GOAL: To Predict Survival of a person in Titanic-like Shipwreck.

LIBRARIES

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.manifold import TSNE
from sklearn.decomposition import PCA
from sklearn.feature_selection import SelectFromModel
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
```

In [2]:
```
train_data= pd.read_csv("train.csv")
train_target=train_data['Survived']
test_data= pd.read_csv("test.csv")
test_target=pd.read_csv("gender_submission.csv")['Survived']
```

In [3]:
```
train_data=train_data.drop(columns=['Survived','Name','PassengerId','Ticket'])
test_data=test_data.drop(columns=['Name','PassengerId','Ticket'])
```

In [4]:
```
train_data.head()
```

Out[4]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
0	3	male	22.0	1	0	7.2500	NaN	S
1	1	female	38.0	1	0	71.2833	C85	C
2	3	female	26.0	0	0	7.9250	NaN	S
3	1	female	35.0	1	0	53.1000	C123	S
4	3	male	35.0	0	0	8.0500	NaN	S

In [5]:
```
test_data.head()
```

Out[5]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
0	3	male	34.5	0	0	7.8292	NaN	Q
1	3	female	47.0	1	0	7.0000	NaN	S
2	2	male	62.0	0	0	9.6875	NaN	Q
3	3	male	27.0	0	0	8.6625	NaN	S
4	3	female	22.0	1	1	12.2875	NaN	S

In [6]:
```
train_target.head()
```

Out[6]:
```
0    0
1    1
2    1
3    1
4    0
Name: Survived, dtype: int64
```

In [7]:
```
test_target.head()
```

Out[7]:
```
0    0
1    1
2    0
3    0
4    1
Name: Survived, dtype: int64
```

Data Cleaning and Value Counts

In [8]:
```python
for col in train_data.columns:
    print(
        'TRAIN DATA\n',
        train_data[col].value_counts(),
        '\n----------------------------'
        'TEST DATA\n',
        test_data[col].value_counts(),
        '\n============================'
    )
```

```
TRAIN DATA
 3    491
1    216
2    184
Name: Pclass, dtype: int64
----------------------------TEST DATA
 3    218
1    107
2     93
Name: Pclass, dtype: int64
============================
TRAIN DATA
 male      577
female    314
Name: Sex, dtype: int64
----------------------------TEST DATA
 male      266
female    152
Name: Sex, dtype: int64
============================
TRAIN DATA
 24.00    30
22.00    27
18.00    26
19.00    25
30.00    25
28.00    25
21.00    24
25.00    23
36.00    22
29.00    20
32.00    18
27.00    18
35.00    18
26.00    18
16.00    17
31.00    17
20.00    15
33.00    15
23.00    15
34.00    15
39.00    14
17.00    13
42.00    13
40.00    13
45.00    12
38.00    11
50.00    10
2.00     10
4.00     10
47.00     9
         ..
71.00     2
59.00     2
63.00     2
0.83      2
30.50     2
70.00     2
57.00     2
0.75      2
13.00     2
10.00     2
64.00     2
40.50     2
32.50     2
45.50     2
20.50     1
24.50     1
0.67      1
14.50     1
0.92      1
74.00     1
34.50     1
80.00     1
12.00     1
36.50     1
53.00     1
55.50     1
70.50     1
66.00     1
23.50     1
0.42      1
Name: Age, Length: 88, dtype: int64
----------------------------TEST DATA
 24.00    17
21.00    17
22.00    16
30.00    15
18.00    13
27.00    12
26.00    12
25.00    11
23.00    11
```

```
29.00    10
36.00     9
45.00     9
20.00     8
17.00     7
28.00     7
32.00     6
31.00     6
55.00     6
33.00     6
39.00     6
35.00     5
41.00     5
47.00     5
40.00     5
50.00     5
42.00     5
48.00     5
19.00     4
43.00     4
1.00      3
         ..
8.00      2
63.00     2
14.00     2
22.50     1
62.00     1
0.83      1
67.00     1
28.50     1
0.33      1
0.17      1
38.50     1
3.00      1
51.00     1
5.00      1
44.00     1
14.50     1
59.00     1
58.00     1
0.75      1
0.92      1
36.50     1
40.50     1
11.50     1
34.00     1
15.00     1
7.00      1
60.50     1
26.50     1
76.00     1
34.50     1
Name: Age, Length: 79, dtype: int64
=============================
TRAIN DATA
 0    608
1    209
2     28
4     18
3     16
8      7
5      5
Name: SibSp, dtype: int64
-----------------------------TEST DATA
 0    283
1    110
2     14
4      4
3      4
8      2
5      1
Name: SibSp, dtype: int64
=============================
TRAIN DATA
 0    678
1    118
2     80
5      5
3      5
4      4
6      1
Name: Parch, dtype: int64
-----------------------------TEST DATA
 0    324
1     52
2     33
3      3
9      2
4      2
6      1
5      1
Name: Parch, dtype: int64
=============================
TRAIN DATA
 8.0500      43
13.0000     42
7.8958      38
7.7500      34
26.0000     31
10.5000     24
7.9250      18
7.7750      16
26.5500     15
0.0000      15
```

```
7.2292      15
7.8542      13
8.6625      13
7.2500      13
7.2250      12
16.1000      9
9.5000       9
24.1500      8
15.5000      8
56.4958      7
52.0000      7
14.5000      7
14.4542      7
69.5500      7
7.0500       7
31.2750      7
46.9000      6
30.0000      6
7.7958       6
39.6875      6
            ..
7.1417       1
42.4000      1
211.5000     1
12.2750      1
61.1750      1
8.4333       1
51.4792      1
7.8875       1
8.6833       1
7.5208       1
34.6542      1
28.7125      1
25.5875      1
7.7292       1
12.2875      1
8.6542       1
8.7125       1
61.3792      1
6.9500       1
9.8417       1
8.3000       1
13.7917      1
9.4750       1
13.4167      1
26.3875      1
8.4583       1
9.8375       1
8.3625       1
14.1083      1
17.4000      1
Name: Fare, Length: 248, dtype: int64
-----------------------------TEST DATA
 7.7500      21
26.0000     19
8.0500      17
13.0000     17
7.8958      11
10.5000     11
7.7750      10
7.2292       9
7.2250       9
8.6625       8
7.8542       8
21.0000      8
26.5500      7
7.8792       6
27.7208      6
7.2500       5
7.9250       5
262.3750     5
211.5000     4
69.5500      4
14.5000      4
7.5500       4
7.7958       4
15.2458      4
55.4417      3
31.3875      3
31.5000      3
14.4542      3
9.5000       3
221.7792     3
            ..
50.4958      1
39.4000      1
34.3750      1
7.7208       1
7.8500       1
76.2917      1
7.7250       1
9.2250       1
39.6875      1
75.2500      1
13.8625      1
6.9500       1
61.1750      1
78.8500      1
20.2125      1
247.5208     1
7.5750       1
28.5375      1
227.5250     1
108.9000     1
```

