Data preparation, exploration, visualization

For this assignment I decided to use the package "pandas profiling report" to do my initial data prep and exploration. From the profiling report I saw that 8.1% of the data was missing, there are 12 variables and were divided into 6 categorical variables, 5 numerical variables and 1 bool type variable. From lines 4-13 I analyzed the size of the training data and the testing data, created a data frame for the testing data and dropped the "survived" column since that is the responsive variable. I looked at the correlation between the variables PClass and survived and found that most passengers from the lowest class didn't survive the event (line 5). I also looked at Gender and survival on line 511, and found that the most amount of deaths came from males (around 450 did not survive).

Review research design and modeling methods

In order to come up with the best predictive model to see which passengers survived the Titanic, I used the classification techniques of logistic regression and naïve bayes classification. First I wanted to transform the dataset used to reflect the variables I wanted to use in my model in line 99. I imputed the values for age, created new column for traveling alone passengers, and decided to use the variables Pclass, sex, Embarked, Deck, Family Size for my model inputs based on my EDA. I then retrained the this training set in line 100 and made sure there were the same numbers of rows and columns in the original test set in line 101.

Review results, evaluate models

The first classifier I used was the naïve bayes classifcation in line 163. I received an accuracy of around 43%. In line 164, I made an ROC curve to juxtapose the true false positive rate with the false positive rate and found that the area under the curve was around 0.54. This showed me that the naives bayes classification was not very accurate in determining the survival rate based on passenger qualities. However, the logistic regression I made on line 123 had an 81% accuracy rate and in line 149, the ROC curve showed an area under the curve of 0.8. This means that the logistic regression I developed was way

more accurate in predicting survival of passengers on the Titanic. I then exported out these results in lines 150 and 158 and used these results for my Kaggle submission.

Implementation and programming as evidenced by Kaggle submission

https://www.kaggle.com/c/titanic/leaderboard#score My Kaggle score was 0.7674 and my rank was 146.

Exposition, problem description, and management recommendations

As for answering the management question of what types of characteristics were associated with survival of the titanic, I would strongly recommend using a logistic regression to model this information. While a naïve bayes classifier is great because it assumes the presence of a feature in a class is unrelated to any other feature, this is not particularly helpful for this dataset. I found that using a logistic regression provided a much more accurate model for predicting the survival rate of the Titanic. This could be due to the fact that the variables I chose to exam (PClass, family size, embarkment, Sex, Deck) are interrelated as I saw in my EDA. For example, I found that the larger a family size was, they were typically in the lowest class. I also found that certain families from a class embarked from on particular port vs families from a higher class. Lastly, the number of women saved from the Titanic were much higher than the men saved, while most of the women saved came from a higher class. Because there is a large amount of correlation between the variables/features of a Titanic passenger, I believe the logistic regression reflected a higher accuracy and this was proven when I cross validated it with the ROC curve. As for providing evidence to the type of person that didn't survive the Titanic, I would say that if a person on board were a male, from the lowest class with a large family (3 children or above), they would likely not have survived. However, if you were an upper class woman with a small family, your chances of surviving were extremely high. I hope to understand how to utilize a naïve bayes classification in the future to get a more accurate understanding of how to use this type of classification, but for now I believe the logistic regression was the best model for this specific problem.

In [1]:
```
import pandas as pd
import math
from math import sqrt
import numpy as np
import pandas_profiling
from pandas_profiling import ProfileReport as pp
import random
np.random.seed(42)

import matplotlib.pyplot as plt
import seaborn as sns
import IPython
from IPython.display import display
from scipy.stats import skew

%matplotlib inline

import sklearn
```

```
/Users/avadhani/anaconda3/lib/python3.7/site-packages/statsmodels/to
ols/_testing.py:19: FutureWarning: pandas.util.testing is deprecated
. Use the functions in the public API at pandas.testing instead.
  import pandas.util.testing as tm
```

In [2]:
```
train = pd.read_csv("train.csv")
test = pd.read_csv("test.csv")
```

In [3]:
```
pp(train, title = 'Pandas Profiling Report')
```

Overview

Dataset statistics

Number of variables	12
Number of observations	891
Missing cells	866
Missing cells (%)	8.1%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	83.7 KiB
Average record size in memory	96.1 B

