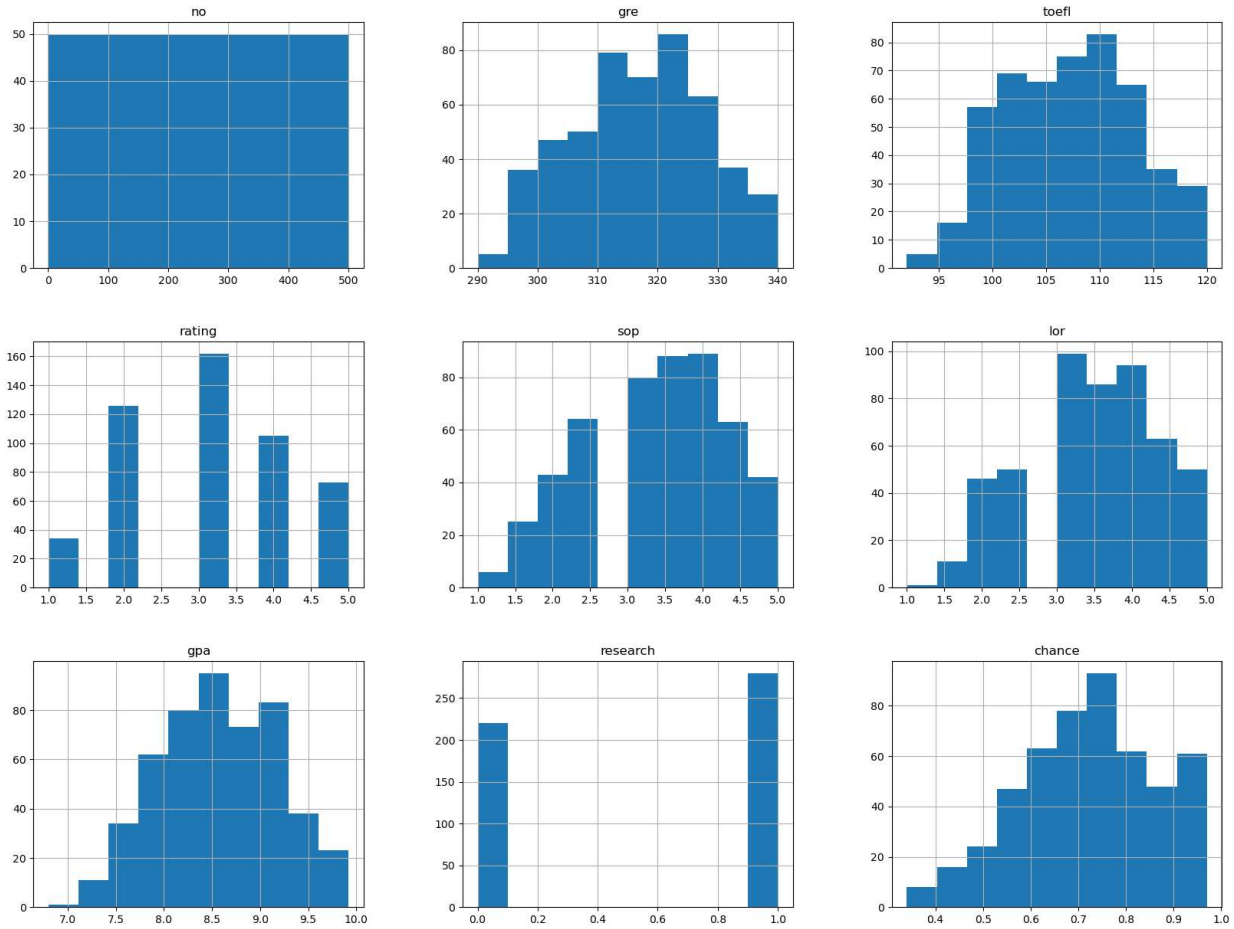


In [52]:
```
df.boxplot(column='chance',by='rating',grid=False,figsize=(12,8))
plt.title('The Chance of Admission for University Ratings')
plt.xlabel('University Rating')
```

```
plt.ylabel('Chance of Admission')
plt.show()
```



In [53]:
```
df.hist(bins=10, figsize=(20,15))
plt.show()
```



In [62]:
```
df.dtypes
```

Out[62]:
```
gre           int64
toefl         int64
rating        int64
sop         float64
lor         float64
gpa         float64
research      int64
chance      float64
dtype: object
```

In [69]:
```
df.drop(['gre'],axis=1,inplace=True)
var=df.columns.values.tolist()
y=df['chance']
x=[i for i in var if i not in ['chance']]
x=df[x]
```