```
6.4958       1
7.6292       1
47.1000      1
22.3583      1
17.4000      1
9.3250       1
14.4583      1
15.0333      1
25.4667      1
21.0750      1
Name: Fare, Length: 169, dtype: int64
=============================
TRAIN DATA
 B96 B98            4
G6                 4
C23 C25 C27        4
F33                3
F2                 3
E101               3
D                  3
C22 C26            3
B49                2
B57 B59 B63 B66    2
E44                2
D17                2
E121               2
E25                2
B18                2
B51 B53 B55        2
E24                2
B77                2
D36                2
C52                2
C2                 2
B58 B60            2
D35                2
C78                2
B20                2
C92                2
C68                2
C93                2
B35                2
C123               2
                  ..
C110               1
D45                1
C90                1
C62 C64            1
C49                1
A34                1
C85                1
C103               1
C87                1
D6                 1
B19                1
B37                1
D7                 1
B101               1
B73                1
C50                1
D50                1
C148               1
A6                 1
C111               1
B71                1
D56                1
C99                1
E77                1
D9                 1
E34                1
E46                1
D30                1
B80                1
D48                1
Name: Cabin, Length: 147, dtype: int64
-----------------------------TEST DATA
 B57 B59 B63 B66    3
C31                2
C78                2
A34                2
B45                2
C6                 2
C116               2
C55 C57            2
C80                2
E34                2
C101               2
C23 C25 C27        2
C89                2
F4                 2
B11                1
B61                1
E31                1
C28                1
E39 E41            1
B26                1
D19                1
B51 B53 B55        1
B58 B60            1
C106               1
C54                1
F2                 1
E46                1
D38                1
```

```
C85                 1
A18                 1
                   ..
A29                 1
E52                 1
B78                 1
D22                 1
B41                 1
B71                 1
C130                1
F                   1
D40                 1
B36                 1
C46                 1
D21                 1
D10 D12             1
C51                 1
F33                 1
C86                 1
G6                  1
C132                1
D15                 1
C32                 1
A9                  1
E50                 1
C22 C26             1
B52 B54 B56         1
A21                 1
C39                 1
C62 C64             1
D37                 1
C7                  1
A11                 1
Name: Cabin, Length: 76, dtype: int64
=============================
TRAIN DATA
 S    644
C    168
Q     77
Name: Embarked, dtype: int64
-----------------------------TEST DATA
 S    270
C    102
Q     46
Name: Embarked, dtype: int64
=============================
```

In [9]:

```
OC=[]
print("OMITTED VALUES in TRAIN DATA\n========================")
for col in train_data.columns:
    if train_data[col].isna().value_counts().shape==(2,):
        OC.append(col)
        print(
            'TRAIN DATA\n',
            train_data[col].isna().value_counts(),
            '\n-----------------------------'

    )

print('Ommitted Value-Columns:\n',
     OC,
     '\nAll Columns:\n',
     train_data.columns)
```

```
OMITTED VALUES in TRAIN DATA
========================
TRAIN DATA
 False    714
True     177
Name: Age, dtype: int64
-----------------------------
TRAIN DATA
 True     687
False    204
Name: Cabin, dtype: int64
-----------------------------
TRAIN DATA
 False    889
True       2
Name: Embarked, dtype: int64
-----------------------------
Ommitted Value-Columns:
 ['Age', 'Cabin', 'Embarked']
All Columns:
 Index(['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Cabin', 'Embarked'], dtype='object')
```

In [10]:
```
OC=[]
print("OMITTED VALUES in TEST DATA\n========================")
for col in test_data.columns:
    if test_data[col].isna().value_counts().shape==(2,):
        OC.append(col)
        print(
            'TEST DATA\n',
            test_data[col].isna().value_counts(),
            '\n------------------------'
    )

print('Ommitted Value-Columns:\n',
     OC,
     '\nAll Columns:\n',
     test_data.columns)
```

```
OMITTED VALUES in TEST DATA
========================
TEST DATA
 False    332
True      86
Name: Age, dtype: int64
------------------------
TEST DATA
 False    417
True       1
Name: Fare, dtype: int64
------------------------
TEST DATA
 True     327
False     91
Name: Cabin, dtype: int64
------------------------
Ommitted Value-Columns:
 ['Age', 'Fare', 'Cabin']
All Columns:
 Index(['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Cabin', 'Embarked'], dtype='object')
```

In [11]:
```
train_data.Age=train_data.Age.fillna(method='ffill')
train_data.Cabin=train_data.Cabin.fillna('B45')
train_data.Embarked=train_data.Embarked.fillna(method='ffill')

test_data.Age=test_data.Age.fillna(method='ffill')
test_data.Fare=test_data.Fare.fillna(method='ffill')
test_data.Cabin=test_data.Cabin.fillna('B45')
```

In [12]:
```
OC=[]
print("OMITTED VALUES in TRAIN DATA\n========================")
for col in train_data.columns:
    if train_data[col].isna().value_counts().shape==(2,):
        OC.append(col)
        print(
            'TRAIN DATA\n',
            train_data[col].isna().value_counts(),
            '\n-----------------------------'

    )

print('Ommitted Value-Columns:\n',
     OC,
     '\nAll Columns:\n',
     train_data.columns)
```

```
OMITTED VALUES in TRAIN DATA
========================
Ommitted Value-Columns:
 []
All Columns:
 Index(['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Cabin', 'Embarked'], dtype='object')
```

In [13]:
```
OC=[]
print("OMITTED VALUES in TEST DATA\n========================")
for col in test_data.columns:
    if test_data[col].isna().value_counts().shape==(2,):
        OC.append(col)
        print(
            'TEST DATA\n',
            test_data[col].isna().value_counts(),
            '\n------------------------'
    )

print('Ommitted Value-Columns:\n',
     OC,
     '\nAll Columns:\n',
     test_data.columns)
```

```
OMITTED VALUES in TEST DATA
========================
Ommitted Value-Columns:
 []
All Columns:
 Index(['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Cabin', 'Embarked'], dtype='object')
```

In [14]:
```
train_target.value_counts()
```

Out[14]:
```
0    549
1    342
Name: Survived, dtype: int64
```

In [15]:
```
test_target.value_counts()
```

Out[15]:
```
0    266
1    152
Name: Survived, dtype: int64
```

In [16]:
```
train_data.head()
```

Out[16]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
0	3	male	22.0	1	0	7.2500	B45	S
1	1	female	38.0	1	0	71.2833	C85	C
2	3	female	26.0	0	0	7.9250	B45	S
3	1	female	35.0	1	0	53.1000	C123	S
4	3	male	35.0	0	0	8.0500	B45	S

In [17]:
```
test_data.head()
```

Out[17]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
0	3	male	34.5	0	0	7.8292	B45	Q
1	3	female	47.0	1	0	7.0000	B45	S
2	2	male	62.0	0	0	9.6875	B45	Q
3	3	male	27.0	0	0	8.6625	B45	S
4	3	female	22.0	1	1	12.2875	B45	S

In [18]:
```
categorical=['Pclass','Sex','SibSp','Parch','Cabin','Embarked']
```

In [19]:
```
print(train_data.info(),
      test_data.info()
     )
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 8 columns):
Pclass      891 non-null int64
Sex         891 non-null object
Age         891 non-null float64
SibSp       891 non-null int64
Parch       891 non-null int64
Fare        891 non-null float64
Cabin       891 non-null object
Embarked    891 non-null object
dtypes: float64(2), int64(3), object(3)
memory usage: 55.8+ KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 8 columns):
Pclass      418 non-null int64
Sex         418 non-null object
Age         418 non-null float64
SibSp       418 non-null int64
Parch       418 non-null int64
Fare        418 non-null float64
Cabin       418 non-null object
Embarked    418 non-null object
dtypes: float64(2), int64(3), object(3)
memory usage: 26.2+ KB
None None
```

In [20]:
```
for col in categorical:
    train_data[col]=train_data[col].astype(str)
    test_data[col]=test_data[col].astype(str)
```

In [21]:
```
print(train_data.info(),
      test_data.info()
     )
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 8 columns):
Pclass      891 non-null object
Sex         891 non-null object
Age         891 non-null float64
SibSp       891 non-null object
Parch       891 non-null object
Fare        891 non-null float64
Cabin       891 non-null object
Embarked    891 non-null object
dtypes: float64(2), object(6)
memory usage: 55.8+ KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 8 columns):
Pclass      418 non-null object
Sex         418 non-null object
Age         418 non-null float64
SibSp       418 non-null object
Parch       418 non-null object
Fare        418 non-null float64
Cabin       418 non-null object
Embarked    418 non-null object
dtypes: float64(2), object(6)
memory usage: 26.2+ KB
None None
```

In [22]:
```
data=pd.concat([train_data,test_data])
```

In [23]:
```
data.head()
```

Out[23]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
0	3	male	22.0	1	0	7.2500	B45	S
1	1	female	38.0	1	0	71.2833	C85	C
2	3	female	26.0	0	0	7.9250	B45	S
3	1	female	35.0	1	0	53.1000	C123	S
4	3	male	35.0	0	0	8.0500	B45	S

In [24]:
```
data.shape
```

Out[24]:
```
(1309, 8)
```

In [25]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1309 entries, 0 to 417
Data columns (total 8 columns):
Pclass      1309 non-null object
Sex         1309 non-null object
Age         1309 non-null float64
SibSp       1309 non-null object
Parch       1309 non-null object
Fare        1309 non-null float64
Cabin       1309 non-null object
Embarked    1309 non-null object
dtypes: float64(2), object(6)
memory usage: 92.0+ KB
```