Variable types

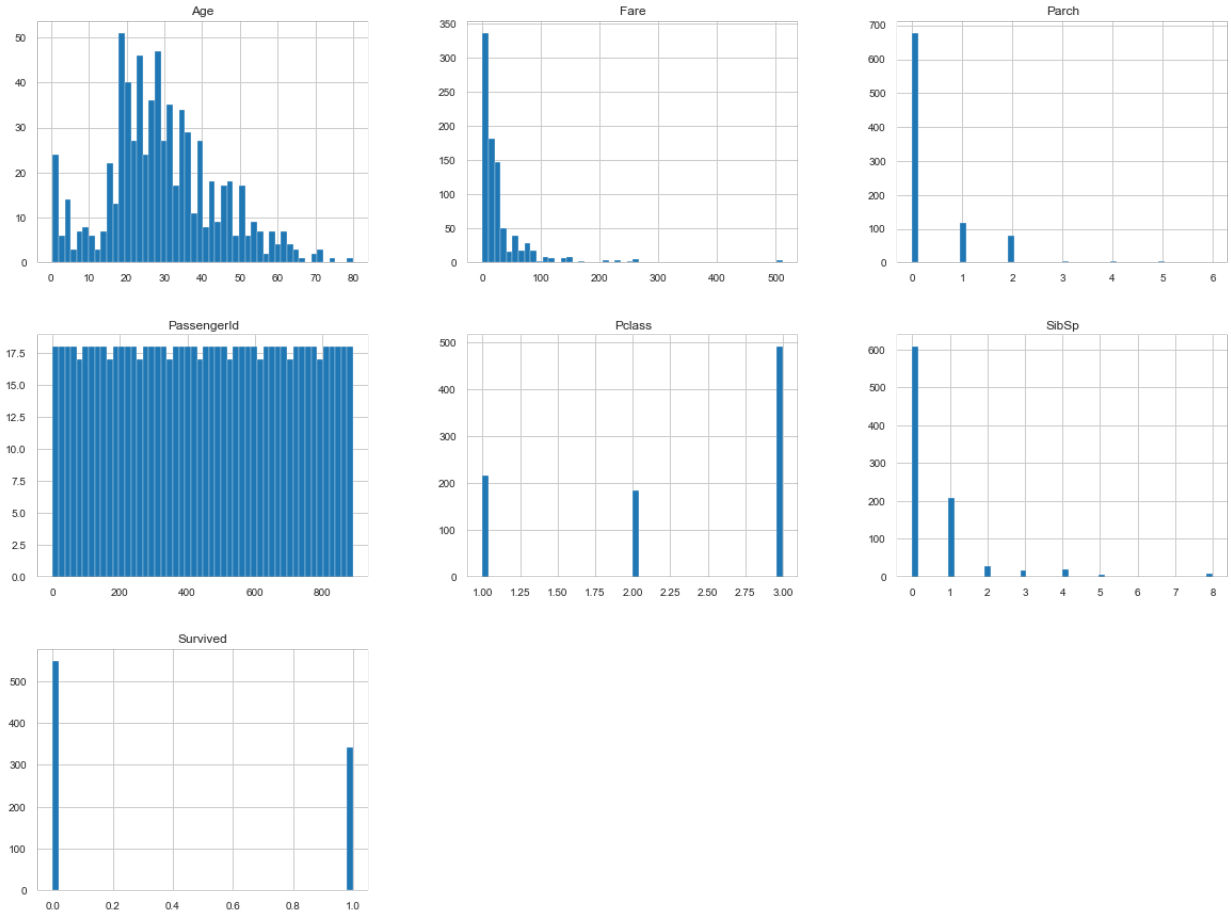
CAT	6
NUM	5
BOOL	1

Reproduction

Analysis started	2020-07-08 01:00:51.184925
Analysis finished	2020-07-08 01:00:58.486819

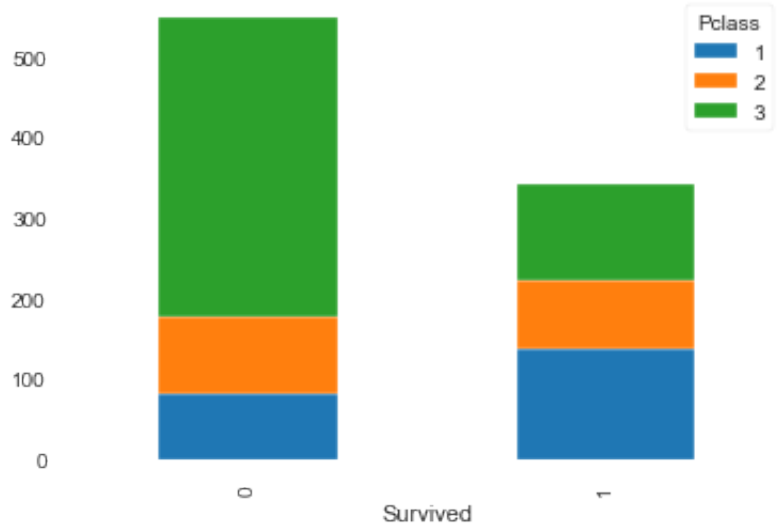
Out[3]:

In [4]:
```
%matplotlib inline
import matplotlib.pyplot as plt
train.hist(bins=50, figsize=(20,15))
plt.show()
```