In [70]:
```
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_state=0)
```

In [71]:
```
from sklearn.preprocessing import MinMaxScaler
xs=MinMaxScaler()
x_train[x_train.columns] = xs.fit_transform(x_train[x_train.columns])
x_test[x_test.columns] = xs.transform(x_test[x_test.columns])
```

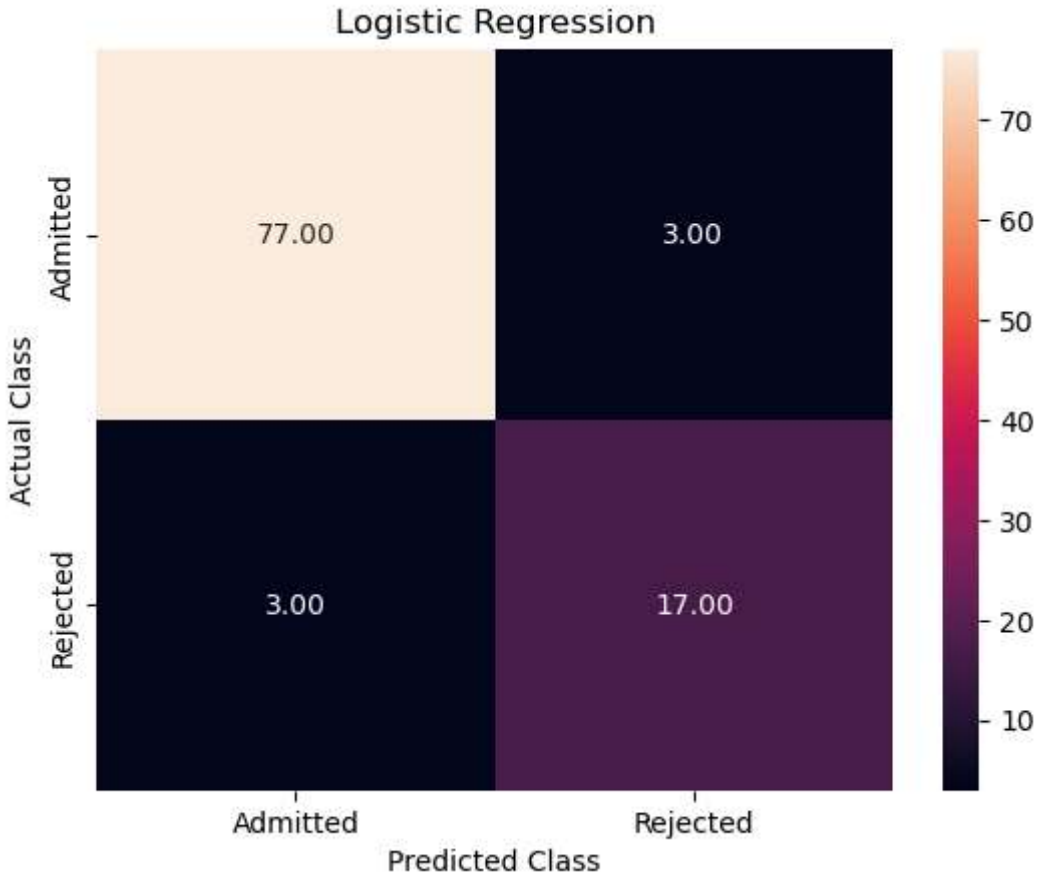
In [72]:
```
import numpy as np
cy_train=[1 if chance > 0.83 else 0 for chance in y_train]
cy_train=np.array(cy_train)
```

```
cy_test=[1 if chance > 0.83 else 0 for chance in y_test]
cy_test=np.array(cy_test)
```

In [76]:

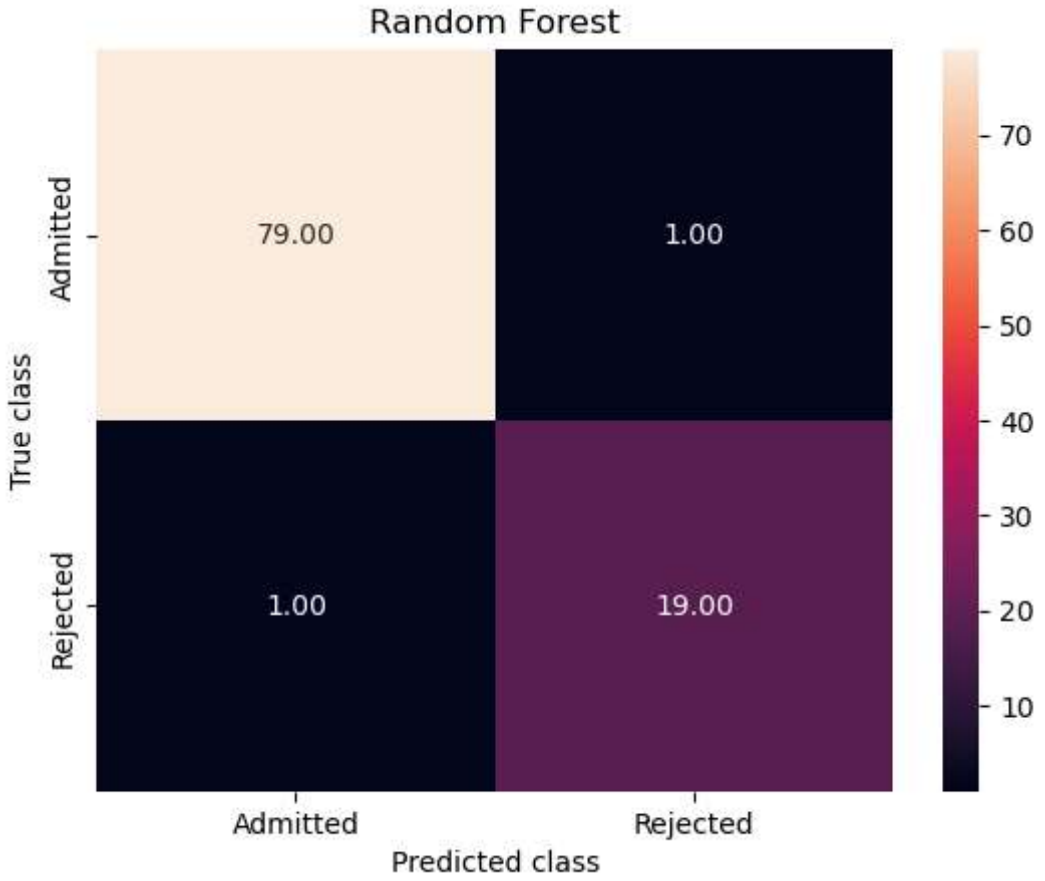
```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(x_train, cy_train)


# Printing accuracy score & confusion matrix
from sklearn.metrics import accuracy_score
print('Logistic regression accuracy: {:.3f}'.format(accuracy_score(cy_test, lr.predict
print('--------------------------------------')
from sklearn.metrics import classification_report
print(classification_report(cy_test, lr.predict(x_test)))

cy = lr.predict(x_test)
from sklearn.metrics import confusion_matrix
import seaborn as sns
lr_confm = confusion_matrix(cy, cy_test)
sns.heatmap(lr_confm, annot=True, fmt='.2f',xticklabels = ["Admitted", "Rejected"] , y
plt.ylabel('Actual Class')
plt.xlabel('Predicted Class')
plt.title('Logistic Regression')
plt.show()
```

```
Logistic regression accuracy: 0.940
--------------------------------------
              precision    recall  f1-score   support

           0       0.96      0.96      0.96        80
           1       0.85      0.85      0.85        20

    accuracy                           0.94       100
   macro avg       0.91      0.91      0.91       100
weighted avg       0.94      0.94      0.94       100
```



In [78]:
```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf.fit(x_train, cy_train)

# Printing accuracy score & confusion matrix
print('Random Forest Accuracy: {:.3f}'.format(accuracy_score(cy_test, rf.predict(x_tes
print('-------------------------------------')
print(classification_report(cy_test, rf.predict(x_test)))

cy = rf.predict(x_test)
rf_confm = confusion_matrix(cy, cy_test)
sns.heatmap(rf_confm, annot=True, fmt='.2f',xticklabels = ["Admitted", "Rejected"] , y
plt.ylabel('True class')
plt.xlabel('Predicted class')
plt.title('Random Forest')
plt.show()
```

```
Random Forest Accuracy: 0.980
-------------------------------------
              precision    recall  f1-score   support

           0       0.99      0.99      0.99        80
           1       0.95      0.95      0.95        20

    accuracy                           0.98       100
   macro avg       0.97      0.97      0.97       100
weighted avg       0.98      0.98      0.98       100
```



In [79]:
```
f_imp=pd.Series(rf.feature_importances_,index=x_train.columns).sort_values(ascending=F
print(f_imp)
```

```
gpa         0.562055
toefl       0.214938
rating      0.118052
lor         0.082683
research    0.022272
dtype: float64
```