In [26]:
```
target=pd.concat([train_target,test_target])
target.values.shape
```

Out[26]:
```
(1309,)
```

VISUALISATION

In [27]:
```
survived=[]
dead=[]
for i in range(1309):
    if target.values[i]==1:
        survived.append(data.iloc[i].values)
    else:
        dead.append(data.iloc[i].values)
```

In [28]:
```
surviveddf=pd.DataFrame(survived, columns=data.columns)
```

In [29]:
```
deaddf=pd.DataFrame(dead, columns=data.columns)
```

In [30]:
```
surviveddf.head()
```

Out[30]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
0	1	female	38.0	1	0	71.2833	C85	C
1	3	female	26.0	0	0	7.9250	B45	S
2	1	female	35.0	1	0	53.1000	C123	S
3	3	female	27.0	0	2	11.1333	B45	S
4	2	female	14.0	1	0	30.0708	B45	C

In [31]:
```
surviveddf.Sex.value_counts()
```

Out[31]:
```
female    385
male      109
Name: Sex, dtype: int64
```

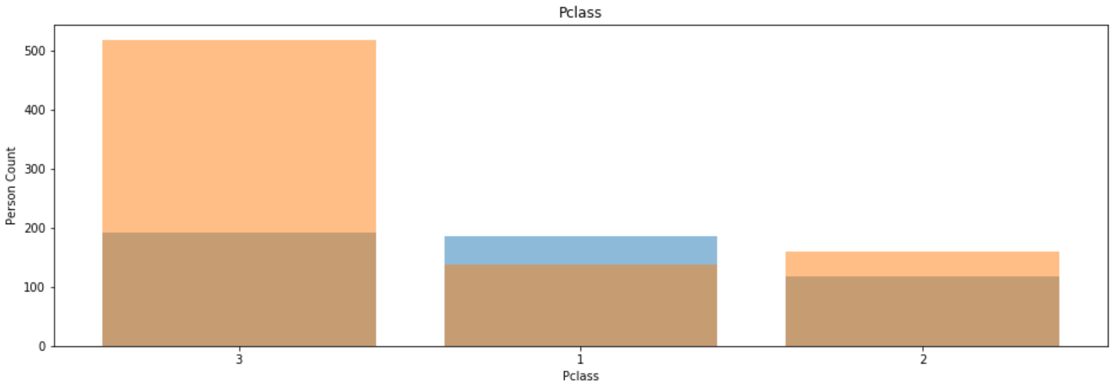
In [32]:
```
def vis(var):
    plt.figure(figsize=(16,5))
    plt.bar(surviveddf[var].value_counts().keys(), surviveddf[var].value_counts().values, alpha=0.5)
    plt.bar(deaddf[var].value_counts().keys(), deaddf[var].value_counts().values, alpha=0.5)
    plt.title(var)
    plt.xlabel(var)
    plt.ylabel("Person Count")
    plt.show()
    plt.figure(figsize=(2,2))
    plt.title("Survived")
    plt.pie(surviveddf[var].value_counts().values,labels=surviveddf[var].value_counts().keys())
    plt.show()
    plt.figure(figsize=(2,2))
    plt.title("Dead")
    plt.pie(deaddf[var].value_counts().values,labels=deaddf[var].value_counts().keys())
    plt.show()
```

In [33]:
```
data.columns
```

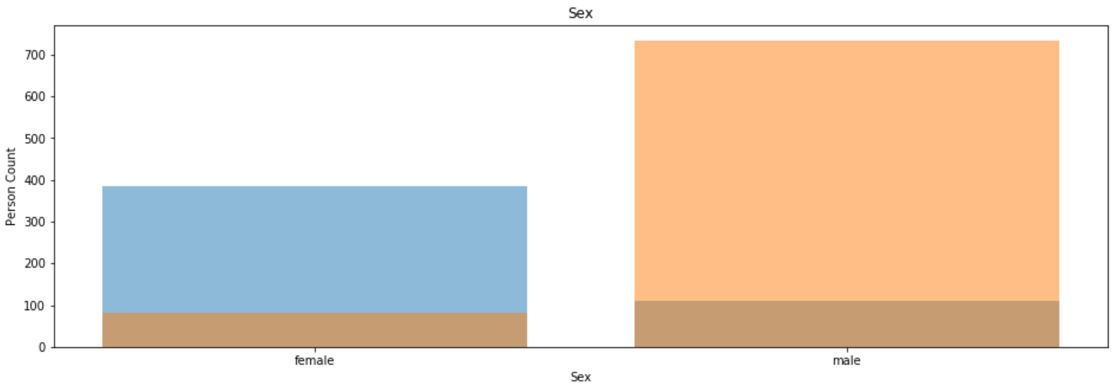
Out[33]:
```
Index(['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Cabin', 'Embarked'], dtype='object')
```

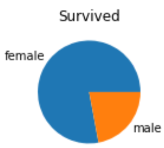
In [34]:
```
for var in data.columns:
    print(var, 'Vs Person Count Vs Survival')
    vis(var)
```

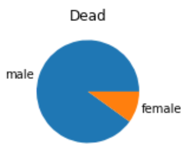
```
Pclass Vs Person Count Vs Survival
```



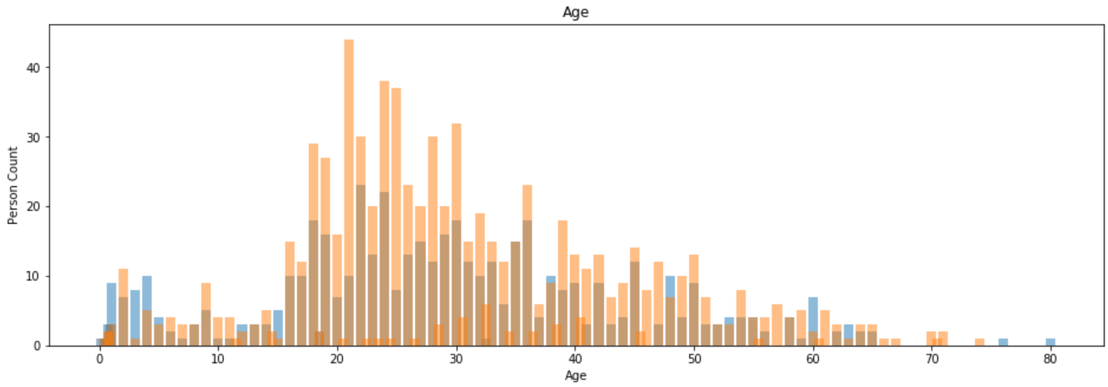

```
Sex Vs Person Count Vs Survival
```

