In [1]:
```
# Allows multiple outputs from a single cell:
from IPython.core.interactiveshell import InteractiveShell as IS; IS.ast_node_
interactivity = "all"
!pip -q install -U statsmodels > log.txt   # ensures no FutureWarnings from st
atsmodels

import pandas as pd, numpy as np, statsmodels.api as sm, pprint, math, seaborn
as sns, matplotlib.pyplot as plt, sklearn as sk
from scipy import stats as stat

from math import floor
from termcolor import colored

from sklearn.datasets import make_classification
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split as tts
from sklearn.metrics import r2_score, roc_auc_score, roc_curve, auc, confusion
_matrix
from datetime import datetime as dt

%matplotlib inline
```

In [2]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [3]:
```
df = pd.read_csv('/content/drive/MyDrive/ColabNotebooks/Data/ncaa_model.csv')
```

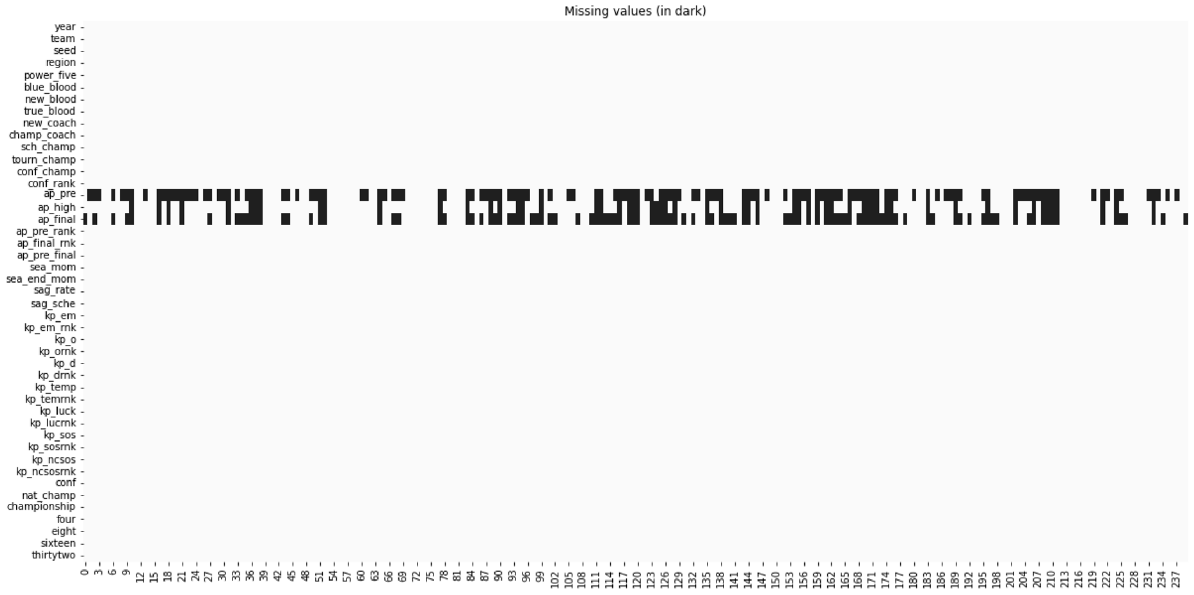
In [4]:
```
df
```

Out[4]:

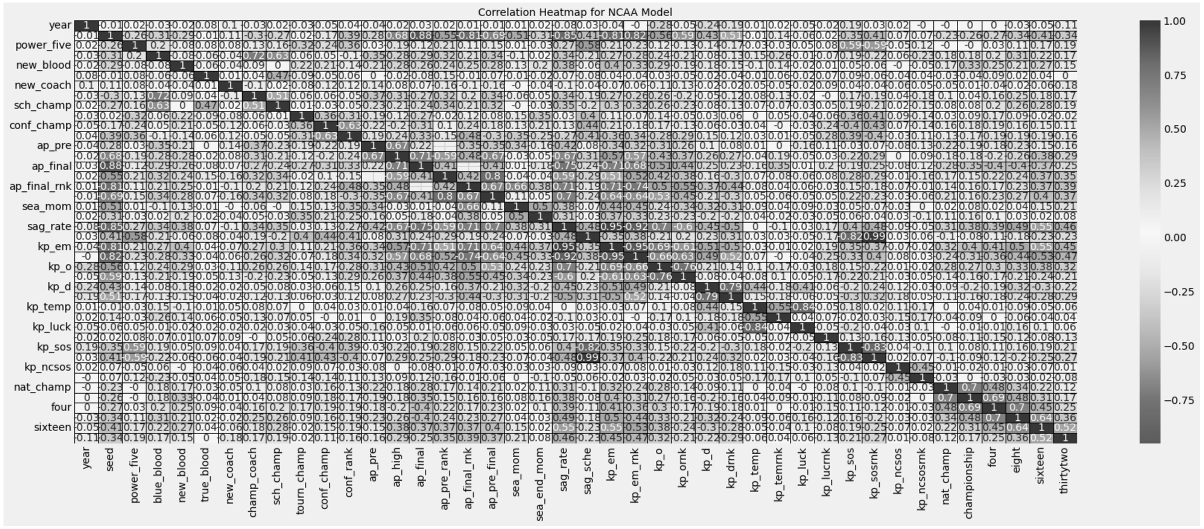
	year	team	seed	region	power_five	blue_blood	new_blood	true_blood	new_coach
0	2022	Alabama	5	west	1	0	0	0	0
1	2021	Alabama	2	east	1	0	0	0	0
2	2018	Alabama	9	east	1	0	0	0	0
3	2022	Arizona	1	south	1	0	0	0	1
4	2018	Arizona	4	south	1	0	0	0	0
...	...	...	...	...	...	...	...	...	...
235	2019	Wisconsin	5	south	1	0	0	0	0
236	2017	Wisconsin	8	east	1	0	0	0	0
237	2019	Wofford	7	midwest	0	0	0	0	0
238	2018	Xavier	1	west	0	0	0	0	0
239	2017	Xavier	11	west	0	0	0	0	0

240 rows × 45 columns

In [5]:
```
plt.rcParams['figure.figsize'] = [20, 10]
ax = sns.heatmap(df.isnull().T, cmap = "BuPu", cbar=False);
ax.set_title('Missing values (in dark)');
```



In [6]:
```
plt.style.use('fivethirtyeight')
fig, ax = plt.subplots(figsize=(25,10))
sns.heatmap(df.corr().round(2), annot=True, cmap='Spectral_r', ax=ax, linecolo
r= 'black', linewidth = 1.0)
ax.set_title(f'Correlation Heatmap for NCAA Model', color = 'black', fontsize=
14)
plt.tight_layout();
```



In [7]:
```
# Loading the data for the 2023 NCAA teams

df_new = pd.read_csv('/content/drive/MyDrive/ColabNotebooks/Data/ncaa_2023.cs
v')
df_new_team = pd.read_csv('/content/drive/MyDrive/ColabNotebooks/Data/ncaa_202
3.csv')
df_new.drop(columns = ['year','team', 'region','ap_pre','ap_high','ap_final'],
inplace=True)
df_new['const'] = 1
```

In [8]:
```
# Creating a new dataframe before calculating the second round teams
df_32 = df.copy()

# Dropping Columns that won't be used in the modeling

df_32.drop(columns = ['team','region','conf','year','sixteen','eight','fou
r','championship','nat_champ','ap_pre','ap_high','ap_final'], inplace=True)

# Adding a constant to the model

df_32['const'] = 1
```

In [9]:
```
# Creating model

tX0, vX0, tY0, vY0 = tts(df_32.drop(['thirtytwo'], axis=1), df_32['thirtytw
o'], test_size = 0.2, random_state=123)
```

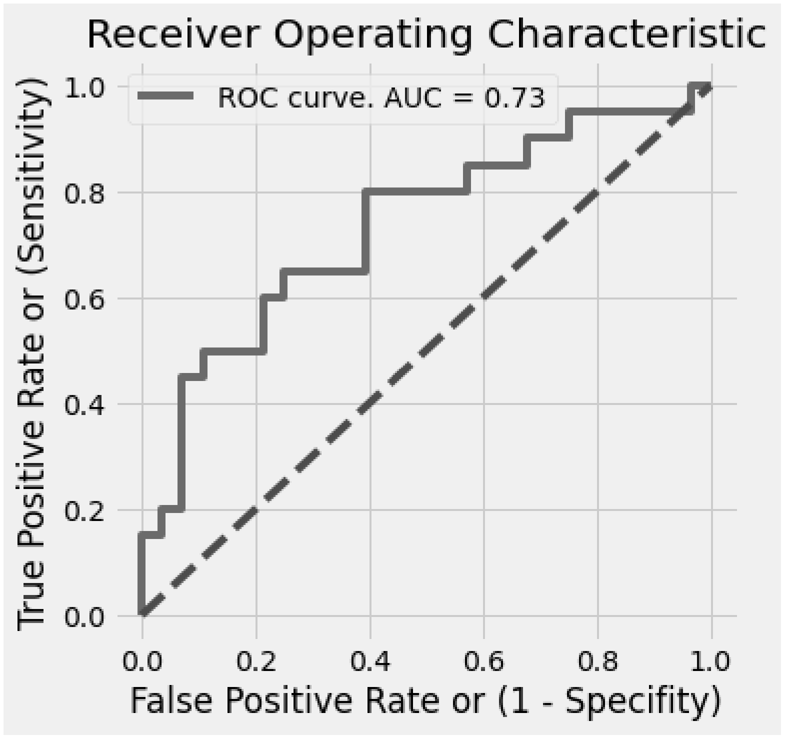
In [10]:
```
# Logistic regression model summary

md0 = sm.Logit(tY0, tX0).fit()
print(md0.summary(title='NCAA Model - Second Round', alpha=.05))
```

```
Optimization terminated successfully.
         Current function value: 0.316814
         Iterations 9
                          NCAA Model - Second Round                         
=============================================================================
=
Dep. Variable:             thirtytwo   No. Observations:                   19
2
Model:                         Logit   Df Residuals:                       15
9
Method:                          MLE   Df Model:                            3
2
Date:               Wed, 22 Mar 2023   Pseudo R-squ.:                   0.514
9
Time:                       19:16:06   Log-Likelihood:                 -60.82
8
converged:                      True   LL-Null:                        -125.3
9
Covariance Type:           nonrobust   LLR p-value:                  1.281e-1
3
=============================================================================
===
                   coef    std err          z      P>|z|      [0.025      0.9
75]
-----------------------------------------------------------------------------
---
seed             0.7998      0.227      3.518      0.000       0.354       1.
245
power_five       1.2825      0.755      1.699      0.089      -0.197       2.
762
blue_blood       1.2639      3.057      0.413      0.679      -4.728       7.
256
new_blood        0.0909      1.543      0.059      0.953      -2.933       3.
115
true_blood       0.9056      1.873      0.484      0.629      -2.765       4.
576
new_coach       -2.5555      0.897     -2.849      0.004      -4.314      -0.
798
champ_coach     -0.2074      0.852     -0.243      0.808      -1.878       1.
463
sch_champ       -0.0309      0.338     -0.091      0.927      -0.693       0.
632
tourn_champ      0.5772      0.838      0.689      0.491      -1.065       2.
219
conf_champ      -0.3039      0.806     -0.377      0.706      -1.883       1.
275
conf_rank       -0.0531      0.188     -0.281      0.778      -0.422       0.
316
ap_pre_rank      0.8498      0.977      0.870      0.384      -1.065       2.
765
ap_final_rnk     1.7850      1.547      1.154      0.248      -1.247       4.
817
ap_pre_final    -0.4624      1.472     -0.314      0.753      -3.347       2.
423
sea_mom          0.7950      1.277      0.622      0.534      -1.709       3.
299
sea_end_mom     -2.1322      1.176     -1.813      0.070      -4.437       0.
```

```
173
sag_rate         0.7044       0.399      1.768      0.077      -0.077       1.
485
sag_sche         0.1296       0.046      2.843      0.004       0.040       0.
219
kp_em           -1.2372       2.495     -0.496      0.620      -6.127       3.
652
kp_em_rnk       -0.1228       0.097     -1.261      0.207      -0.314       0.
068
kp_o             1.3049       2.441      0.534      0.593      -3.480       6.
090
kp_ornk          0.0306       0.014      2.135      0.033       0.003       0.
059
kp_d            -1.1026       2.435     -0.453      0.651      -5.876       3.
671
kp_drnk         -0.0096       0.016     -0.582      0.560      -0.042       0.
023
kp_temp          0.3050       0.324      0.941      0.347      -0.331       0.
941
kp_temrnk        0.0108       0.009      1.223      0.221      -0.006       0.
028
kp_luck         -0.3019       5.095     -0.059      0.953     -10.288       9.
684
kp_lucrnk       -0.0165       0.006     -2.944      0.003      -0.028      -0.
006
kp_sos           0.0262       0.141      0.186      0.852      -0.249       0.
302
kp_sosrnk       -0.1293       0.046     -2.824      0.005      -0.219      -0.
040
kp_ncsos        -0.0046       0.019     -0.248      0.804      -0.041       0.
032
kp_ncsosrnk     -0.0036       0.003     -1.140      0.254      -0.010       0.
003
const         -104.4071      41.464     -2.518      0.012    -185.675     -23.
139
==============================================================================
===
```