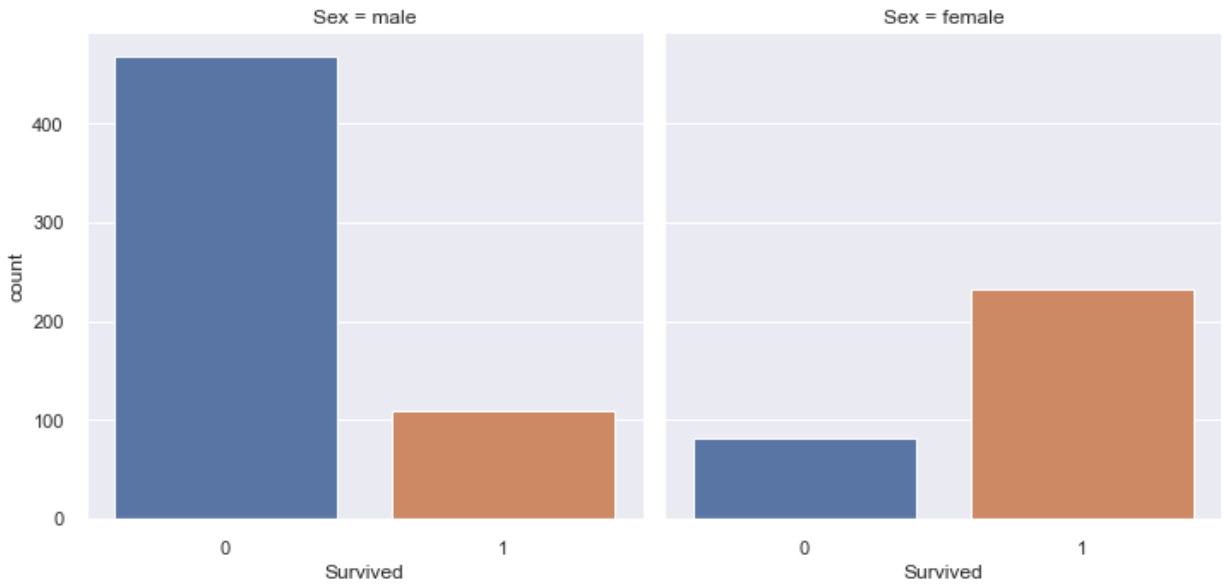


In [5]:
```
df_plot = train.groupby(['Pclass', 'Survived']).size().reset_index().pivot(columns='Pclass', index='Survived', values=0)
df_plot.plot(kind='bar', stacked=True)
```

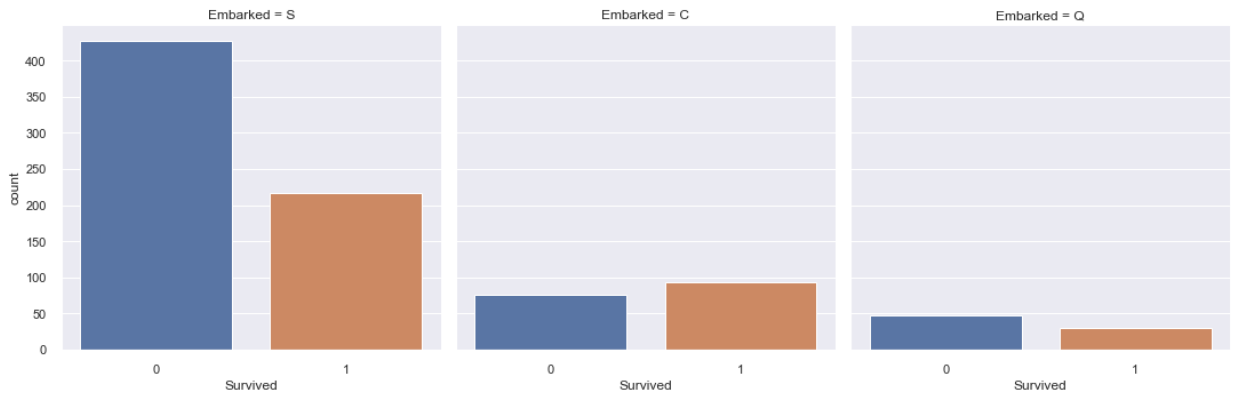
Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x13efdb6d8>



In [511]:
```
import seaborn as sns
from sklearn import tree
from sklearn.metrics import accuracy_score

# Figures inline and set visualization style
%matplotlib inline
sns.set()
sns.catplot(x='Survived', col='Sex', kind='count', data=train);
```



In [512]:
```
sns.catplot(x='Survived', col='Embarked', kind='count', data=train);
```



In [ ]:

In [ ]:

In [ ]:

In [520]:
```
train[['Pclass', 'Survived']].groupby(['Pclass'], as_index=False).mean().sort_values(by='Survived', ascending=False)
```

Out[520]:

	Pclass	Survived
0	1	0.629630
1	2	0.472826
2	3	0.242363

In [ ]:

In [521]:
```
train.shape, test.shape
train = pd.read_csv("train.csv")
```

In [8]:
```
train.describe(include = 'all')
```

Out[8]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Par
count	891.000000	891.000000	891.000000	891	891	714.000000	891.000000	891.0000
unique	NaN	NaN	NaN	891	2	NaN	NaN	N
top	NaN	NaN	NaN	Betros, Mr. Tannous	male	NaN	NaN	N
freq	NaN	NaN	NaN	1	577	NaN	NaN	N
mean	446.000000	0.383838	2.308642	NaN	NaN	29.699118	0.523008	0.3815
std	257.353842	0.486592	0.836071	NaN	NaN	14.526497	1.102743	0.8060
min	1.000000	0.000000	1.000000	NaN	NaN	0.420000	0.000000	0.0000
25%	223.500000	0.000000	2.000000	NaN	NaN	20.125000	0.000000	0.0000
50%	446.000000	0.000000	3.000000	NaN	NaN	28.000000	0.000000	0.0000
75%	668.500000	1.000000	3.000000	NaN	NaN	38.000000	1.000000	0.0000
max	891.000000	1.000000	3.000000	NaN	NaN	80.000000	8.000000	6.0000

In [9]:
```
train.shape, test.shape
```

Out[9]:
```
((891, 12), (418, 11))
```

In [10]:
```
df=pd.DataFrame(train)
df['Family_Size']=df['SibSp']+df['Parch']+1
```

In [11]:
```
train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')
```

In [12]:
```
X_train=train.drop(['Survived'], axis=1)

all_data = pd.concat((X_train, test))
```

In [13]:
```
df=pd.DataFrame(all_data)
print(df)
```

```
     PassengerId  Pclass
Name  \
0              1       3                            Braund, Mr. Owen
Harris
1              2       1  Cumings, Mrs. John Bradley (Florence Brigg
s Th...
```

```
2              3       3                                Heikkinen, Miss
. Laina
3              4       1       Futrelle, Mrs. Jacques Heath (Lily Ma
y Peel)
4              5       3                              Allen, Mr. Willia
m Henry
..           ...     ...
...
413         1305       3                                  Spector, Mr
. Woolf
414         1306       1                       Oliva y Ocana, Dona.
Fermina
415         1307       3                       Saether, Mr. Simon Si
vertsen
416         1308       3                                Ware, Mr. Fr
ederick
417         1309       3                            Peter, Master. Mi
chael J

        Sex   Age  SibSp  Parch              Ticket      Fare Cabin
Embarked
0      male  22.0      1      0           A/5 21171    7.2500   NaN
S
1    female  38.0      1      0            PC 17599   71.2833   C85
C
2    female  26.0      0      0    STON/O2. 3101282    7.9250   NaN
S
3    female  35.0      1      0              113803   53.1000  C123
S
4      male  35.0      0      0              373450    8.0500   NaN
S
..      ...   ...    ...    ...                 ...       ...   ...
...
413    male   NaN      0      0           A.5. 3236    8.0500   NaN
S
414  female  39.0      0      0            PC 17758  108.9000  C105
C
415    male  38.5      0      0  SOTON/O.Q. 3101262    7.2500   NaN
S
416    male   NaN      0      0              359309    8.0500   NaN
S
417    male   NaN      1      1                2668   22.3583   NaN
C

[1309 rows x 11 columns]
```

In [99]:
```
# 2b. Transform Data

df=pd.DataFrame(all_data)

# Impute Age with median age:
df['Age'].fillna(df['Age'].median(), inplace = True)

# Complete embarked with mode:
df['Embarked'].fillna(df['Embarked'].mode()[0], inplace = True)

#Turn cabin number into Deck:
df.Cabin = df.Cabin.fillna('Unknown')
df['Deck'] = train['Cabin'].apply(lambda x: x[0] if not pd.isna(x) els
e 'U')

# Creating new family_size column
df['Family_Size']=df['SibSp']+df['Parch']+1

# Creating new travel alone column
df['TravelAlone']=np.where((df['Family_Size'])>1, 0, 1)

# Creating new fare per passenger column
df['Fare_Per_Passenger']=df['Fare']/df['Family_Size']

# Logistic Regression works better with binary or categorical variable
s, so convert family size dummy variables
df=pd.get_dummies(df, columns=["Pclass","Sex","Embarked","Deck","Famil
y_Size"])

# Drop variables which don't are useful predictors or are replicated e
lsewhere:
df.drop(['PassengerId','Name','Ticket', 'Cabin', 'Fare', 'SibSp', 'Par
ch' ], axis=1, inplace = True)

# Impute missing Fare_Per_Passenger
df = df.fillna(all_data.mean())

all_data_transformed=df

display(df)
```

	Age	TravelAlone	Fare_Per_Passenger	Pclass_1	Pclass_2	Pclass_3	Sex_female	Sex_m
0	22.0	0	3.625000	0	0	1	0	
1	38.0	0	35.641650	1	0	0	1	
2	26.0	1	7.925000	0	0	1	1	
3	35.0	0	26.550000	1	0	0	1	
4	35.0	1	8.050000	0	0	1	0	
...	...	...	...	...	...	...	...	
413	28.0	1	8.050000	0	0	1	0	
414	39.0	1	108.900000	1	0	0	1	
415	38.5	1	7.250000	0	0	1	0	
416	28.0	1	8.050000	0	0	1	0	
417	28.0	0	7.452767	0	0	1	0	