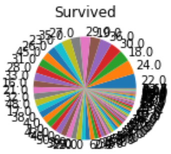


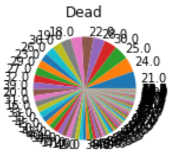




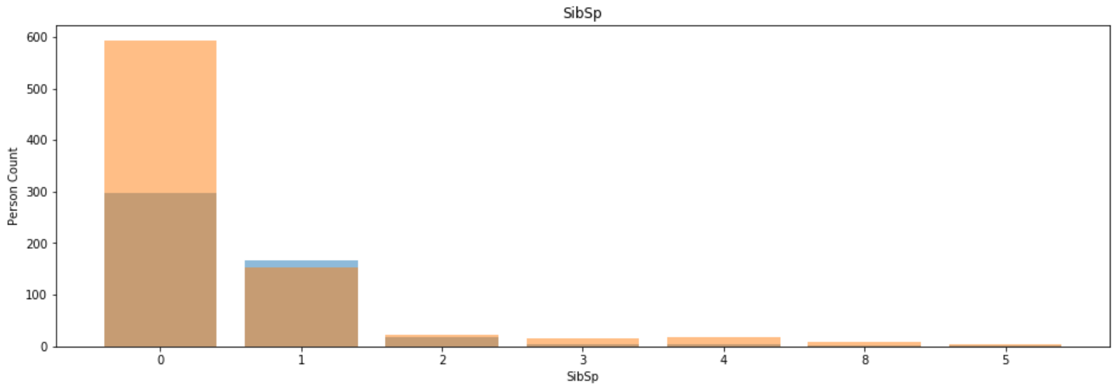

```
Age Vs Person Count Vs Survival
```