In [11]:
```
# Creating prediction probabilities and labels

pY_prob0 = md0.predict(vX0)
pY_prob0 = pY_prob0
pY0 = (pY_prob0 > 0.9) * 1
AUC = roc_auc_score(vY0, pY_prob0)

# Creating confusion matrix

dfCM = pd.DataFrame(confusion_matrix(vY0, pY0), index=['True-','True+'], colum
ns=['Pred-','Pred+'])
print(f'Confusion matrix:\n{dfCM}')
print(f'Out of sample accuracy: {np.mean(pY0 == vY0):.2f} and AUC:{AUC:.2f}')

# Creating ROC & AUC plot

fpr, tpr, thresholds = roc_curve(vY0, pY_prob0)

plt.rcParams['figure.figsize'] = [5, 5]
ax = pd.DataFrame([fpr, tpr], index=['fpr','tpr']).T.plot(
    'fpr','tpr', kind='line', grid=True, title='Receiver Operating Characteris
tic', label=f'ROC curve. AUC = {AUC:.2f}');

ax.plot([0, 1], [0, 1], 'r--');  # random predictions curve
ax.set_ylabel('True Positive Rate or (Sensitivity)');
ax.set_xlabel('False Positive Rate or (1 - Specifity)');
```

```
Confusion matrix:
       Pred-  Pred+
True-     25      3
True+     10     10
Out of sample accuracy: 0.73 and AUC:0.73
```



In [12]:
```
# Fitting the model to the 2023 NCAA Teams

pY_prob0 = md0.predict(df_new)
pY_prob0 = pY_prob0
pY0 = (pY_prob0 > 0.9) * 1
```

In [13]:
```
# Printing 2023 NCAA teams prediction probabilites and prediction labels for t
he second round

df_prob = pd.Series(pY_prob0)
df_prob = pd.DataFrame(df_prob, columns=['prob'])
df_pred = pd.DataFrame(pY0, columns=['thirtytwo'])
prediction_results =df_prob.merge(df_pred['thirtytwo'], left_index=True, right
_index=True)
prediction_results = prediction_results.merge(df_new_team['team'], left_index=
True, right_index=True).merge(df_new_team['seed'], left_index=True, right_inde
x=True).merge(df_new_team['region'], left_index=True, right_index=True)\
.merge(df_new_team['conf_champ'], left_index=True, right_index=True).merge(df_
new_team['tourn_champ'], left_index=True, right_index=True).merge(df_new_team
['sag_rate'], left_index=True, right_index=True).merge(df_new_team['kp_em_rn
k'], left_index=True, right_index=True)\
.merge(df_new_team['kp_ornk'], left_index=True, right_index=True).merge(df_new
_team['kp_drnk'], left_index=True, right_index=True).merge(df_new_team['kp_luc
rnk'], left_index=True, right_index=True).merge(df_new_team['kp_sosrnk'], left
_index=True, right_index=True)\
.merge(df_new_team['ap_final'], left_index=True, right_index=True).merge(df_ne
w_team['new_coach'], left_index=True, right_index=True)
prediction_results.sort_values(['prob'], ascending=False)
```

Out[13]:

	prob	thirtytwo	team	seed	region	conf_champ	tourn_champ	sag_rate	kp_em
12	1.000000	1	Creighton	6	south	0	0	87.63	
9	0.999997	1	Arizona	2	south	0	1	89.86	
13	0.999924	1	San Diego St.	5	south	1	1	85.19	
1	0.999558	1	UCLA	2	west	1	0	91.21	
7	0.998562	1	Gonzaga	3	west	1	1	90.47	
2	0.996878	1	Alabama	1	south	1	1	92.33	
8	0.995489	1	Kansas	1	west	1	0	89.95	
17	0.988260	1	Utah St.	10	south	0	0	84.12	
0	0.979514	1	Houston	1	midwest	1	0	91.85	
20	0.979323	1	Duke	5	east	0	1	87.34	
14	0.967520	1	Baylor	3	south	0	0	87.36	
25	0.949100	1	Florida Atlantic	9	east	1	1	83.66	
4	0.946360	1	Tennessee	4	east	0	0	89.28	
6	0.944307	1	Purdue	1	east	1	1	89.16	
30	0.932215	1	Boise St.	10	west	0	0	81.89	
10	0.929174	1	Saint Mary's	5	west	1	0	86.43	
47	0.928974	1	Arizona St.	11	west	0	0	80.99	
34	0.918333	1	USC	10	east	0	0	84.12	
24	0.893580	0	Texas A&M	7	midwest	0	0	86.08	
29	0.865486	0	Indiana	4	midwest	0	0	86.21	
18	0.850201	0	Memphis	8	east	0	1	86.53	
5	0.845760	0	Texas	2	midwest	0	1	90.46	
44	0.836421	0	Oral Roberts	12	east	1	1	80.49	
28	0.788687	0	Auburn	9	midwest	0	0	85.32	
41	0.784477	0	Mississippi St.	11	midwest	0	0	82.26	
39	0.763661	0	Nevada	11	west	0	0	79.21	
31	0.732642	0	Michigan St.	7	east	0	0	84.93	
16	0.723841	0	West Virginia	9	south	0	0	85.86	
19	0.685933	0	Arkansas	8	west	0	0	85.76	
27	0.553009	0	Kentucky	6	east	0	0	85.95	
26	0.403795	0	TCU	6	west	0	0	85.95	
38	0.308480	0	Northwestern	7	west	0	0	83.93	
11	0.295325	0	Marquette	2	east	1	1	87.51	
45	0.266572	0	VCU	12	west	1	1	83.45	

	prob	thirtytwo	team	seed	region	conf_champ	tourn_champ	sag_rate	kp_em
22	0.265721	0	Iowa St.	6	midwest	0	0	84.82	
36	0.260228	0	Penn St.	10	midwest	0	0	83.88	
15	0.238529	0	Xavier	3	midwest	0	0	86.50	
33	0.232211	0	Virginia	4	south	1	0	85.08	
23	0.222117	0	Kansas St.	3	east	0	0	85.34	
3	0.199233	0	UConn	4	west	0	0	90.07	
53	0.146159	0	Louisiana	13	east	0	1	77.39	
21	0.069423	0	Maryland	8	south	0	0	85.47	
43	0.061475	0	NC State	11	south	0	0	83.25	
48	0.048747	0	Kent St.	13	midwest	0	1	81.07	
50	0.041810	0	Iona	13	west	1	1	79.70	
32	0.039806	0	Illinois	9	west	0	0	85.30	
42	0.025953	0	Missouri	7	south	0	0	82.38	
37	0.011489	0	Miami	5	midwest	1	0	84.18	
40	0.010919	0	Providence	11	east	0	0	84.12	
35	0.008644	0	Iowa	8	midwest	0	0	84.85	
49	0.004056	0	Charleston	12	south	1	1	80.69	
51	0.003339	0	Pitt	11	midwest	0	0	81.54	
46	0.002738	0	Drake	12	midwest	0	1	81.90	
52	0.000022	0	Furman	13	south	1	1	79.48	

In [14]:
```
# Creating a new dataframe before calculating the sweet sixteen teams

df_16 = df.copy()

# Dropping columns that won't be used in the modeling

df_16.drop(columns = ['team','region','conf','year','thirtytwo','eight','fou
r','championship','nat_champ','ap_pre','ap_high','ap_final'], inplace=True)

# Adding a constant to the model

df_16['const'] = 1
```

In [15]:
```
# Creating model

tX1, vX1, tY1, vY1 = tts(df_16.drop(['sixteen'], axis=1), df_16['sixteen'], te
st_size = 0.2, random_state=123)
```