1309 rows × 29 columns

In [100]:
```
df = df.loc[:,'Age':'Family_Size_11']

X_train = df[:train.shape[0]]
X_test = df[train.shape[0]:]
y = train.Survived

print(X_train)
```

```
      Age  TravelAlone  Fare_Per_Passenger  Pclass_1  Pclass_2  Pcla
ss_3  \
0    22.0            0             3.62500         0         0
1
1    38.0            0            35.64165         1         0
0
2    26.0            1             7.92500         0         0
1
3    35.0            0            26.55000         1         0
0
4    35.0            1             8.05000         0         0
1
..    ...          ...                 ...       ...       ...
...
886  27.0            1            13.00000         0         1
0
887  19.0            1            30.00000         1         0
0
888  28.0            0             5.86250         0         0
```

```
1
889  26.0            1            30.00000         1         0
0
890  32.0            1             7.75000         0         0
1

     Sex_female  Sex_male  Embarked_C  Embarked_Q  ...  Deck_U  Fami
ly_Size_1  \
0             0         1           0           0  ...       1
0
1             1         0           1           0  ...       0
0
2             1         0           0           0  ...       1
1
3             1         0           0           0  ...       0
0
4             0         1           0           0  ...       1
1
..          ...       ...         ...         ...  ...     ...
...
886           0         1           0           0  ...       1
1
887           1         0           0           0  ...       0
1
888           1         0           0           0  ...       1
0
889           0         1           1           0  ...       0
1
890           0         1           0           1  ...       1
1

     Family_Size_2  Family_Size_3  Family_Size_4  Family_Size_5  \
0                1              0              0              0
1                1              0              0              0
2                0              0              0              0
3                1              0              0              0
4                0              0              0              0
..             ...            ...            ...            ...
886              0              0              0              0
887              0              0              0              0
888              0              0              1              0
889              0              0              0              0
890              0              0              0              0

     Family_Size_6  Family_Size_7  Family_Size_8  Family_Size_11
0                0              0              0               0
1                0              0              0               0
2                0              0              0               0
3                0              0              0               0
4                0              0              0               0
```

```
..          ...          ...          ...          ...
886           0            0            0            0
887           0            0            0            0
888           0            0            0            0
889           0            0            0            0
890           0            0            0            0

[891 rows x 29 columns]
```

In [101]:
```
X_train.shape, y.shape, X_test.shape
```

Out[101]:
```
((891, 29), (891,), (418, 29))
```

In [102]:
```

```

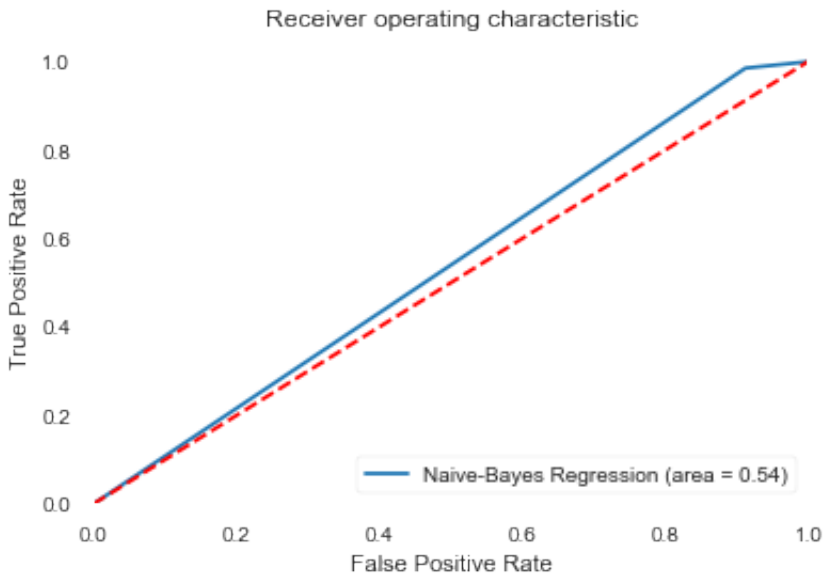
```
43.21
```

In [163]:
```
nb = GaussianNB()
nb_scores = cross_val_score(nb, X_train, y, cv=5, scoring = "accuracy"
)
nb.fit(X_train, y)
nb_X_pred = nb.predict(X_train)
nb_Y_pred = nb.predict(X_test)
print("Accuracy of Naive Bayes:", nb_scores)
print("Mean of NB:", nb_scores.mean())
print("Standard Deviation of NB:", nb_scores.std())
```

```
Accuracy of Naive Bayes: [0.44692737 0.71348315 0.41573034 0.4101123
6 0.42134831]
Mean of NB: 0.4815203063210093
Standard Deviation of NB: 0.11666320197101959
```

In [164]:
```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve

print("Area Under the Curve:", roc_auc_score(y,nb_X_pred))
nb_roc_auc = roc_auc_score(y, nb_X_pred)
fpr, tpr, thresholds = roc_curve(y, nb_X_pred)
plt.figure()
plt.plot(fpr, tpr, label='Naive-Bayes Regression (area = %0.2f)' % nb_
roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
```

```
Area Under the Curve: 0.5364059054740677
```