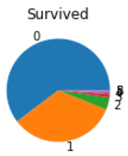


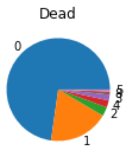




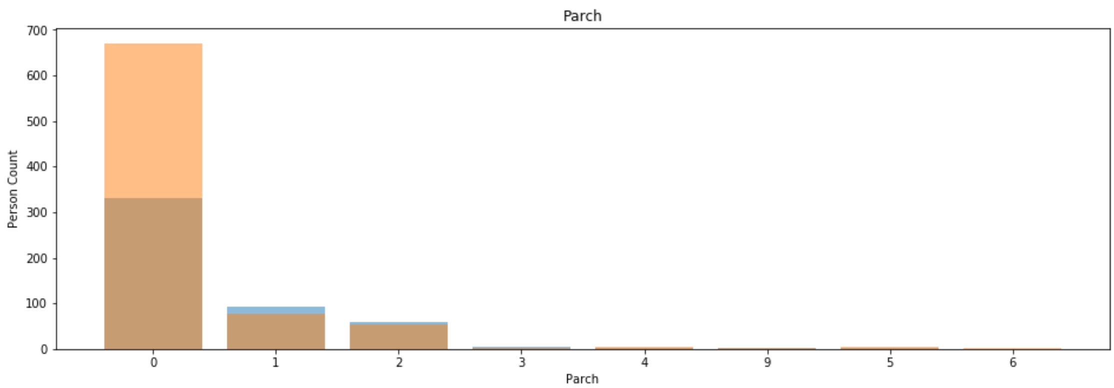
SibSp Vs Person Count Vs Survival

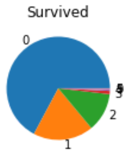


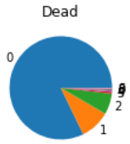




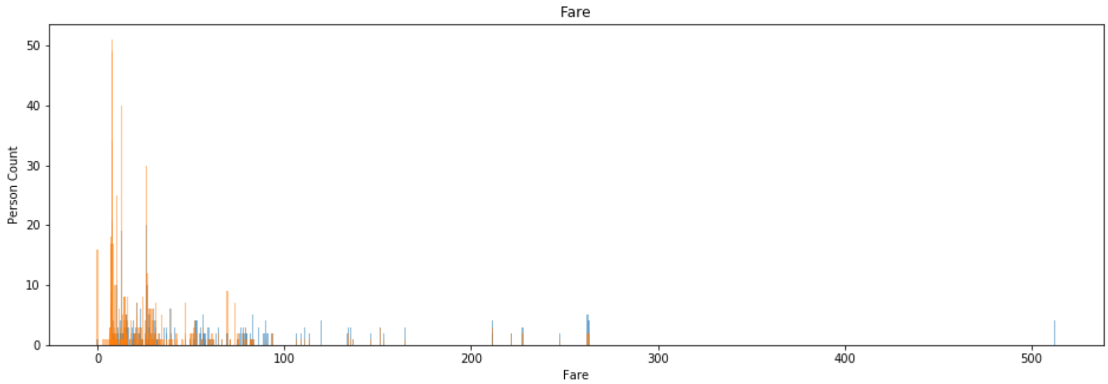
Parch Vs Person Count Vs Survival

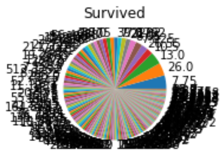


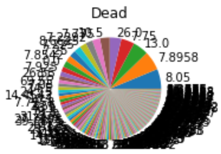




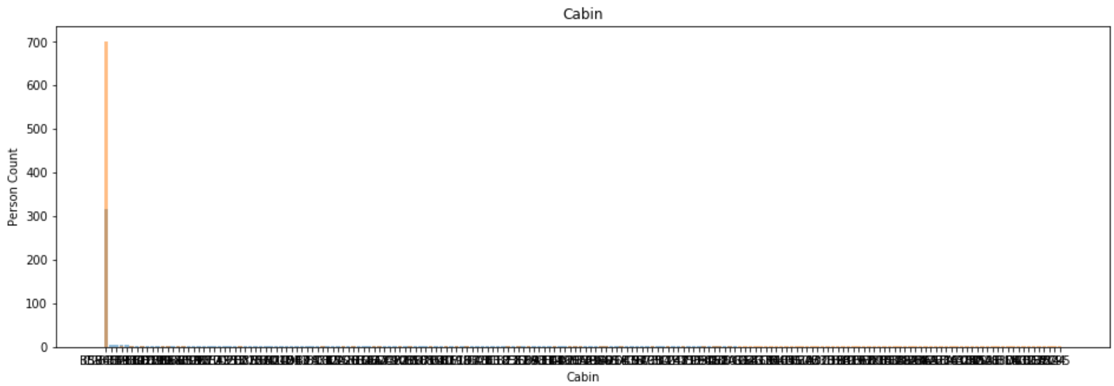
Fare Vs Person Count Vs Survival

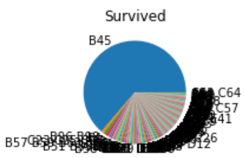


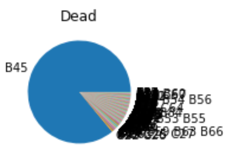




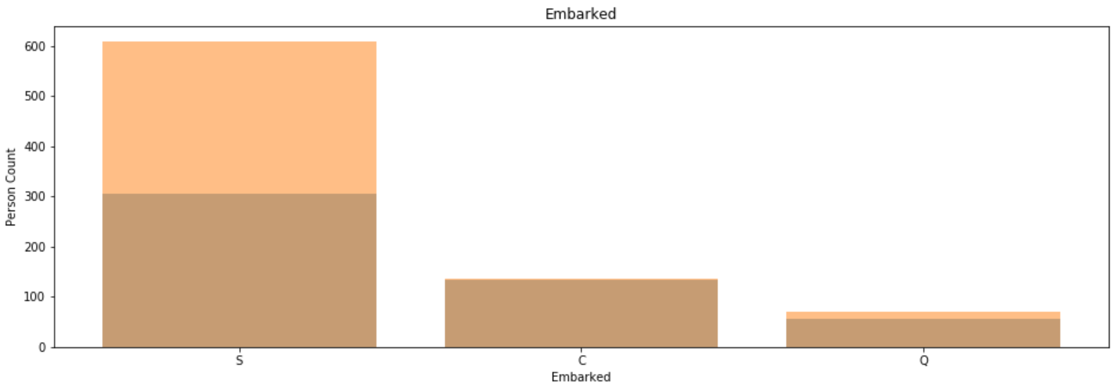
Cabin Vs Person Count Vs Survival







Embarked Vs Person Count Vs Survival



In [35]:
```
data_encoded=pd.get_dummies(data)
```

In [36]:
```
data_encoded.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1309 entries, 0 to 417
Columns: 211 entries, Age to Embarked_S
dtypes: float64(2), uint8(209)
memory usage: 297.8 KB
```

In [37]:
```
data_encoded.values.shape
```

Out[37]:
```
(1309, 211)
```

In [38]:
```
X=data_encoded.values
y=target.values
```

In [39]:
```
X_train,X_test, y_train, y_test = train_test_split(X,y, random_state=42, test_size=0.25)
```

In [40]:
```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[40]:
```
((981, 211), (328, 211), (981,), (328,))
```

In [41]:
```
X_train
```

Out[41]:
```
array([[29.    , 26.    , 0.    , ..., 0.    , 0.    , 1.    ],
       [43.    , 46.9   , 0.    , ..., 0.    , 0.    , 1.    ],
       [26.    , 78.85  , 1.    , ..., 0.    , 0.    , 1.    ],
       ...,
       [17.    , 47.1   , 1.    , ..., 0.    , 0.    , 1.    ],
       [41.    , 14.1083, 0.    , ..., 0.    , 0.    , 1.    ],
       [20.    ,  7.8542, 0.    , ..., 0.    , 0.    , 1.    ]])
```

EVALUATION FUNCTION

In [42]:
```
def classreport(model,xtest):
    print("MODEL RESULT\n============")
    predictions=model.predict(xtest)
    confusion = confusion_matrix(y_test, predictions)
    print('Confusion Matrix:\n',confusion)
    print('\nClassification Report:\n',classification_report(y_test, predictions,
                            target_names=["1", "0"]))
    plt.imshow(confusion)
    plt.xticks([0,1])
    plt.xlabel("Predicted Class")
    plt.yticks([0,1])
    plt.ylabel("True Class")
    plt.colorbar()
    plt.show()
```

MODEL SELECTION

1. SVC

In [43]:
```
param_grid={
    'C':[0.001,0.01,0.1,1,10,100,1000],
    'gamma': [0.001,0.01,0.1,1,10,100,1000],
}
```

In [44]:
```
grid= GridSearchCV(SVC(), param_grid, n_jobs=-1,cv=5)
```

In [45]:
```
grid.fit(X_train,y_train)
```

Out[45]:
```
GridSearchCV(cv=5, error_score='raise-deprecating',
       estimator=SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
  kernel='rbf', max_iter=-1, probability=False, random_state=None,
  shrinking=True, tol=0.001, verbose=False),
       fit_params=None, iid='warn', n_jobs=-1,
       param_grid={'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000], 'gamma': [0.001, 0.01, 0.1, 1, 10, 100, 1000]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring=None, verbose=0)
```

In [46]:
```
grid.best_estimator_
```

Out[46]:
```
SVC(C=100, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

In [47]:
```
grid.best_score_
```

Out[47]:
```
0.8430173292558614
```

In [48]:
```
grid.best_estimator_.score(X_test,y_test)
```

Out[48]:
```
0.8475609756097561
```

In [49]:
```
grid.best_estimator_.score(X_train,y_train)
```

Out[49]:
```
0.8827726809378186
```

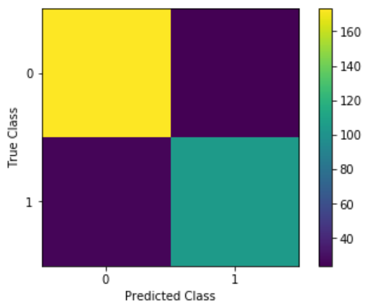
In [50]:
```
classreport(grid.best_estimator_,X_test)
```

```
MODEL RESULT
============
Confusion Matrix:
 [[173  24]
 [ 26 105]]

Classification Report:
               precision    recall  f1-score   support

           1       0.87      0.88      0.87       197
           0       0.81      0.80      0.81       131

   micro avg       0.85      0.85      0.85       328
   macro avg       0.84      0.84      0.84       328
weighted avg       0.85      0.85      0.85       328
```



SCALING

Standard Scaler

In [51]:
```
scaler= StandardScaler()
```

In [52]:
```
scaler.fit(X_train)
```

Out[52]:
```
StandardScaler(copy=True, with_mean=True, with_std=True)
```

In [53]:
```
X_train_standard=scaler.transform(X_train)
```

In [54]:
```
X_test_standard=scaler.transform(X_test)
```

In [55]:
```
grid= GridSearchCV(SVC(), param_grid, n_jobs=-1,cv=5)
grid.fit(X_train_standard,y_train)
```

Out[55]:
```
GridSearchCV(cv=5, error_score='raise-deprecating',
       estimator=SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
  kernel='rbf', max_iter=-1, probability=False, random_state=None,
  shrinking=True, tol=0.001, verbose=False),
       fit_params=None, iid='warn', n_jobs=-1,
       param_grid={'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000], 'gamma': [0.001, 0.01, 0.1, 1, 10, 100, 1000]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring=None, verbose=0)
```

In [56]:
```
grid.best_estimator_
```

Out[56]:
```
SVC(C=10, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

In [57]:
```
grid.best_score_
```

Out[57]:
```
0.8583078491335372
```

In [58]:
```
grid.best_estimator_.score(X_test_standard,y_test)
```

Out[58]:
```
0.8353658536585366
```

In [59]:
```
grid.best_estimator_.score(X_train_standard,y_train)
```

Out[59]:
```
0.8990825688073395
```

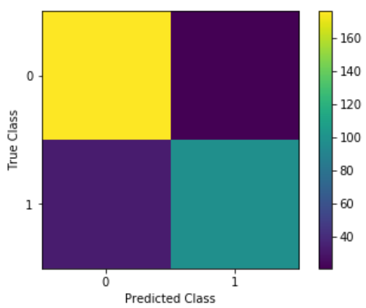
In [60]:
```
classreport(grid.best_estimator_,X_test_standard)
```

```
MODEL RESULT
============
Confusion Matrix:
 [[176  21]
 [ 33  98]]