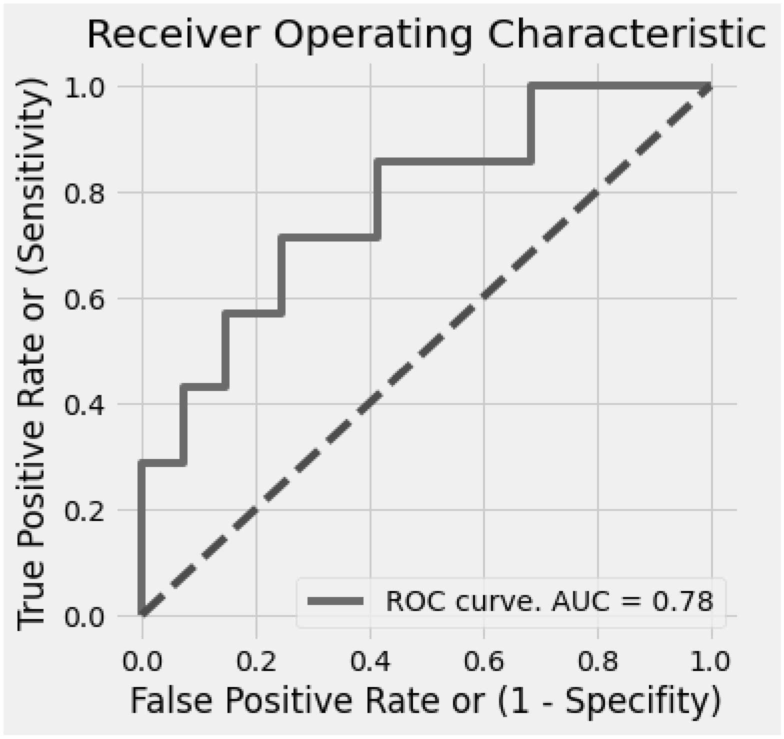
In [16]:
```
md1 = sm.Logit(tY1, tX1).fit(method = 'powell')
print(md1.summary(title='NCAA Model - Sweet Sixteen ', alpha=.05))
```

```
Optimization terminated successfully.
         Current function value: 0.326041
         Iterations: 33
         Function evaluations: 12270
                         NCAA Model - Sweet Sixteen
==============================================================================
=
Dep. Variable:                sixteen   No. Observations:                   19
2
Model:                          Logit   Df Residuals:                       15
9
Method:                           MLE   Df Model:                            3
2
Date:                Wed, 22 Mar 2023   Pseudo R-squ.:                   0.498
4
Time:                        19:16:09   Log-Likelihood:                 -62.60
0
converged:                       True   LL-Null:                        -124.8
0
Covariance Type:            nonrobust   LLR p-value:                 7.853e-1
3
==============================================================================
===
                  coef    std err          z      P>|z|      [0.025      0.9
75]
------------------------------------------------------------------------------
---
seed            0.5529      0.209      2.643      0.008       0.143       0.
963
power_five     -1.0248      0.784     -1.306      0.191      -2.562       0.
513
blue_blood     -1.8584      1.827     -1.017      0.309      -5.439       1.
722
new_blood       1.2835      1.028      1.249      0.212      -0.730       3.
297
true_blood     -1.0206      1.736     -0.588      0.557      -4.423       2.
381
new_coach      -1.4900      1.032     -1.444      0.149      -3.512       0.
532
champ_coach     0.1817      0.419      0.433      0.665      -0.640       1.
004
sch_champ       0.3344      0.226      1.477      0.140      -0.109       0.
778
tourn_champ    -0.3836      0.726     -0.529      0.597      -1.806       1.
039
conf_champ      0.3197      0.724      0.442      0.659      -1.099       1.
738
conf_rank       0.0702      0.209      0.335      0.737      -0.340       0.
481
ap_pre_rank     0.2785      1.173      0.237      0.812      -2.021       2.
578
ap_final_rnk    1.8762      1.413      1.328      0.184      -0.893       4.
646
ap_pre_final   -1.0485      1.378     -0.761      0.447      -3.749       1.
652
sea_mom         0.2563      0.835      0.307      0.759      -1.381       1.
893
```

```
sea_end_mom     -2.7130      0.969     -2.800      0.005      -4.612      -0.
814
sag_rate        -0.0018      0.397     -0.004      0.996      -0.781       0.
777
sag_sche         0.0158      0.049      0.325      0.745      -0.079       0.
111
kp_em            0.1948      3.032      0.064      0.949      -5.749       6.
138
kp_em_rnk       -0.2038      0.093     -2.190      0.029      -0.386      -0.
021
kp_o             0.0296      3.034      0.010      0.992      -5.917       5.
977
kp_ornk          0.0248      0.017      1.438      0.150      -0.009       0.
059
kp_d            -0.0146      3.028     -0.005      0.996      -5.949       5.
920
kp_drnk          0.0084      0.018      0.453      0.650      -0.028       0.
044
kp_temp          0.0062      0.315      0.020      0.984      -0.611       0.
623
kp_temrnk       -0.0061      0.009     -0.699      0.485      -0.023       0.
011
kp_luck          0.5690      6.757      0.084      0.933     -12.674      13.
812
kp_lucrnk       -0.0170      0.006     -2.900      0.004      -0.028      -0.
006
kp_sos           0.0610      0.167      0.365      0.715      -0.267       0.
389
kp_sosrnk       -0.0444      0.048     -0.935      0.350      -0.138       0.
049
kp_ncsos        -0.1572      0.239     -0.659      0.510      -0.625       0.
311
kp_ncsosrnk     -0.0023      0.011     -0.219      0.827      -0.023       0.
019
const           -2.5433     36.583     -0.070      0.945     -74.244      69.
158
==============================================================================
===
```

In [17]:
```
# Creating prediction probabilities and labels

pY_prob1 = md1.predict(vX1)
pY_prob1 = pY_prob1
pY1 = (pY_prob1 > 0.85) * 1
AUC = roc_auc_score(vY1, pY_prob1)

# Creating confusion matrix

dfCM = pd.DataFrame(confusion_matrix(vY1, pY1), index=['True-','True+'], colum
ns=['Pred-','Pred+'])
print(f'Confusion matrix:\n{dfCM}')
print(f'Out of sample accuracy: {np.mean(pY1 == vY1):.2f} and AUC:{AUC:.2f}')

# Creating ROC and AUC plot

fpr, tpr, thresholds = roc_curve(vY1, pY_prob1)

plt.rcParams['figure.figsize'] = [5, 5]
ax = pd.DataFrame([fpr, tpr], index=['fpr','tpr']).T.plot(
    'fpr','tpr', kind='line', grid=True, title='Receiver Operating Characteris
tic', label=f'ROC curve. AUC = {AUC:.2f}');

ax.plot([0, 1], [0, 1], 'r--');  # random predictions curve
ax.set_ylabel('True Positive Rate or (Sensitivity)');
ax.set_xlabel('False Positive Rate or (1 - Specifity)');
```

```
Confusion matrix:
       Pred-  Pred+
True-     40      1
True+      5      2
Out of sample accuracy: 0.88 and AUC:0.78
```



In [18]:
```
# Fitting the model to the 2023 NCAA Teams

pY_prob1 = md1.predict(df_new)
pY_prob1 = pY_prob1
pY1 = (pY_prob1 > 0.85) * 1
```

In [19]:
```
# Printing 2023 NCAA teams prediction probabilites and prediction labels for t
he sweet sixteen

df_prob = pd.Series(pY_prob1)
df_prob = pd.DataFrame(df_prob, columns=['prob'])
df_pred = pd.DataFrame(pY1, columns=['sixteen'])
prediction_results =df_prob.merge(df_pred['sixteen'], left_index=True, right_i
ndex=True)
prediction_results = prediction_results.merge(df_new_team['team'], left_index=
True, right_index=True).merge(df_new_team['seed'], left_index=True, right_inde
x=True).merge(df_new_team['region'], left_index=True, right_index=True)\
.merge(df_new_team['conf_champ'], left_index=True, right_index=True).merge(df_
new_team['tourn_champ'], left_index=True, right_index=True).merge(df_new_team
['sag_rate'], left_index=True, right_index=True).merge(df_new_team['kp_em_rn
k'], left_index=True, right_index=True)\
.merge(df_new_team['kp_ornk'], left_index=True, right_index=True).merge(df_new
_team['kp_drnk'], left_index=True, right_index=True)\
.merge(df_new_team['kp_lucrnk'], left_index=True, right_index=True).merge(df_n
ew_team['ap_final_rnk'], left_index=True, right_index=True).merge(df_new_team
['ap_final'], left_index=True, right_index=True)
prediction_results.sort_values(['prob'], ascending=False)
```

Out[19]:

	prob	sixteen	team	seed	region	conf_champ	tourn_champ	sag_rate	kp_em_
12	1.000000	1	Creighton	6	south	0	0	87.63	
7	0.980232	1	Gonzaga	3	west	1	1	90.47	
8	0.947771	1	Kansas	1	west	1	0	89.95	
13	0.942490	1	San Diego St.	5	south	1	1	85.19	
1	0.933380	1	UCLA	2	west	1	0	91.21	
14	0.916488	1	Baylor	3	south	0	0	87.36	
0	0.865725	1	Houston	1	midwest	1	0	91.85	
9	0.827693	0	Arizona	2	south	0	1	89.86	
2	0.821911	0	Alabama	1	south	1	1	92.33	
17	0.801904	0	Utah St.	10	south	0	0	84.12	
11	0.779840	0	Marquette	2	east	1	1	87.51	
3	0.763153	0	UConn	4	west	0	0	90.07	
6	0.695480	0	Purdue	1	east	1	1	89.16	
5	0.667406	0	Texas	2	midwest	0	1	90.46	
15	0.624780	0	Xavier	3	midwest	0	0	86.50	
10	0.588863	0	Saint Mary's	5	west	1	0	86.43	
25	0.583147	0	Florida Atlantic	9	east	1	1	83.66	
16	0.481044	0	West Virginia	9	south	0	0	85.86	
23	0.427605	0	Kansas St.	3	east	0	0	85.34	
18	0.374114	0	Memphis	8	east	0	1	86.53	
30	0.311849	0	Boise St.	10	west	0	0	81.89	
4	0.265009	0	Tennessee	4	east	0	0	89.28	
26	0.232904	0	TCU	6	west	0	0	85.95	
47	0.205123	0	Arizona St.	11	west	0	0	80.99	
19	0.189379	0	Arkansas	8	west	0	0	85.76	
20	0.087754	0	Duke	5	east	0	1	87.34	
24	0.080083	0	Texas A&M	7	midwest	0	0	86.08	
22	0.078740	0	Iowa St.	6	midwest	0	0	84.82	
42	0.071083	0	Missouri	7	south	0	0	82.38	
41	0.059848	0	Mississippi St.	11	midwest	0	0	82.26	
34	0.054833	0	USC	10	east	0	0	84.12	
28	0.052540	0	Auburn	9	midwest	0	0	85.32	
39	0.051805	0	Nevada	11	west	0	0	79.21	
36	0.033274	0	Penn St.	10	midwest	0	0	83.88	