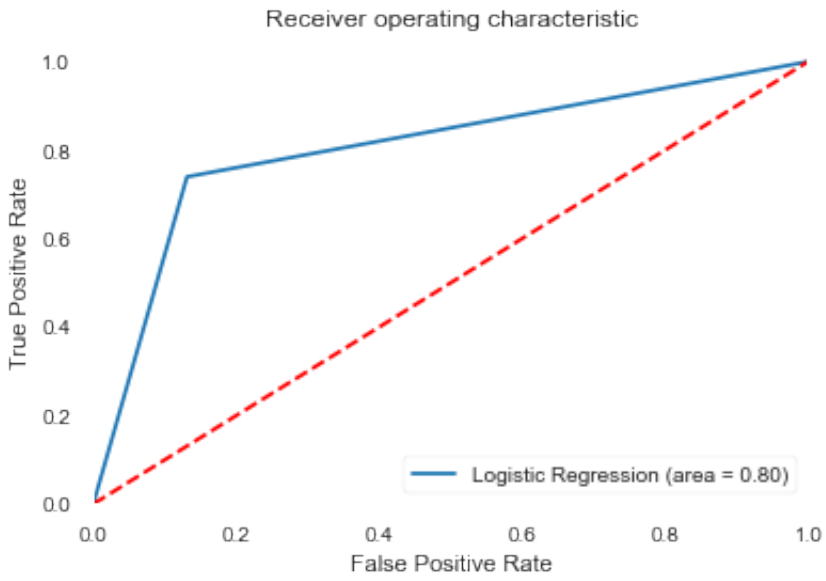
In [123]:
```
from sklearn.model_selection import cross_val_score
log_r = LogisticRegression()
log_scores = cross_val_score(log_r, X_train, y, cv=5, scoring = "accur
acy")
log_r.fit(X_train, y)
log_X_pred = log_r.predict(X_train)
log_Y_pred = log_r.predict(X_test)
print("Accuracy of Logistic Regression:", log_scores)
print("Mean of LR:", log_scores.mean())
print("Standard Deviation of LR:", log_scores.std())
```

```
Accuracy of Logistic Regression: [0.81005587 0.81460674 0.80898876 0
.80337079 0.8258427 ]
Mean of LR: 0.8125729709371665
Standard Deviation of LR: 0.007537697231272079

/Users/avadhani/anaconda3/lib/python3.7/site-packages/sklearn/linear
_model/_logistic.py:764: ConvergenceWarning: lbfgs failed to converg
e (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as sh
own in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver option
s:
    https://scikit-learn.org/stable/modules/linear_model.html#logist
ic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/avadhani/anaconda3/lib/python3.7/site-packages/sklearn/linear
_model/_logistic.py:764: ConvergenceWarning: lbfgs failed to converg
e (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as sh
own in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver option
s:
    https://scikit-learn.org/stable/modules/linear_model.html#logist
ic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/avadhani/anaconda3/lib/python3.7/site-packages/sklearn/linear
_model/_logistic.py:764: ConvergenceWarning: lbfgs failed to converg
e (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as sh
own in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver option
s:
    https://scikit-learn.org/stable/modules/linear_model.html#logist
ic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/avadhani/anaconda3/lib/python3.7/site-packages/sklearn/linear
_model/_logistic.py:764: ConvergenceWarning: lbfgs failed to converg
e (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as sh
```

```
own in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver option
s:
    https://scikit-learn.org/stable/modules/linear_model.html#logist
ic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/avadhani/anaconda3/lib/python3.7/site-packages/sklearn/linear
_model/_logistic.py:764: ConvergenceWarning: lbfgs failed to converg
e (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as sh
own in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver option
s:
    https://scikit-learn.org/stable/modules/linear_model.html#logist
ic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
/Users/avadhani/anaconda3/lib/python3.7/site-packages/sklearn/linear
_model/_logistic.py:764: ConvergenceWarning: lbfgs failed to converg
e (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as sh
own in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver option
s:
    https://scikit-learn.org/stable/modules/linear_model.html#logist
ic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

In [149]:
```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve

print("Area Under the Curve:", roc_auc_score(y,log_X_pred))
log_roc_auc = roc_auc_score(y, log_X_pred)
fpr, tpr, thresholds = roc_curve(y, log_X_pred)
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % log_ro
c_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
```

```
Area Under the Curve: 0.8043092704438694
```



In [150]:
```
log_submission = pd.DataFrame({
        "PassengerId": test["PassengerId"],
        "Survived": log_Y_pred
    })
log_submission.to_csv('log_submission.csv', index=False)
```

In [158]:
```
nb_submission = pd.DataFrame({
        "PassengerId": test["PassengerId"],
        "Survived": nb_Y_pred
    })
nb_submission.to_csv('nb_submission.csv', index=False)
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

log_submission

PassengerId	Survived
892	0
893	0
894	0
895	0
896	1
897	0
898	1
899	0
900	1
901	0
902	0
903	0
904	1
905	0
906	1
907	1
908	0
909	0
910	1
911	1
912	0
913	0
914	1
915	1
916	0
917	0
918	1
919	0
920	0