Classification Report:
               precision    recall  f1-score   support

           1       0.84      0.89      0.87       197
           0       0.82      0.75      0.78       131

   micro avg       0.84      0.84      0.84       328
   macro avg       0.83      0.82      0.83       328
weighted avg       0.83      0.84      0.83       328
```



Min Max Scaler

In [61]:
```
scaler=MinMaxScaler()
```

In [62]:
```
scaler.fit(X_train)
X_train_minmax=scaler.transform(X_train)
X_test_minmax=scaler.transform(X_test)
```

In [63]:
```
grid= GridSearchCV(SVC(), param_grid, n_jobs=-1,cv=5)
grid.fit(X_train_minmax,y_train)
```

Out[63]:
```
GridSearchCV(cv=5, error_score='raise-deprecating',
       estimator=SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
  kernel='rbf', max_iter=-1, probability=False, random_state=None,
  shrinking=True, tol=0.001, verbose=False),
       fit_params=None, iid='warn', n_jobs=-1,
       param_grid={'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000], 'gamma': [0.001, 0.01, 0.1, 1, 10, 100, 1000]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring=None, verbose=0)
```

In [64]:
```python
print(
    'Best Estimator:\n',
    grid.best_estimator_,
    '\nBest CV Score:\n',
    grid.best_score_,
    '\nTest Score:\n',
    grid.best_estimator_.score(X_test_minmax,y_test),
    '\nTrain Score:\n',
    grid.best_estimator_.score(X_train_minmax,y_train)
)
```

```
Best Estimator:
 SVC(C=1000, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma=0.01, kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
Best CV Score:
 0.8583078491335372
Test Score:
 0.8475609756097561
Train Score:
 0.9072375127420998
```

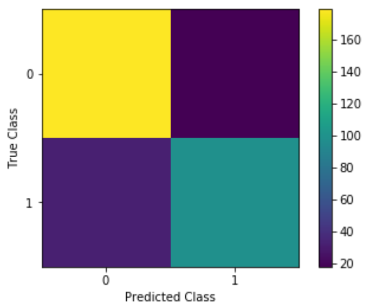
In [65]:
```python
classreport(grid.best_estimator_,X_test_minmax)
```

```
MODEL RESULT
============
Confusion Matrix:
 [[179  18]
 [ 32  99]]

Classification Report:
               precision    recall  f1-score   support

           1       0.85      0.91      0.88       197
           0       0.85      0.76      0.80       131

   micro avg       0.85      0.85      0.85       328
   macro avg       0.85      0.83      0.84       328
weighted avg       0.85      0.85      0.85       328
```

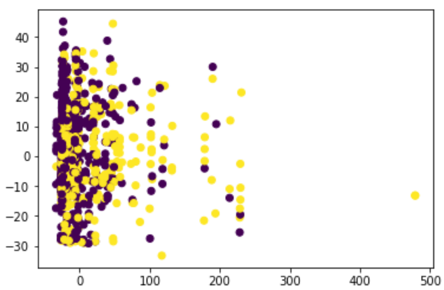


Feature Engineering

PCA

In [66]:
```python
pca= PCA(n_components=2)
X_train_pca=pca.fit_transform(X_train)
X_test_pca=pca.fit_transform(X_test)
plt.scatter(X_train_pca[:,0],X_train_pca[:,1],c=y_train)
```

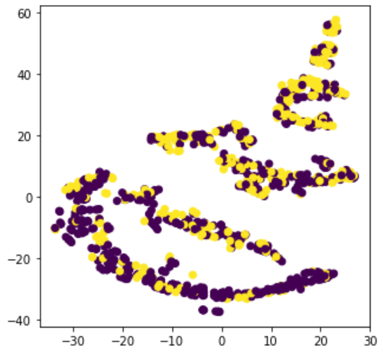
Out[66]: <matplotlib.collections.PathCollection at 0x19cf050bbe0>



TSNE

```
In [67]: tsne=TSNE(n_components=2,random_state=34)
         X_train_tsne=tsne.fit_transform(X_train)
         plt.figure(figsize=(5,5))
         plt.scatter(X_train_tsne[:,0],X_train_tsne[:,1],c=y_train)
```

```
Out[67]: <matplotlib.collections.PathCollection at 0x19cf03ef5c0>
```



Scaled PCA

```
In [68]: n=150
         X_train_minmax_pca=PCA(n_components=n).fit_transform(X_train_minmax)
         X_test_minmax_pca=PCA(n_components=n).fit_transform(X_test_minmax)
         X_train_standard_pca=PCA(n_components=n).fit_transform(X_train_standard)
         X_test_standard_pca=PCA(n_components=n).fit_transform(X_test_standard)
```

```
In [69]: grid= GridSearchCV(SVC(), param_grid, n_jobs=-1,cv=5)
         grid.fit(X_train_minmax_pca,y_train)
         print(
             'Best Estimator:\n',
             grid.best_estimator_,
             '\nBest CV Score:\n',
             grid.best_score_,
             '\nTest Score:\n',
             grid.best_estimator_.score(X_test_minmax_pca,y_test),
             '\nTrain Score:\n',
             grid.best_estimator_.score(X_train_minmax_pca,y_train)
         )
```

```
Best Estimator:
 SVC(C=1000, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma=0.01, kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
Best CV Score:
 0.8583078491335372
Test Score:
 0.8201219512195121
Train Score:
 0.9072375127420998
```

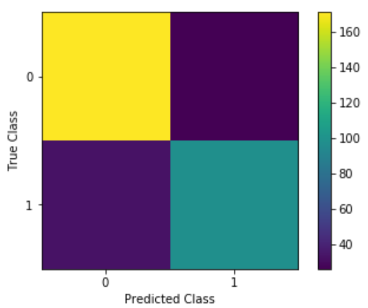
```
In [70]: classreport(grid.best_estimator_,X_test_minmax_pca)
```

```
MODEL RESULT
============
Confusion Matrix:
 [[171  26]
 [ 33  98]]

Classification Report:
               precision    recall  f1-score   support

           1       0.84      0.87      0.85       197
           0       0.79      0.75      0.77       131

   micro avg       0.82      0.82      0.82       328
   macro avg       0.81      0.81      0.81       328
weighted avg       0.82      0.82      0.82       328
```



In [71]:
```
grid= GridSearchCV(SVC(), param_grid, n_jobs=-1,cv=5)
grid.fit(X_train_standard_pca,y_train)
print(
    'Best Estimator:\n',
    grid.best_estimator_,
    '\nBest CV Score:\n',
    grid.best_score_,
    '\nTest Score:\n',
    grid.best_estimator_.score(X_test_standard_pca,y_test),
    '\nTrain Score:\n',
    grid.best_estimator_.score(X_train_standard_pca,y_train)
)
```

```
Best Estimator:
 SVC(C=1, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
Best CV Score:
 0.8511722731906218
Test Score:
 0.32621951219512196
Train Score:
 0.8827726809378186
```

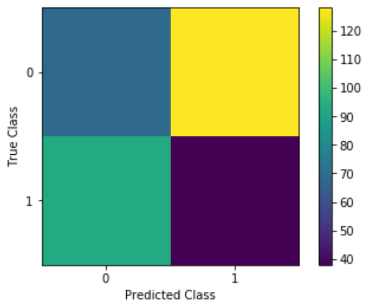
In [72]:
```
classreport(grid.best_estimator_,X_test_standard_pca)
```

```
MODEL RESULT
============
Confusion Matrix:
 [[ 69 128]
 [ 93  38]]

Classification Report:
               precision    recall  f1-score   support

           1       0.43      0.35      0.38       197
           0       0.23      0.29      0.26       131

   micro avg       0.33      0.33      0.33       328
   macro avg       0.33      0.32      0.32       328
weighted avg       0.35      0.33      0.33       328
```



Bad Idea!

2. Random Forests Classifier

In [73]:
```
param_grid={
    'max_depth':[2,3,4,5,10,15],
    'n_estimators':[50,100,150,200,500,1000],
}
```

In [74]:
```
grid_rfc=GridSearchCV(RandomForestClassifier(), param_grid, n_jobs=-1,cv=5)
grid_rfc.fit(X_train,y_train)
print(
    'Best Estimator:\n',
    grid_rfc.best_estimator_,
    '\nBest CV Score:\n',
    grid_rfc.best_score_,
    '\nTest Score:\n',
    grid_rfc.best_estimator_.score(X_test,y_test),
    '\nTrain Score:\n',
    grid_rfc.best_estimator_.score(X_train,y_train)
)
```