	prob	sixteen	team	seed	region	conf_champ	tourn_champ	sag_rate	kp_em_
43	0.028193	0	NC State	11	south	0	0	83.25	
33	0.027431	0	Virginia	4	south	1	0	85.08	
29	0.025127	0	Indiana	4	midwest	0	0	86.21	
32	0.025030	0	Illinois	9	west	0	0	85.30	
38	0.019846	0	Northwestern	7	west	0	0	83.93	
31	0.016313	0	Michigan St.	7	east	0	0	84.93	
21	0.014475	0	Maryland	8	south	0	0	85.47	
27	0.013013	0	Kentucky	6	east	0	0	85.95	
40	0.012991	0	Providence	11	east	0	0	84.12	
35	0.009507	0	Iowa	8	midwest	0	0	84.85	
45	0.008502	0	VCU	12	west	1	1	83.45	
37	0.006341	0	Miami	5	midwest	1	0	84.18	
53	0.001562	0	Louisiana	13	east	0	1	77.39	
51	0.000551	0	Pitt	11	midwest	0	0	81.54	
44	0.000375	0	Oral Roberts	12	east	1	1	80.49	
48	0.000346	0	Kent St.	13	midwest	0	1	81.07	
46	0.000220	0	Drake	12	midwest	0	1	81.90	
49	0.000032	0	Charleston	12	south	1	1	80.69	
50	0.000015	0	Iona	13	west	1	1	79.70	
52	0.000002	0	Furman	13	south	1	1	79.48	

In [20]:
```
# Creating a new dataframe before calculating the elite eight teams

df_8 = df.copy()

# Dropping columns that won't be used in the modeling

df_8.drop(columns = ['team','region','conf','year','thirtytwo','sixteen','fou
r','championship','nat_champ','ap_pre','ap_high','ap_final'], inplace=True)

# Adding a constant to the model

df_8['const'] = 1
```

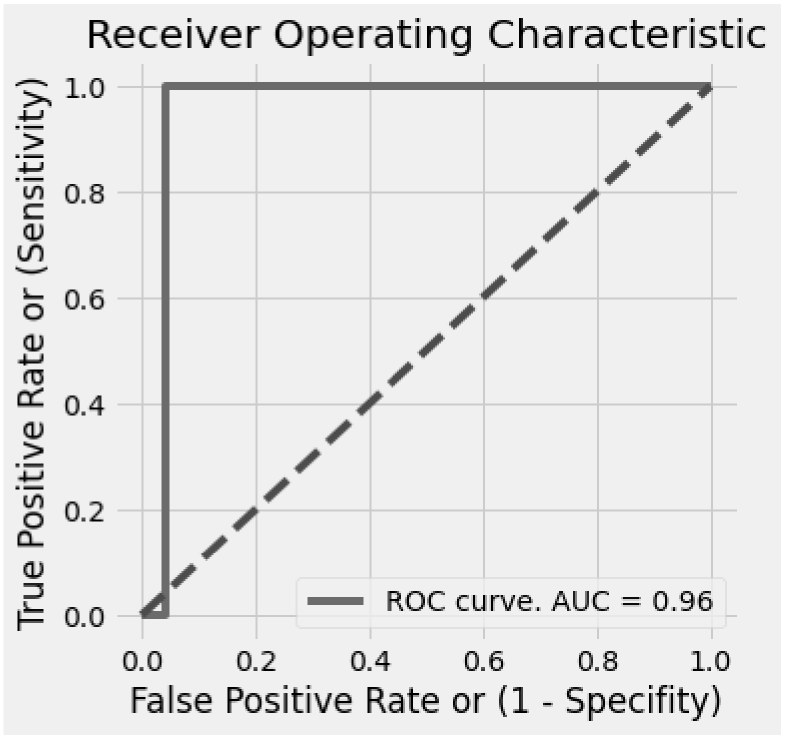
In [21]:
```
tX2, vX2, tY2, vY2 = tts(df_8.drop(['eight'], axis=1), df_8['eight'], test_siz
e = 0.2, random_state=123)

md2 = sm.Logit(tY2, tX2).fit(method = 'ncg')
print(md2.summary(title='NCAA Model - Elite Eight', alpha=.05))
```

```
Optimization terminated successfully.
         Current function value: 0.265176
         Iterations: 11
         Function evaluations: 15
         Gradient evaluations: 15
         Hessian evaluations: 11
                          NCAA Model - Elite Eight
=============================================================================
=
Dep. Variable:                  eight   No. Observations:                  19
2
Model:                          Logit   Df Residuals:                      15
9
Method:                           MLE   Df Model:                           3
2
Date:                Wed, 22 Mar 2023   Pseudo R-squ.:                  0.459
0
Time:                        19:16:11   Log-Likelihood:                -50.91
4
converged:                       True   LL-Null:                       -94.10
4
Covariance Type:            nonrobust   LLR p-value:                6.836e-0
7
=============================================================================
===
                  coef    std err          z      P>|z|      [0.025      0.9
75]
-----------------------------------------------------------------------------
---
seed            0.1231      0.239      0.515      0.607      -0.346       0.
592
power_five     -0.0073      0.867     -0.008      0.993      -1.707       1.
692
blue_blood      0.0103      1.648      0.006      0.995      -3.219       3.
240
new_blood       0.0062      0.974      0.006      0.995      -1.902       1.
914
true_blood     -0.0006      1.673     -0.000      1.000      -3.279       3.
278
new_coach      -0.0012      1.013     -0.001      0.999      -1.987       1.
984
champ_coach     0.0317      0.378      0.084      0.933      -0.710       0.
773
sch_champ       0.0912      0.199      0.458      0.647      -0.299       0.
482
tourn_champ     0.0149      0.750      0.020      0.984      -1.455       1.
485
conf_champ      0.0037      0.732      0.005      0.996      -1.431       1.
439
conf_rank       0.0194      0.263      0.074      0.941      -0.497       0.
536
ap_pre_rank    -0.0012      1.595     -0.001      0.999      -3.127       3.
124
ap_final_rnk   -0.0246      1.781     -0.014      0.989      -3.515       3.
466
ap_pre_final   -0.0090      1.797     -0.005      0.996      -3.532       3.
514
```

```
sea_mom         -0.0360      0.798     -0.045      0.964      -1.600       1.
528
sea_end_mom     -0.0256      0.906     -0.028      0.977      -1.801       1.
750
sag_rate         0.0350      0.443      0.079      0.937      -0.832       0.
903
sag_sche         0.0877      0.051      1.714      0.086      -0.013       0.
188
kp_em            0.1162      4.038      0.029      0.977      -7.799       8.
031
kp_em_rnk       -0.1168      0.099     -1.178      0.239      -0.311       0.
077
kp_o             0.0660      3.995      0.017      0.987      -7.764       7.
896
kp_ornk          0.0222      0.020      1.135      0.256      -0.016       0.
061
kp_d            -0.0707      3.983     -0.018      0.986      -7.876       7.
735
kp_drnk          0.0049      0.027      0.182      0.855      -0.048       0.
057
kp_temp         -0.0745      0.361     -0.206      0.836      -0.782       0.
633
kp_temrnk       -0.0025      0.010     -0.253      0.800      -0.022       0.
017
kp_luck          0.0087      9.338      0.001      0.999     -18.294      18.
311
kp_lucrnk       -0.0136      0.007     -2.055      0.040      -0.027      -0.
001
kp_sos           0.0080      0.176      0.046      0.964      -0.337       0.
353
kp_sosrnk       -0.1002      0.050     -1.988      0.047      -0.199      -0.
001
kp_ncsos        -0.0414      0.206     -0.201      0.841      -0.445       0.
363
kp_ncsosrnk      0.0006      0.010      0.061      0.951      -0.018       0.
019
const           -0.0005     40.096  -1.28e-05      1.000     -78.586      78.
585
==============================================================================
===
```

In [22]:
```
pY_prob2 = md2.predict(vX2)
pY_prob2 = pY_prob2
pY2 = (pY_prob2 > 0.50) * 1
AUC = roc_auc_score(vY2, pY_prob2)

dfCM = pd.DataFrame(confusion_matrix(vY2, pY2), index=['True-','True+'], colum
ns=['Pred-','Pred+'])
print(f'Confusion matrix:\n{dfCM}')
print(f'Out of sample accuracy: {np.mean(pY2 == vY2):.2f} and AUC:{AUC:.2f}')

fpr, tpr, thresholds = roc_curve(vY2, pY_prob2)

plt.rcParams['figure.figsize'] = [5, 5]
ax = pd.DataFrame([fpr, tpr], index=['fpr','tpr']).T.plot(
    'fpr','tpr', kind='line', grid=True, title='Receiver Operating Characteris
tic', label=f'ROC curve. AUC = {AUC:.2f}');

ax.plot([0, 1], [0, 1], 'r--');  # random predictions curve
ax.set_ylabel('True Positive Rate or (Sensitivity)');
ax.set_xlabel('False Positive Rate or (1 - Specifity)');
```

```
Confusion matrix:
       Pred-  Pred+
True-     45      2
True+      0      1
Out of sample accuracy: 0.96 and AUC:0.96
```



In [23]:
```
# Fitting the model to the 2023 NCAA Teams
pY_prob2 = md2.predict(df_new)
pY_prob2 = pY_prob2
pY2 = (pY_prob2 > 0.5) * 1
```

In [24]:
```
df_prob = pd.Series(round(pY_prob2,2))
df_prob = pd.DataFrame(df_prob, columns=['prob'])
df_pred = pd.DataFrame(pY2, columns=['eight'])
prediction_results =df_prob.merge(df_pred['eight'], left_index=True, right_ind
ex=True)
prediction_results = prediction_results.merge(df_new_team['team'], left_index=
True, right_index=True).merge(df_new_team['seed'], left_index=True, right_inde
x=True).merge(df_new_team['region'], left_index=True, right_index=True)\
.merge(df_new_team['conf_champ'], left_index=True, right_index=True).merge(df_
new_team['tourn_champ'], left_index=True, right_index=True).merge(df_new_team
['sag_sche'], left_index=True, right_index=True)\
.merge(df_new_team['sag_rate'], left_index=True, right_index=True).merge(df_ne
w_team['kp_em_rnk'], left_index=True, right_index=True).merge(df_new_team['kp_
ornk'], left_index=True, right_index=True)\
.merge(df_new_team['kp_drnk'], left_index=True, right_index=True).merge(df_new
_team['kp_lucrnk'], left_index=True, right_index=True).merge(df_new_team['ap_p
re'], left_index=True, right_index=True).merge(df_new_team['ap_final'], left_i
ndex=True, right_index=True)
prediction_results.sort_values(['prob'], ascending=False)
```