921	0
922	0
923	1
924	1
925	1
926	0
927	0
928	1
929	1
930	0
931	0
932	0
933	0
934	0
935	1
936	1
937	0
938	0
939	0
940	1
941	1
942	0
943	0
944	1
945	1
946	0
947	0
948	0
949	0
950	0
951	1
952	0

953	0
954	0
955	1
956	0
957	1
958	1
959	0
960	0
961	0
962	1
963	0
964	1
965	0
966	1
967	1
968	0
969	1
970	0
971	1
972	0
973	0
974	0
975	0
976	0
977	0
978	1
979	1
980	1
981	0
982	1
983	0
984	1

985	0
986	0
987	0
988	1
989	0
990	1
991	0
992	1
993	0
994	0
995	0
996	1
997	0
998	0
999	0
1000	0
1001	0
1002	0
1003	1
1004	1
1005	1
1006	1
1007	0
1008	0
1009	1
1010	1
1011	1
1012	1
1013	0
1014	1
1015	0
1016	0

1017	1
1018	0
1019	1
1020	0
1021	0
1022	0
1023	0
1024	0
1025	0
1026	0
1027	0
1028	0
1029	0
1030	1
1031	0
1032	0
1033	1
1034	0
1035	0
1036	0
1037	0
1038	0
1039	0
1040	0
1041	0
1042	1
1043	0
1044	0
1045	1
1046	0
1047	0
1048	1

1049	1
1050	0
1051	1
1052	1
1053	0
1054	1
1055	0
1056	0
1057	1
1058	1
1059	0
1060	1
1061	1
1062	0
1063	0
1064	0
1065	0
1066	0
1067	1
1068	1
1069	0
1070	1
1071	1
1072	0
1073	0
1074	1
1075	0
1076	1
1077	0
1078	1
1079	0
1080	0

1081	0
1082	0
1083	0
1084	0
1085	0
1086	1
1087	0
1088	1
1089	1
1090	0
1091	1
1092	1
1093	0
1094	0
1095	1
1096	0
1097	0
1098	1
1099	0
1100	1
1101	0
1102	0
1103	0
1104	0
1105	1
1106	0
1107	0
1108	1
1109	0
1110	1
1111	0
1112	1

1113	0
1114	1
1115	0
1116	1
1117	1
1118	0
1119	1
1120	0
1121	0
1122	0
1123	1
1124	0
1125	0
1126	0
1127	0
1128	0
1129	0
1130	1
1131	1
1132	1
1133	1
1134	0
1135	0
1136	0
1137	0
1138	1
1139	0
1140	1
1141	1
1142	1
1143	0
1144	1

1145	0
1146	0
1147	0
1148	0
1149	0
1150	1
1151	0
1152	0
1153	0
1154	1
1155	1
1156	0
1157	0
1158	0
1159	0
1160	1
1161	0
1162	0
1163	0
1164	1
1165	1
1166	0
1167	1
1168	0
1169	0
1170	0
1171	0
1172	1
1173	0
1174	1
1175	1
1176	1

1177	0
1178	0
1179	0
1180	0
1181	0
1182	0
1183	1
1184	0
1185	0
1186	0
1187	0
1188	1
1189	0
1190	0
1191	0
1192	0
1193	0
1194	0
1195	0
1196	1
1197	1
1198	0
1199	0
1200	0
1201	1
1202	0
1203	0
1204	0
1205	1
1206	1
1207	1
1208	0

1209	0
1210	0
1211	1
1212	0
1213	0
1214	0
1215	0
1216	1
1217	0
1218	1
1219	1
1220	0
1221	0
1222	1
1223	0
1224	0
1225	1
1226	0
1227	0
1228	0
1229	0
1230	0
1231	0
1232	0
1233	0
1234	0
1235	1
1236	0
1237	1
1238	0
1239	1
1240	0

1241	1
1242	1
1243	0
1244	0
1245	0
1246	1
1247	0
1248	1
1249	0
1250	0
1251	1
1252	0
1253	1
1254	1
1255	0
1256	1
1257	0
1258	0
1259	1
1260	1
1261	0
1262	1
1263	1
1264	0
1265	0
1266	1
1267	1
1268	1
1269	0
1270	0
1271	0
1272	0

1273	0
1274	1
1275	1
1276	0
1277	1
1278	0
1279	0
1280	0
1281	0
1282	1
1283	1
1284	0
1285	0
1286	0
1287	1
1288	0
1289	1
1290	0
1291	0
1292	1
1293	0
1294	1
1295	0
1296	0
1297	0
1298	0
1299	0
1300	1
1301	1
1302	1
1303	1
1304	1