```
Best Estimator:
 RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=10, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=500, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
Best CV Score:
 0.8623853211009175
Test Score:
 0.8567073170731707
Train Score:
 0.8939857288481141
```

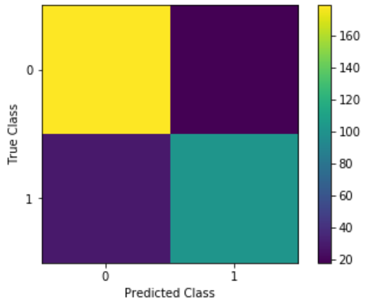
In [75]:
```
classreport(grid_rfc.best_estimator_,X_test)
```

```
MODEL RESULT
============
Confusion Matrix:
 [[179  18]
 [ 29 102]]

Classification Report:
               precision    recall  f1-score   support

           1       0.86      0.91      0.88       197
           0       0.85      0.78      0.81       131

   micro avg       0.86      0.86      0.86       328
   macro avg       0.86      0.84      0.85       328
weighted avg       0.86      0.86      0.86       328
```

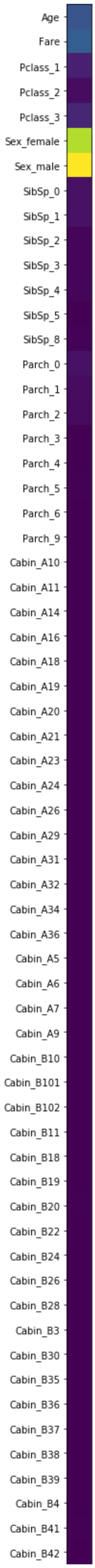


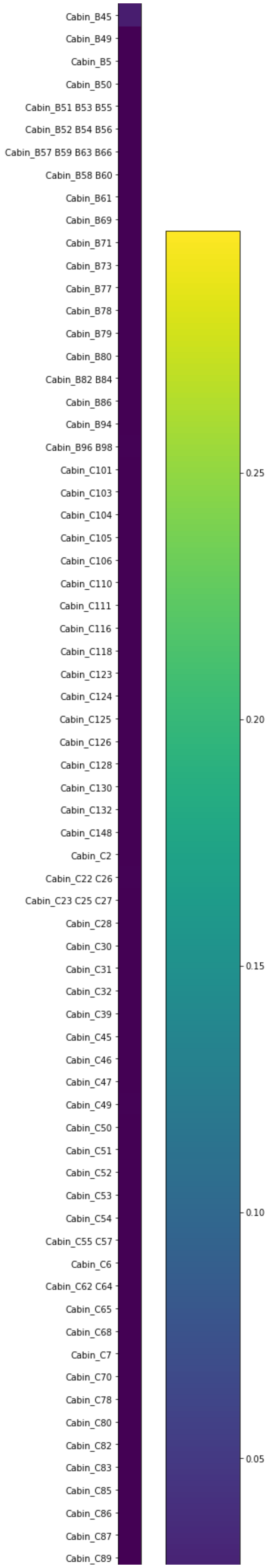
In [76]:
```
grid_rfc.best_estimator_.feature_importances_.reshape(1,-1)
```

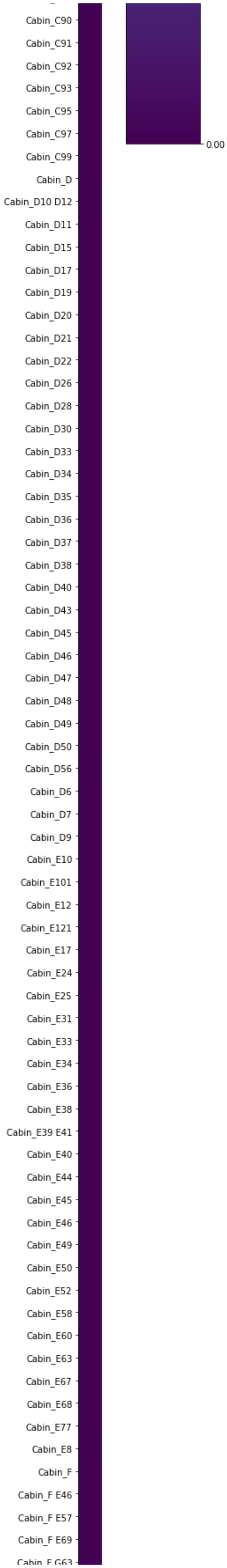
Out[76]:
```
array([[7.86678957e-02, 8.96774289e-02, 2.57960246e-02, 1.00010157e-02,
        3.23450054e-02, 2.64540974e-01, 2.99078053e-01, 1.30813224e-02,
        1.48765076e-02, 3.90073329e-03, 4.09276702e-03, 4.83208627e-03,
        1.14893199e-03, 3.01558769e-03, 1.43473851e-02, 9.64087918e-03,
        8.69727306e-03, 9.52629865e-04, 7.56560534e-04, 1.66165596e-03,
        9.32208696e-04, 5.63481388e-04, 2.03686417e-04, 3.14776712e-04,
        9.73860246e-05, 1.16111855e-04, 1.22576253e-04, 8.10284602e-05,
        8.32641591e-04, 1.31890061e-04, 0.00000000e+00, 0.00000000e+00,
        1.09406521e-03, 1.07882129e-04, 0.00000000e+00, 1.75060857e-04,
        7.30314884e-04, 1.38684419e-04, 0.00000000e+00, 0.00000000e+00,
        0.00000000e+00, 1.30241218e-04, 0.00000000e+00, 6.70883046e-04,
        9.40331466e-05, 9.61032695e-05, 5.97245568e-04, 0.00000000e+00,
        1.35355589e-03, 3.05790850e-04, 3.12976849e-04, 1.69544051e-04,
        4.72028469e-04, 5.51019594e-05, 0.00000000e+00, 5.93451757e-04,
        1.44472011e-04, 8.71452846e-05, 0.00000000e+00, 2.67305025e-04,
        2.48147129e-04, 1.46468022e-03, 2.34389432e-04, 2.41024633e-02,
        5.71988679e-04, 1.91721559e-04, 8.92105241e-04, 0.00000000e+00,
        1.01292466e-04, 3.20258115e-04, 3.88231633e-04, 1.51331144e-04,
        2.39448702e-04, 2.13976608e-04, 1.38260155e-04, 3.99660959e-04,
        3.32342472e-04, 1.38528528e-04, 3.36832545e-04, 2.35172067e-04,
        3.06034670e-04, 8.42946093e-05, 1.26502734e-03, 6.00724342e-04,
        3.03260155e-04, 1.03677576e-03, 8.14897923e-05, 3.54897497e-05,
        1.09434836e-04, 1.87387768e-04, 2.64297290e-04, 3.23176028e-04,
        1.33453994e-04, 0.00000000e+00, 4.20853000e-04, 1.21669986e-04,
        0.00000000e+00, 1.62966942e-04, 2.94440291e-04, 0.00000000e+00,
        1.76878844e-04, 1.68738047e-03, 6.68031900e-04, 0.00000000e+00,
        1.12849175e-04, 2.12737683e-04, 4.22149117e-04, 9.47335468e-05,
        0.00000000e+00, 1.97191420e-04, 9.63873404e-04, 1.29177494e-03,
        0.00000000e+00, 2.14552587e-04, 1.12194299e-03, 2.02705077e-04,
        3.92133584e-04, 1.95600622e-04, 4.92628018e-04, 1.34050637e-04,
        2.17752294e-04, 2.95966621e-04, 7.48223056e-04, 7.61260036e-04,
        4.68664382e-04, 1.69907592e-04, 0.00000000e+00, 2.05803832e-04,
        1.59314903e-04, 2.77742781e-04, 8.79649281e-05, 1.58174981e-04,
        0.00000000e+00, 3.35571176e-04, 1.05848841e-03, 8.76214541e-04,
        0.00000000e+00, 6.91763154e-05, 9.14275946e-05, 4.35550067e-04,
        1.22806392e-03, 0.00000000e+00, 0.00000000e+00, 5.09875918e-04,
        1.11544778e-03, 1.12337002e-04, 1.56771979e-04, 0.00000000e+00,
        7.51384187e-04, 7.04244513e-04, 4.19612829e-04, 8.75154090e-04,
        9.71118666e-05, 8.79832853e-04, 3.58940357e-04, 1.72657304e-04,