Out[24]:

	prob	eight	team	seed	region	conf_champ	tourn_champ	sag_sche	sag_rate	kp_
12	0.99	1	Creighton	6	south	0	0	16	87.63	
13	0.94	1	San Diego St.	5	south	1	1	72	85.19	
2	0.85	1	Alabama	1	south	1	1	11	92.33	
1	0.85	1	UCLA	2	west	1	0	43	91.21	
8	0.79	1	Kansas	1	west	1	0	1	89.95	
7	0.65	1	Gonzaga	3	west	1	1	71	90.47	
9	0.64	1	Arizona	2	south	0	1	46	89.86	
5	0.52	1	Texas	2	midwest	0	1	7	90.46	
0	0.48	0	Houston	1	midwest	1	0	88	91.85	
17	0.42	0	Utah St.	10	south	0	0	83	84.12	
6	0.28	0	Purdue	1	east	1	1	14	89.16	
14	0.26	0	Baylor	3	south	0	0	5	87.36	
23	0.26	0	Kansas St.	3	east	0	0	23	85.34	
30	0.23	0	Boise St.	10	west	0	0	80	81.89	
10	0.14	0	Saint Mary's	5	west	1	0	77	86.43	
4	0.11	0	Tennessee	4	east	0	0	38	89.28	
11	0.10	0	Marquette	2	east	1	1	35	87.51	
47	0.09	0	Arizona St.	11	west	0	0	44	80.99	
39	0.08	0	Nevada	11	west	0	0	84	79.21	
24	0.07	0	Texas A&M	7	midwest	0	0	51	86.08	
3	0.06	0	UConn	4	west	0	0	37	90.07	
25	0.05	0	Florida Atlantic	9	east	1	1	135	83.66	
22	0.05	0	Iowa St.	6	midwest	0	0	2	84.82	
19	0.04	0	Arkansas	8	west	0	0	28	85.76	
29	0.04	0	Indiana	4	midwest	0	0	12	86.21	
34	0.04	0	USC	10	east	0	0	63	84.12	
20	0.03	0	Duke	5	east	0	1	45	87.34	
26	0.03	0	TCU	6	west	0	0	15	85.95	
41	0.03	0	Mississippi St.	11	midwest	0	0	53	82.26	
16	0.03	0	West Virginia	9	south	0	0	4	85.86	
15	0.03	0	Xavier	3	midwest	0	0	22	86.50	
31	0.02	0	Michigan St.	7	east	0	0	6	84.93	
38	0.02	0	Northwestern	7	west	0	0	31	83.93	
42	0.01	0	Missouri	7	south	0	0	50	82.38	

	prob	eight	team	seed	region	conf_champ	tourn_champ	sag_sche	sag_rate	kp_
36	0.01	0	Penn St.	10	midwest	0	0	20	83.88	
27	0.01	0	Kentucky	6	east	0	0	41	85.95	
28	0.01	0	Auburn	9	midwest	0	0	27	85.32	
18	0.01	0	Memphis	8	east	0	1	57	86.53	
35	0.00	0	Iowa	8	midwest	0	0	17	84.85	
46	0.00	0	Drake	12	midwest	0	1	144	81.90	
52	0.00	0	Furman	13	south	1	1	227	79.48	
51	0.00	0	Pitt	11	midwest	0	0	74	81.54	
50	0.00	0	Iona	13	west	1	1	254	79.70	
49	0.00	0	Charleston	12	south	1	1	274	80.69	
48	0.00	0	Kent St.	13	midwest	0	1	161	81.07	
45	0.00	0	VCU	12	west	1	1	113	83.45	
37	0.00	0	Miami	5	midwest	1	0	68	84.18	
44	0.00	0	Oral Roberts	12	east	1	1	277	80.49	
43	0.00	0	NC State	11	south	0	0	67	83.25	
21	0.00	0	Maryland	8	south	0	0	34	85.47	
40	0.00	0	Providence	11	east	0	0	58	84.12	
32	0.00	0	Illinois	9	west	0	0	29	85.30	
33	0.00	0	Virginia	4	south	1	0	56	85.08	
53	0.00	0	Louisiana	13	east	0	1	162	77.39	

In [25]:
```
# Creating a new dataframe before calculating the final four teams

df_4 = df.copy()

# Dropping columns that won't be used in the modeling

df_4.drop(columns = ['team','region','conf','year','thirtytwo','sixteen','eigh
t','championship','nat_champ','ap_pre','ap_high','ap_final'], inplace=True)

# Adding a constant to the model

df_4['const'] = 1
```

In [26]:
```
tX3, vX3, tY3, vY3 = tts(df_4.drop(['four'], axis=1), df_4['four'], test_size
= 0.2, random_state=123)
```

In [27]:
```
# Logistic regression model summary

md3 = sm.Logit(tY3, tX3).fit(method = 'ncg')
print(md3.summary(title='NCAA Model - Final Four', alpha=.05))
```

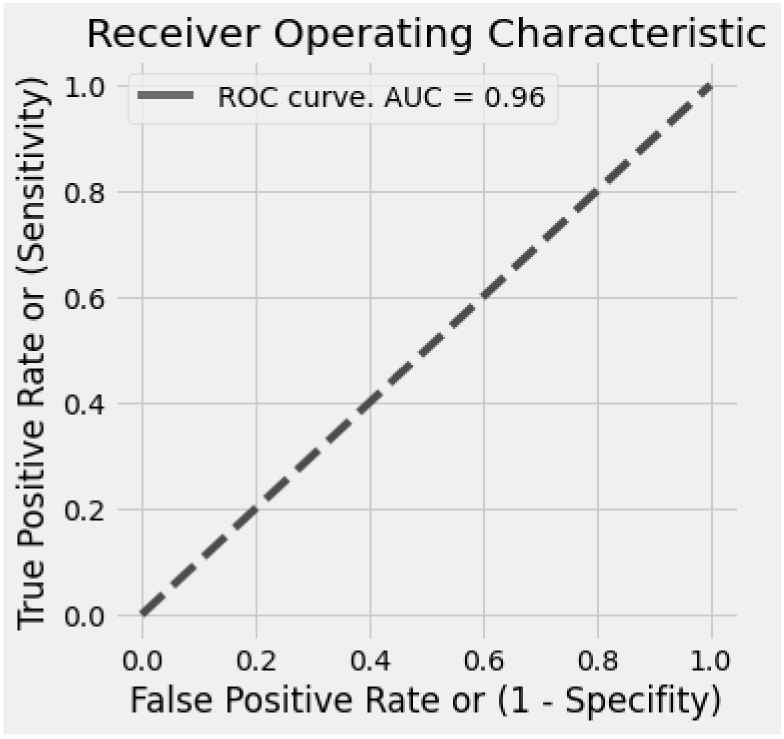
```
Optimization terminated successfully.
         Current function value: 0.169111
         Iterations: 16
         Function evaluations: 26
         Gradient evaluations: 26
         Hessian evaluations: 16
                           NCAA Model - Final Four
==============================================================================
=
Dep. Variable:                   four   No. Observations:                   19
2
Model:                          Logit   Df Residuals:                       15
9
Method:                           MLE   Df Model:                            3
2
Date:                Wed, 22 Mar 2023   Pseudo R-squ.:                   0.493
9
Time:                        19:16:13   Log-Likelihood:                 -32.46
9
converged:                       True   LL-Null:                        -64.15
5
Covariance Type:            nonrobust   LLR p-value:                  0.000784
9
==============================================================================
===
                   coef    std err          z      P>|z|      [0.025      0.9
75]
------------------------------------------------------------------------------
---
seed             0.1078      0.393      0.274      0.784      -0.663       0.
878
power_five      -0.0175      1.110     -0.016      0.987      -2.193       2.
158
blue_blood       0.0103      2.047      0.005      0.996      -4.002       4.
023
new_blood        0.0124      1.209      0.010      0.992      -2.357       2.
381
true_blood       0.0079      1.899      0.004      0.997      -3.714       3.
730
new_coach       -0.0070      1.461     -0.005      0.996      -2.870       2.
856
champ_coach      0.0329      0.480      0.068      0.945      -0.909       0.
974
sch_champ        0.1356      0.247      0.549      0.583      -0.348       0.
619
tourn_champ      0.0396      0.974      0.041      0.968      -1.869       1.
948
conf_champ       0.0062      1.005      0.006      0.995      -1.964       1.
976
conf_rank       -0.0203      0.449     -0.045      0.964      -0.900       0.
860
ap_pre_rank      0.0121      2.813      0.004      0.997      -5.501       5.
525
ap_final_rnk    -0.0352      2.919     -0.012      0.990      -5.757       5.
686
ap_pre_final    -0.0034      3.022     -0.001      0.999      -5.926       5.
919
```

```
sea_mom          -0.0533      1.049     -0.051      0.959      -2.110       2.
004
sea_end_mom      -0.0232      1.053     -0.022      0.982      -2.087       2.
041
sag_rate         -0.0235      0.699     -0.034      0.973      -1.393       1.
346
sag_sche          0.0698      0.077      0.912      0.362      -0.080       0.
220
kp_em             0.1702      5.724      0.030      0.976     -11.049      11.
389
kp_em_rnk        -0.2108      0.152     -1.383      0.167      -0.509       0.
088
kp_o              0.0430      5.717      0.008      0.994     -11.162      11.
248
kp_ornk           0.0430      0.029      1.473      0.141      -0.014       0.
100
kp_d             -0.0719      5.690     -0.013      0.990     -11.225      11.
081
kp_drnk           0.0276      0.042      0.661      0.508      -0.054       0.
109
kp_temp           0.0150      0.479      0.031      0.975      -0.925       0.
955
kp_temrnk         0.0044      0.013      0.326      0.744      -0.022       0.
031
kp_luck          -0.0236     13.365     -0.002      0.999     -26.218      26.
171
kp_lucrnk        -0.0218      0.011     -2.041      0.041      -0.043      -0.
001
kp_sos           -0.0227      0.280     -0.081      0.935      -0.571       0.
526
kp_sosrnk        -0.0781      0.079     -0.988      0.323      -0.233       0.
077
kp_ncsos         -0.0447      0.341     -0.131      0.896      -0.714       0.
624
kp_ncsosrnk      -0.0007      0.016     -0.048      0.962      -0.031       0.
030
const            -0.0017     55.422  -3.02e-05      1.000    -108.628     108.
624
==============================================================================
===
```