1305	0
1306	1
1307	0
1308	0
1309	0

nb_submission

PassengerId	Survived
892	1
893	1
894	1
895	1
896	1
897	1
898	1
899	1
900	1
901	1
902	1
903	1
904	1
905	1
906	1
907	1
908	1
909	1
910	1
911	1
912	1
913	1
914	1
915	1
916	0
917	1
918	1
919	1
920	1

921	1
922	1
923	1
924	1
925	1
926	1
927	1
928	1
929	1
930	1
931	1
932	1
933	1
934	1
935	1
936	1
937	1
938	1
939	1
940	1
941	1
942	1
943	1
944	1
945	0
946	1
947	0
948	1
949	1
950	1
951	1
952	1

953	1
954	1
955	1
956	0
957	1
958	1
959	1
960	1
961	0
962	1
963	1
964	1
965	1
966	1
967	1
968	1
969	1
970	1
971	1
972	1
973	1
974	1
975	1
976	1
977	1
978	1
979	1
980	1
981	1
982	1
983	1
984	1

985	1
986	1
987	1
988	1
989	1
990	1
991	1
992	1
993	1
994	1
995	1
996	1
997	1
998	1
999	1
1000	1
1001	1
1002	1
1003	1
1004	1
1005	1
1006	1
1007	1
1008	1
1009	1
1010	1
1011	1
1012	1
1013	1
1014	1
1015	1
1016	1

1017	1
1018	1
1019	1
1020	1
1021	1
1022	1
1023	1
1024	0
1025	1
1026	1
1027	1
1028	1
1029	1
1030	1
1031	0
1032	0
1033	1
1034	0
1035	1
1036	1
1037	1
1038	1
1039	1
1040	1
1041	1
1042	1
1043	1
1044	1
1045	1
1046	0
1047	1
1048	1

1049	1
1050	1
1051	1
1052	1
1053	1
1054	1
1055	1
1056	1
1057	1
1058	1
1059	0
1060	1
1061	1
1062	1
1063	1
1064	1
1065	1
1066	1
1067	1
1068	1
1069	1
1070	1
1071	1
1072	1
1073	1
1074	1
1075	1
1076	1
1077	1
1078	1
1079	1
1080	0

1081	1
1082	1
1083	1
1084	1
1085	1
1086	1
1087	1
1088	1
1089	1
1090	1
1091	1
1092	1
1093	1
1094	1
1095	1
1096	1
1097	1
1098	1
1099	1
1100	1
1101	1
1102	1
1103	1
1104	1
1105	1
1106	1
1107	1
1108	1
1109	1
1110	1
1111	1
1112	1

1113	1
1114	1
1115	1
1116	1
1117	1
1118	1
1119	1
1120	1
1121	1
1122	1
1123	1
1124	1
1125	1
1126	1
1127	1
1128	1
1129	1
1130	1
1131	1
1132	1
1133	1
1134	1
1135	1
1136	1
1137	1
1138	1
1139	1
1140	1
1141	1
1142	1
1143	1
1144	1

1145	1
1146	1
1147	1
1148	1
1149	1
1150	1
1151	1
1152	1
1153	1
1154	1
1155	1
1156	1
1157	1
1158	1
1159	1
1160	1
1161	1
1162	1
1163	1
1164	1
1165	1
1166	1
1167	1
1168	1
1169	1
1170	1
1171	1
1172	1
1173	1
1174	1
1175	1
1176	1

1177	1
1178	1
1179	1
1180	1
1181	1
1182	1
1183	1
1184	1
1185	1
1186	1
1187	1
1188	1
1189	1
1190	1
1191	1
1192	1
1193	1
1194	1
1195	1
1196	1
1197	1
1198	1
1199	1
1200	1
1201	1
1202	1
1203	1
1204	1
1205	1
1206	1
1207	1
1208	1

1209	1
1210	1
1211	1
1212	1
1213	1
1214	1
1215	1
1216	1
1217	1
1218	1
1219	1
1220	1
1221	1
1222	1
1223	1
1224	1
1225	1
1226	1
1227	1
1228	1
1229	1
1230	1
1231	0
1232	1
1233	1
1234	0
1235	1
1236	1
1237	1
1238	1
1239	1
1240	1

1241	1
1242	1
1243	1
1244	1
1245	1
1246	1
1247	1
1248	1
1249	1
1250	1
1251	1
1252	0
1253	1
1254	1
1255	1
1256	1
1257	0
1258	1
1259	1
1260	1
1261	1
1262	1
1263	1
1264	1
1265	1
1266	1
1267	1
1268	1
1269	1
1270	1
1271	0
1272	1

1273	1
1274	1
1275	1
1276	1
1277	1
1278	1
1279	1
1280	1
1281	0
1282	1
1283	1
1284	1
1285	1
1286	1
1287	1
1288	1
1289	1
1290	1
1291	1
1292	1
1293	1
1294	1
1295	1
1296	1
1297	1
1298	1
1299	1
1300	1
1301	1
1302	1
1303	1
1304	1

1305	1
1306	1
1307	1
1308	1
1309	1