In [28]:
```
pY_prob3 = md3.predict(vX3)
pY_prob3 = pY_prob3
pY3 = (pY_prob3 > 0.72) * 1
#AUC = roc_auc_score(vY3, pY_prob3)

dfCM = pd.DataFrame(confusion_matrix(vY3, pY3), index=['True-','True+'], colum
ns=['Pred-','Pred+'])
print(f'Confusion matrix:\n{dfCM}')
print(f'Out of sample accuracy: {np.mean(pY3 == vY3):.2f} and AUC:{AUC:.2f}')

fpr, tpr, thresholds = roc_curve(vY3, pY_prob3)

plt.rcParams['figure.figsize'] = [5, 5]
ax = pd.DataFrame([fpr, tpr], index=['fpr','tpr']).T.plot(
    'fpr','tpr', kind='line', grid=True, title='Receiver Operating Characteris
tic', label=f'ROC curve. AUC = {AUC:.2f}');

ax.plot([0, 1], [0, 1], 'r--');  # random predictions curve
ax.set_ylabel('True Positive Rate or (Sensitivity)');
ax.set_xlabel('False Positive Rate or (1 - Specifity)');
```

```
Confusion matrix:
       Pred-  Pred+
True-     47      1
True+      0      0
Out of sample accuracy: 0.98 and AUC:0.96
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_ranking.py:1029: Unde
finedMetricWarning: No positive samples in y_true, true positive value should
be meaningless
  warnings.warn(
```



In [29]:
```
# Fitting the model to the 2023 NCAA Teams
pY_prob3 = md3.predict(df_new)
pY_prob3 = pY_prob3
pY3 = (pY_prob3 > 0.72) * 1
```

In [30]:
```
df_prob = pd.Series(round(pY_prob3,2))
df_prob = pd.DataFrame(df_prob, columns=['prob'])
df_pred = pd.DataFrame(pY3, columns=['four'])
prediction_results =df_prob.merge(df_pred['four'], left_index=True, right_inde
x=True)
prediction_results = prediction_results.merge(df_new_team['team'], left_index=
True, right_index=True).merge(df_new_team['seed'], left_index=True, right_inde
x=True).merge(df_new_team['region'], left_index=True, right_index=True)\
.merge(df_new_team['conf_champ'], left_index=True, right_index=True).merge(df_
new_team['tourn_champ'], left_index=True, right_index=True).merge(df_new_team
['sag_rate'], left_index=True, right_index=True)\
.merge(df_new_team['kp_em_rnk'], left_index=True, right_index=True).merge(df_n
ew_team['kp_ornk'], left_index=True, right_index=True).merge(df_new_team['kp_d
rnk'], left_index=True, right_index=True)\
.merge(df_new_team['kp_lucrnk'], left_index=True, right_index=True).merge(df_n
ew_team['ap_pre'], left_index=True, right_index=True).merge(df_new_team['ap_fi
nal'], left_index=True, right_index=True)
prediction_results.sort_values(['prob'], ascending=False)
```

Out[30]:

	prob	four	team	seed	region	conf_champ	tourn_champ	sag_rate	kp_em_rnk	kp
13	0.93	1	San Diego St.	5	south	1	1	85.19	14	
1	0.79	1	UCLA	2	west	1	0	91.21	2	
2	0.74	1	Alabama	1	south	1	1	92.33	3	
8	0.68	0	Kansas	1	west	1	0	89.95	9	
0	0.67	0	Houston	1	midwest	1	0	91.85	1	
7	0.59	0	Gonzaga	3	west	1	1	90.47	8	
6	0.32	0	Purdue	1	east	1	1	89.16	7	
9	0.30	0	Arizona	2	south	0	1	89.86	10	
14	0.24	0	Baylor	3	south	0	0	87.36	15	
5	0.18	0	Texas	2	midwest	0	1	90.46	6	
10	0.10	0	Saint Mary's	5	west	1	0	86.43	11	
17	0.09	0	Utah St.	10	south	0	0	84.12	18	
20	0.05	0	Duke	5	east	0	1	87.34	21	
11	0.04	0	Marquette	2	east	1	1	87.51	12	
30	0.03	0	Boise St.	10	west	0	0	81.89	31	
23	0.03	0	Kansas St.	3	east	0	0	85.34	24	
4	0.03	0	Tennessee	4	east	0	0	89.28	5	
25	0.02	0	Florida Atlantic	9	east	1	1	83.66	26	
47	0.02	0	Arizona St.	11	west	0	0	80.99	48	
24	0.02	0	Texas A&M	7	midwest	0	0	86.08	25	
41	0.01	0	Mississippi St.	11	midwest	0	0	82.26	42	
42	0.01	0	Missouri	7	south	0	0	82.38	43	
39	0.01	0	Nevada	11	west	0	0	79.21	40	
38	0.01	0	Northwestern	7	west	0	0	83.93	39	
22	0.01	0	Iowa St.	6	midwest	0	0	84.82	23	
3	0.01	0	UConn	4	west	0	0	90.07	4	
48	0.00	0	Kent St.	13	midwest	0	1	81.07	49	
46	0.00	0	Drake	12	midwest	0	1	81.90	47	
37	0.00	0	Miami	5	midwest	1	0	84.18	38	
45	0.00	0	VCU	12	west	1	1	83.45	46	
44	0.00	0	Oral Roberts	12	east	1	1	80.49	45	
50	0.00	0	Iona	13	west	1	1	79.70	51	
43	0.00	0	NC State	11	south	0	0	83.25	44	
51	0.00	0	Pitt	11	midwest	0	0	81.54	52	

	prob	four	team	seed	region	conf_champ	tourn_champ	sag_rate	kp_em_rnk	kp
52	0.00	0	Furman	13	south	1	1	79.48	53	
40	0.00	0	Providence	11	east	0	0	84.12	41	
49	0.00	0	Charleston	12	south	1	1	80.69	50	
27	0.00	0	Kentucky	6	east	0	0	85.95	28	
36	0.00	0	Penn St.	10	midwest	0	0	83.88	37	
35	0.00	0	Iowa	8	midwest	0	0	84.85	36	
34	0.00	0	USC	10	east	0	0	84.12	35	
33	0.00	0	Virginia	4	south	1	0	85.08	34	
32	0.00	0	Illinois	9	west	0	0	85.30	33	
31	0.00	0	Michigan St.	7	east	0	0	84.93	32	
29	0.00	0	Indiana	4	midwest	0	0	86.21	30	
28	0.00	0	Auburn	9	midwest	0	0	85.32	29	
26	0.00	0	TCU	6	west	0	0	85.95	27	
21	0.00	0	Maryland	8	south	0	0	85.47	22	
19	0.00	0	Arkansas	8	west	0	0	85.76	20	
18	0.00	0	Memphis	8	east	0	1	86.53	19	
16	0.00	0	West Virginia	9	south	0	0	85.86	17	
15	0.00	0	Xavier	3	midwest	0	0	86.50	16	
12	0.00	0	Creighton	6	south	0	0	87.63	13	
53	0.00	0	Louisiana	13	east	0	1	77.39	54	

In [31]:
```
# Creating a new dataframe before calculating the championship teams

df_2 = df.copy()

# Dropping columns that won't be used in the modeling

df_2.drop(columns = ['team','region','conf','year','thirtytwo','sixteen','eigh
t','four','nat_champ','ap_pre','ap_high','ap_final'], inplace=True)

# Adding a constant to the model

df_2['const'] = 1
```

In [32]:
```
tX4, vX4, tY4, vY4 = tts(df_2.drop(['championship'], axis=1), df_2['championsh
ip'], test_size = 0.2, random_state=123)
```

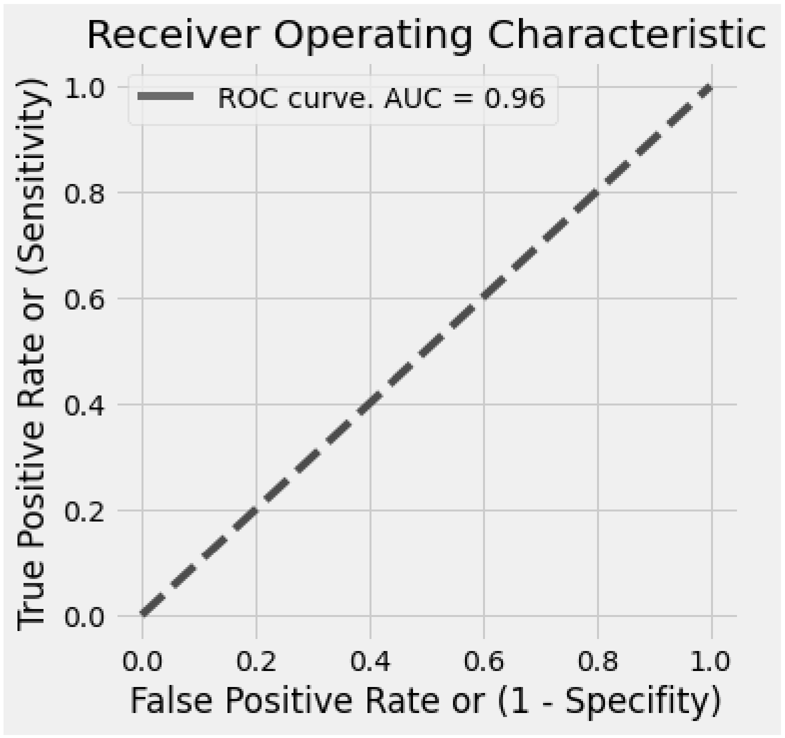
In [33]:
```
# Logistic regression model summary

md4 = sm.Logit(tY4, tX4).fit(method='ncg')
print(md4.summary(title='NCAA Model - Championship Game', alpha=.05))
```

```
Optimization terminated successfully.
         Current function value: 0.045617
         Iterations: 23
         Function evaluations: 33
         Gradient evaluations: 33
         Hessian evaluations: 23
                        NCAA Model - Championship Game
==============================================================================
=
Dep. Variable:           championship   No. Observations:                  19
2
Model:                          Logit   Df Residuals:                      15
9
Method:                           MLE   Df Model:                           3
2
Date:                Wed, 22 Mar 2023   Pseudo R-squ.:                  0.777
0
Time:                        19:16:14   Log-Likelihood:                -8.758
5
converged:                       True   LL-Null:                       -39.28
4
Covariance Type:            nonrobust   LLR p-value:                  0.00147
3
==============================================================================
===
                   coef    std err          z      P>|z|      [0.025      0.9
75]
------------------------------------------------------------------------------
---
seed             0.3490      1.782      0.196      0.845      -3.143       3.
841
power_five      -0.1221      2.382     -0.051      0.959      -4.790       4.
546
blue_blood      -0.0020      7.549     -0.000      1.000     -14.798      14.
794
new_blood        0.1068      4.262      0.025      0.980      -8.246       8.
460
true_blood      -0.0391      4.317     -0.009      0.993      -8.500       8.
422
new_coach        0.0496      5.558      0.009      0.993     -10.844      10.
943
champ_coach     -0.2312      1.771     -0.131      0.896      -3.702       3.
239
sch_champ        0.0789      1.349      0.058      0.953      -2.565       2.
723
tourn_champ      0.0542      2.881      0.019      0.985      -5.592       5.
701
conf_champ      -0.0387      3.732     -0.010      0.992      -7.352       7.
275
conf_rank       -0.0560      2.661     -0.021      0.983      -5.271       5.
159
ap_pre_rank     -0.0082     30.824     -0.000      1.000     -60.422      60.
405
ap_final_rnk    -0.0893     30.589     -0.003      0.998     -60.043      59.
865
ap_pre_final    -0.0951     29.915     -0.003      0.997     -58.728      58.
538
```

```
sea_mom         -0.0305      3.197     -0.010      0.992      -6.297       6.
236
sea_end_mom     -0.0444      3.064     -0.014      0.988      -6.050       5.
961
sag_rate        -0.0779      3.020     -0.026      0.979      -5.996       5.
840
sag_sche         0.1560      0.303      0.516      0.606      -0.437       0.
749
kp_em            0.4025     34.359      0.012      0.991     -66.941      67.
746
kp_em_rnk       -0.2476      0.558     -0.444      0.657      -1.341       0.
846
kp_o             0.1443     34.578      0.004      0.997     -67.627      67.
915
kp_ornk         -0.0028      0.403     -0.007      0.994      -0.793       0.
787
kp_d            -0.1315     34.546     -0.004      0.997     -67.841      67.
578
kp_drnk         -0.1197      0.408     -0.294      0.769      -0.918       0.
679
kp_temp          0.0269      1.974      0.014      0.989      -3.843       3.
897
kp_temrnk       -0.0060      0.060     -0.100      0.920      -0.123       0.
111
kp_luck         -0.0529     82.695     -0.001      0.999    -162.131     162.
025
kp_lucrnk       -0.0433      0.052     -0.831      0.406      -0.145       0.
059
kp_sos          -0.2532      1.074     -0.236      0.814      -2.358       1.
851
kp_sosrnk       -0.2161      0.356     -0.608      0.543      -0.913       0.
481
kp_ncsos        -0.1797      1.083     -0.166      0.868      -2.302       1.
943
kp_ncsosrnk      0.0033      0.048      0.069      0.945      -0.090       0.
097
const           -0.0031    249.595  -1.22e-05      1.000    -489.200     489.
194
==============================================================================
===

Possibly complete quasi-separation: A fraction 0.66 of observations can be
perfectly predicted. This might indicate that there is complete
quasi-separation. In this case some parameters will not be identified.
```

In [34]:
```
pY_prob4 = md4.predict(vX4)
pY_prob4 = pY_prob4
pY4 = (pY_prob4 > 0.25) * 1
#AUC = roc_auc_score(vY4, pY_prob4)

#dfCM = pd.DataFrame(confusion_matrix(vY4, pY4), index=['True-','True+'], colu
mns=['Pred-','Pred+'])
#print(f'Confusion matrix:\n{dfCM}')
print(f'Out of sample accuracy: {np.mean(pY4 == vY4):.2f} and AUC:{AUC:.2f}')

fpr, tpr, thresholds = roc_curve(vY4, pY_prob4)

plt.rcParams['figure.figsize'] = [5, 5]
ax = pd.DataFrame([fpr, tpr], index=['fpr','tpr']).T.plot(
    'fpr','tpr', kind='line', grid=True, title='Receiver Operating Characteris
tic', label=f'ROC curve. AUC = {AUC:.2f}');

ax.plot([0, 1], [0, 1], 'r--');  # random predictions curve
ax.set_ylabel('True Positive Rate or (Sensitivity)');
ax.set_xlabel('False Positive Rate or (1 - Specifity)');
```

```
Out of sample accuracy: 0.98 and AUC:0.96

/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_ranking.py:1029: Unde
finedMetricWarning: No positive samples in y_true, true positive value should
be meaningless
  warnings.warn(
```



In [35]:
```
# Fitting the model to the 2023 NCAA Teams
pY_prob4 = md4.predict(df_new)
pY_prob4 = pY_prob4
pY4 = (pY_prob4 > 0.25) * 1
```

In [36]:
```
df_prob = pd.Series(round(pY_prob4,3))
df_prob = pd.DataFrame(df_prob, columns=['prob'])
df_pred = pd.DataFrame(pY4, columns=['champgame'])
prediction_results =df_prob.merge(df_pred['champgame'], left_index=True, right
_index=True)
prediction_results = prediction_results.merge(df_new_team['team'], left_index=
True, right_index=True).merge(df_new_team['seed'], left_index=True, right_inde
x=True).merge(df_new_team['region'], left_index=True, right_index=True)\
.merge(df_new_team['conf_champ'], left_index=True, right_index=True).merge(df_
new_team['tourn_champ'], left_index=True, right_index=True).merge(df_new_team
['sag_rate'], left_index=True, right_index=True).merge(df_new_team['kp_ornk'],
left_index=True, right_index=True)\
.merge(df_new_team['kp_lucrnk'], left_index=True, right_index=True).merge(df_n
ew_team['ap_final'], left_index=True, right_index=True)
prediction_results.sort_values(['prob'], ascending=False)
```

Out[36]:

	prob	champgame	team	seed	region	conf_champ	tourn_champ	sag_rate	kp_orn
2	0.953	1	Alabama	1	south	1	1	92.33	1
13	0.418	1	San Diego St.	5	south	1	1	85.19	6
8	0.378	1	Kansas	1	west	1	0	89.95	2
1	0.290	1	UCLA	2	west	1	0	91.21	2
5	0.087	0	Texas	2	midwest	0	1	90.46	1
0	0.039	0	Houston	1	midwest	1	0	91.85	1
9	0.012	0	Arizona	2	south	0	1	89.86	
6	0.002	0	Purdue	1	east	1	1	89.16	
23	0.002	0	Kansas St.	3	east	0	0	85.34	5
34	0.000	0	USC	10	east	0	0	84.12	4
41	0.000	0	Mississippi St.	11	midwest	0	0	82.26	16
35	0.000	0	Iowa	8	midwest	0	0	84.85	
36	0.000	0	Penn St.	10	midwest	0	0	83.88	1
37	0.000	0	Miami	5	midwest	1	0	84.18	1
33	0.000	0	Virginia	4	south	1	0	85.08	7
38	0.000	0	Northwestern	7	west	0	0	83.93	10
39	0.000	0	Nevada	11	west	0	0	79.21	6
40	0.000	0	Providence	11	east	0	0	84.12	1
45	0.000	0	VCU	12	west	1	1	83.45	14
42	0.000	0	Missouri	7	south	0	0	82.38	1
43	0.000	0	NC State	11	south	0	0	83.25	3
44	0.000	0	Oral Roberts	12	east	1	1	80.49	2
31	0.000	0	Michigan St.	7	east	0	0	84.93	4
46	0.000	0	Drake	12	midwest	0	1	81.90	9
47	0.000	0	Arizona St.	11	west	0	0	80.99	13
48	0.000	0	Kent St.	13	midwest	0	1	81.07	11
49	0.000	0	Charleston	12	south	1	1	80.69	7
50	0.000	0	Iona	13	west	1	1	79.70	8
51	0.000	0	Pitt	11	midwest	0	0	81.54	2
52	0.000	0	Furman	13	south	1	1	79.48	3
32	0.000	0	Illinois	9	west	0	0	85.30	5
27	0.000	0	Kentucky	6	east	0	0	85.95	1
30	0.000	0	Boise St.	10	west	0	0	81.89	7
29	0.000	0	Indiana	4	midwest	0	0	86.21	2

	prob	champgame	team	seed	region	conf_champ	tourn_champ	sag_rate	kp_orn
3	0.000	0	UConn	4	west	0	0	90.07	
4	0.000	0	Tennessee	4	east	0	0	89.28	4
7	0.000	0	Gonzaga	3	west	1	1	90.47	
10	0.000	0	Saint Mary's	5	west	1	0	86.43	4
11	0.000	0	Marquette	2	east	1	1	87.51	
12	0.000	0	Creighton	6	south	0	0	87.63	2
14	0.000	0	Baylor	3	south	0	0	87.36	
15	0.000	0	Xavier	3	midwest	0	0	86.50	
16	0.000	0	West Virginia	9	south	0	0	85.86	1
17	0.000	0	Utah St.	10	south	0	0	84.12	1
18	0.000	0	Memphis	8	east	0	1	86.53	2
19	0.000	0	Arkansas	8	west	0	0	85.76	5
20	0.000	0	Duke	5	east	0	1	87.34	4
21	0.000	0	Maryland	8	south	0	0	85.47	3
22	0.000	0	Iowa St.	6	midwest	0	0	84.82	9
24	0.000	0	Texas A&M	7	midwest	0	0	86.08	3
25	0.000	0	Florida Atlantic	9	east	1	1	83.66	3
26	0.000	0	TCU	6	west	0	0	85.95	5
28	0.000	0	Auburn	9	midwest	0	0	85.32	4
53	0.000	0	Louisiana	13	east	0	1	77.39	5

In [37]:
```
# Creating a new dataframe before calculating the championship teams

df_1 = df.copy()

# Dropping columns that won't be used in the modeling

df_1.drop(columns = ['team','region','conf','year','thirtytwo','sixteen','eigh
t','four','championship','ap_pre','ap_high','ap_final'], inplace=True)

# Adding a constant to the model

df_1['const'] = 1
```

In [38]:
```
tX5, vX5, tY5, vY5 = tts(df_1.drop(['nat_champ'], axis=1), df_1['nat_champ'],
test_size = 0.2, random_state=123)
```

In [39]:
```
# Logistic regression model summary

md5 = sm.Logit(tY5, tX5).fit(method='minimize') #method='ncg'
print(md5.summary(title='NCAA Model - Championship Game', alpha=.05))
```

```
Warning: Maximum number of iterations has been exceeded.
         Current function value: 0.006296
         Iterations: 35
         Function evaluations: 48
         Gradient evaluations: 48
                              NCAA Model - Championship Game
==============================================================================
=
Dep. Variable:              nat_champ   No. Observations:                   19
2
Model:                          Logit   Df Residuals:                       15
9
Method:                           MLE   Df Model:                            3
2
Date:                Wed, 22 Mar 2023   Pseudo R-squ.:                   0.947
8
Time:                        19:16:15   Log-Likelihood:                 -1.208
9
converged:                      False   LL-Null:                        -23.17
5
Covariance Type:            nonrobust   LLR p-value:                    0.0779
0
==============================================================================
===
                  coef    std err          z      P>|z|      [0.025      0.9
75]
------------------------------------------------------------------------------
---
seed            1.6147    108.339      0.015      0.988    -210.727     213.
956
power_five     -2.4950    206.549     -0.012      0.990    -407.323     402.
333
blue_blood      0.4281    366.690      0.001      0.999    -718.272     719.
128
new_blood      -0.3968    402.515     -0.001      0.999    -789.311     788.
517
true_blood     -0.9703    365.970     -0.003      0.998    -718.258     716.
318
new_coach      -0.4033    511.840     -0.001      0.999   -1003.590    1002.
784
champ_coach    -1.1731    114.761     -0.010      0.992    -226.100     223.
753
sch_champ       0.9242     77.939      0.012      0.991    -151.833     153.
681
tourn_champ    -0.0637    241.844     -0.000      1.000    -474.069     473.
942
conf_champ     -0.0848    179.817     -0.000      1.000    -352.519     352.
349
conf_rank      -0.5065    166.581     -0.003      0.998    -326.999     325.
986
ap_pre_rank     0.0773    818.036   9.44e-05      1.000   -1603.244    1603.
398
ap_final_rnk   -0.5929    650.033     -0.001      0.999   -1274.633    1273.
447
ap_pre_final   -0.0283    864.907  -3.27e-05      1.000   -1695.215    1695.
159
sea_mom        -2.1969    334.693     -0.007      0.995    -658.184     653.
```

```
790
sea_end_mom     -0.8162    212.126     -0.004      0.997    -416.575     414.
943
sag_rate        -3.9113    352.774     -0.011      0.991    -695.337     687.
514
sag_sche        -0.5421     49.722     -0.011      0.991     -97.996      96.
912
kp_em            3.1588   2716.993      0.001      0.999   -5322.050    5328.
367
kp_em_rnk       -0.0040     43.579  -9.22e-05      1.000     -85.417      85.
409
kp_o             1.7769   2574.437      0.001      0.999   -5044.026    5047.
580
kp_ornk          0.1437     15.777      0.009      0.993     -30.780      31.
067
kp_d             1.5352   2564.975      0.001      1.000   -5025.724    5028.
794
kp_drnk         -1.1247     18.528     -0.061      0.952     -37.438      35.
189
kp_temp         -0.8241    127.488     -0.006      0.995    -250.697     249.
049
kp_temrnk       -0.0306      4.848     -0.006      0.995      -9.532       9.
470
kp_luck         -1.3231   1.99e+04  -6.63e-05      1.000   -3.91e+04    3.91e
+04
kp_lucrnk       -0.0754      9.186     -0.008      0.993     -18.079      17.
928
kp_sos           1.8258    118.749      0.015      0.988    -230.918     234.
570
kp_sosrnk        0.5016     61.382      0.008      0.993    -119.804     120.
807
kp_ncsos        -3.0728    111.454     -0.028      0.978    -221.519     215.
374
kp_ncsosrnk     -0.1559      7.374     -0.021      0.983     -14.608      14.
296
const           -0.0682   2.61e+04  -2.62e-06      1.000   -5.11e+04    5.11e
+04
==============================================================================
===

Possibly complete quasi-separation: A fraction 0.90 of observations can be
perfectly predicted. This might indicate that there is complete
quasi-separation. In this case some parameters will not be identified.

/usr/local/lib/python3.9/dist-packages/statsmodels/base/model.py:604: Converg
enceWarning: Maximum Likelihood optimization failed to converge. Check mle_re
tvals
  warnings.warn("Maximum Likelihood optimization failed to "
```

In [40]:
```
pY_prob5 = md5.predict(vX5)
pY_prob5 = pY_prob5
pY5 = (pY_prob5 > 0.25) * 1
#AUC = roc_auc_score(vY5, pY_prob5)

#dfCM = pd.DataFrame(confusion_matrix(vY4, pY4), index=['True-','True+'], colu
mns=['Pred-','Pred+'])
#print(f'Confusion matrix:\n{dfCM}')
print(f'Out of sample accuracy: {np.mean(pY5 == vY5):.2f}')
# and AUC:{AUC:.2f}')

fpr, tpr, thresholds = roc_curve(vY5, pY_prob5)

plt.rcParams['figure.figsize'] = [5, 5]
#ax = pd.DataFrame([fpr, tpr], index=['fpr','tpr']).T.plot(
#    'fpr','tpr', kind='line', grid=True, title='Receiver Operating Characteri
stic', label=f'ROC curve. AUC = {AUC:.2f}');

ax.plot([0, 1], [0, 1], 'r--');  # random predictions curve
ax.set_ylabel('True Positive Rate or (Sensitivity)');
ax.set_xlabel('False Positive Rate or (1 - Specifity)');
```

```
Out of sample accuracy: 0.96
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_ranking.py:1029: Unde
finedMetricWarning: No positive samples in y_true, true positive value should
be meaningless
  warnings.warn(
```

In [41]:
```
# Fitting the model to the 2023 NCAA Teams
pY_prob5 = md5.predict(df_new)
pY_prob5 = pY_prob5
pY5 = (pY_prob5 > 0.25) * 1
```

In [42]:
```
df_prob = pd.Series(round(pY_prob5,3))
df_prob = pd.DataFrame(df_prob, columns=['prob'])
df_pred = pd.DataFrame(pY5, columns=['champion'])
prediction_results =df_prob.merge(df_pred['champion'], left_index=True, right_
index=True)
prediction_results = prediction_results.merge(df_new_team['team'], left_index=
True, right_index=True).merge(df_new_team['seed'], left_index=True, right_inde
x=True).merge(df_new_team['region'], left_index=True, right_index=True)\
.merge(df_new_team['conf_champ'], left_index=True, right_index=True).merge(df_
new_team['tourn_champ'], left_index=True, right_index=True).merge(df_new_team
['sag_rate'], left_index=True, right_index=True).merge(df_new_team['kp_ornk'],
left_index=True, right_index=True)\
.merge(df_new_team['kp_lucrnk'], left_index=True, right_index=True).merge(df_n
ew_team['ap_final'], left_index=True, right_index=True)
prediction_results.sort_values(['prob'], ascending=False)
```

Out[42]:

	prob	champion	team	seed	region	conf_champ	tourn_champ	sag_rate	kp_ornk
12	1.000	1	Creighton	6	south	0	0	87.63	28
8	0.831	1	Kansas	1	west	1	0	89.95	29
0	0.000	0	Houston	1	midwest	1	0	91.85	11
41	0.000	0	Mississippi St.	11	midwest	0	0	82.26	164
30	0.000	0	Boise St.	10	west	0	0	81.89	78
31	0.000	0	Michigan St.	7	east	0	0	84.93	41
32	0.000	0	Illinois	9	west	0	0	85.30	58
33	0.000	0	Virginia	4	south	1	0	85.08	74
34	0.000	0	USC	10	east	0	0	84.12	43
35	0.000	0	Iowa	8	midwest	0	0	84.85	3
36	0.000	0	Penn St.	10	midwest	0	0	83.88	17
37	0.000	0	Miami	5	midwest	1	0	84.18	12
38	0.000	0	Northwestern	7	west	0	0	83.93	109
39	0.000	0	Nevada	11	west	0	0	79.21	61
40	0.000	0	Providence	11	east	0	0	84.12	16
42	0.000	0	Missouri	7	south	0	0	82.38	10
28	0.000	0	Auburn	9	midwest	0	0	85.32	48
43	0.000	0	NC State	11	south	0	0	83.25	37
44	0.000	0	Oral Roberts	12	east	1	1	80.49	23
45	0.000	0	VCU	12	west	1	1	83.45	140
46	0.000	0	Drake	12	midwest	0	1	81.90	98
47	0.000	0	Arizona St.	11	west	0	0	80.99	133
48	0.000	0	Kent St.	13	midwest	0	1	81.07	111
49	0.000	0	Charleston	12	south	1	1	80.69	70
50	0.000	0	Iona	13	west	1	1	79.70	80
51	0.000	0	Pitt	11	midwest	0	0	81.54	24
52	0.000	0	Furman	13	south	1	1	79.48	33
29	0.000	0	Indiana	4	midwest	0	0	86.21	27
27	0.000	0	Kentucky	6	east	0	0	85.95	14
1	0.000	0	UCLA	2	west	1	0	91.21	25
14	0.000	0	Baylor	3	south	0	0	87.36	2
2	0.000	0	Alabama	1	south	1	1	92.33	19
3	0.000	0	UConn	4	west	0	0	90.07	6
4	0.000	0	Tennessee	4	east	0	0	89.28	49
5	0.000	0	Texas	2	midwest	0	1	90.46	18

	prob	champion	team	seed	region	conf_champ	tourn_champ	sag_rate	kp_ornk
6	0.000	0	Purdue	1	east	1	1	89.16	7
7	0.000	0	Gonzaga	3	west	1	1	90.47	1
9	0.000	0	Arizona	2	south	0	1	89.86	4
10	0.000	0	Saint Mary's	5	west	1	0	86.43	40
11	0.000	0	Marquette	2	east	1	1	87.51	8
13	0.000	0	San Diego St.	5	south	1	1	85.19	64
15	0.000	0	Xavier	3	midwest	0	0	86.50	9
26	0.000	0	TCU	6	west	0	0	85.95	53
16	0.000	0	West Virginia	9	south	0	0	85.86	15
17	0.000	0	Utah St.	10	south	0	0	84.12	13
18	0.000	0	Memphis	8	east	0	1	86.53	26
19	0.000	0	Arkansas	8	west	0	0	85.76	51
20	0.000	0	Duke	5	east	0	1	87.34	42
21	0.000	0	Maryland	8	south	0	0	85.47	35
22	0.000	0	Iowa St.	6	midwest	0	0	84.82	96
23	0.000	0	Kansas St.	3	east	0	0	85.34	52
24	0.000	0	Texas A&M	7	midwest	0	0	86.08	30
25	0.000	0	Florida Atlantic	9	east	1	1	83.66	32
53	0.000	0	Louisiana	13	east	0	1	77.39	57