        1.96526714e-04, 0.00000000e+00, 0.00000000e+00, 6.87136477e-04,
        1.60057577e-04, 3.54101623e-04, 4.55823570e-04, 8.19585452e-04,
        1.11436227e-04, 1.34758987e-03, 2.23139774e-04, 0.00000000e+00,
        1.29955066e-04, 0.00000000e+00, 3.00851708e-04, 1.08098428e-03,
        1.14769369e-03, 0.00000000e+00, 1.89757269e-03, 1.05030072e-03,
        1.63218488e-04, 4.87996306e-04, 7.07146326e-04, 0.00000000e+00,
        1.14331992e-04, 2.25937309e-04, 9.59638742e-05, 1.54272337e-04,
        0.00000000e+00, 4.19732540e-04, 2.63265513e-04, 1.13800343e-03,
        3.19551206e-04, 7.45302956e-05, 5.80040440e-05, 1.22215375e-04,
        2.23621981e-04, 0.00000000e+00, 0.00000000e+00, 1.30166564e-03,
        2.87986985e-04, 6.84376129e-05, 1.02808786e-04, 0.00000000e+00,
        5.26528414e-05, 1.89812115e-04, 1.85511454e-03, 2.32781066e-03,
        9.81046132e-05, 1.10927566e-03, 1.41216539e-03, 1.60147434e-04,
        9.14623264e-03, 5.82873184e-03, 1.06146563e-02]])
```

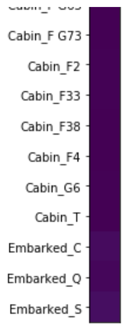
In [77]:
```
plt.figure(figsize=(10,100))
plt.imshow((grid_rfc.best_estimator_.feature_importances_.reshape(-1,1)))
plt.xticks(())
plt.yticks((np.arange(211)),data_encoded.columns)
plt.colorbar()
```

Out[77]: <matplotlib.colorbar.Colorbar at 0x19cf03a0f60>




Cabin_B45
Cabin_B49
Cabin_B5
Cabin_B50
Cabin_B51 B53 B55
Cabin_B52 B54 B56
Cabin_B57 B59 B63 B66
Cabin_B58 B60
Cabin_B61
Cabin_B69
Cabin_B71
Cabin_B73
Cabin_B77
Cabin_B78
Cabin_B79
Cabin_B80
Cabin_B82 B84
Cabin_B86
Cabin_B94
Cabin_B96 B98
Cabin_C101
Cabin_C103
Cabin_C104
Cabin_C105
Cabin_C106
Cabin_C110
Cabin_C111
Cabin_C116
Cabin_C118
Cabin_C123
Cabin_C124
Cabin_C125
Cabin_C126
Cabin_C128
Cabin_C130
Cabin_C132
Cabin_C148
Cabin_C2
Cabin_C22 C26
Cabin_C23 C25 C27
Cabin_C28
Cabin_C30
Cabin_C31
Cabin_C32
Cabin_C39
Cabin_C45
Cabin_C46
Cabin_C47
Cabin_C49
Cabin_C50
Cabin_C51
Cabin_C52
Cabin_C53
Cabin_C54
Cabin_C55 C57
Cabin_C6
Cabin_C62 C64
Cabin_C65
Cabin_C68
Cabin_C7
Cabin_C70
Cabin_C78
Cabin_C80
Cabin_C82
Cabin_C83
Cabin_C85
Cabin_C86
Cabin_C87
Cabin_C89
0.25
0.20
0.15
0.10
0.05


Cabin_C90
Cabin_C91
Cabin_C92
Cabin_C93
Cabin_C95
Cabin_C97
Cabin_C99
Cabin_D
Cabin_D10 D12
Cabin_D11
Cabin_D15
Cabin_D17
Cabin_D19
Cabin_D20
Cabin_D21
Cabin_D22
Cabin_D26
Cabin_D28
Cabin_D30
Cabin_D33
Cabin_D34
Cabin_D35
Cabin_D36
Cabin_D37
Cabin_D38
Cabin_D40
Cabin_D43
Cabin_D45
Cabin_D46
Cabin_D47
Cabin_D48
Cabin_D49
Cabin_D50
Cabin_D56
Cabin_D6
Cabin_D7
Cabin_D9
Cabin_E10
Cabin_E101
Cabin_E12
Cabin_E121
Cabin_E17
Cabin_E24
Cabin_E25
Cabin_E31
Cabin_E33
Cabin_E34
Cabin_E36
Cabin_E38
Cabin_E39 E41
Cabin_E40
Cabin_E44
Cabin_E45
Cabin_E46
Cabin_E49
Cabin_E50
Cabin_E52
Cabin_E58
Cabin_E60
Cabin_E63
Cabin_E67
Cabin_E68
Cabin_E77
Cabin_E8
Cabin_F
Cabin_F E46
Cabin_F E57
Cabin_F E69
0.00



Feature Engineering

In [78]:
```python
select=SelectFromModel(grid_rfc.best_estimator_)
select.fit(X_train,y_train)
```

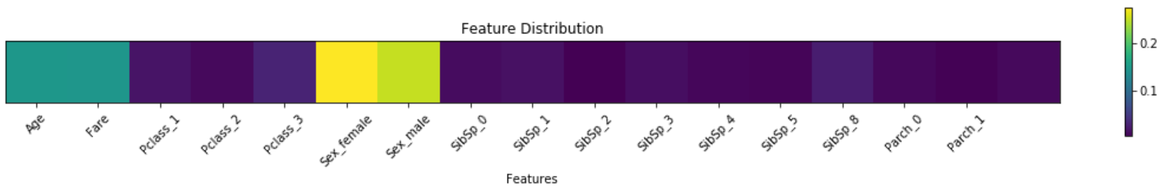
Out[78]:
```
SelectFromModel(estimator=RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=10, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=500, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False),
        max_features=None, norm_order=1, prefit=False, threshold=None)
```

In [79]:
```python
X_train_rfc_selected= select.transform(X_train)
X_test_rfc_selected= select.transform(X_test)
```

In [80]:
```python
grid_rfc.best_estimator_.fit(X_train_rfc_selected,y_train)
print(
    'Test Score:\n',
    grid_rfc.best_estimator_.score(X_test_rfc_selected,y_test),
    '\nTrain Score:\n',
    grid_rfc.best_estimator_.score(X_train_rfc_selected,y_train)
)
```

```
Test Score:
 0.8567073170731707
Train Score:
 0.9439347604485219
```

In [81]:
```python
plt.figure(figsize=(20,2))
plt.imshow(grid_rfc.best_estimator_.feature_importances_.reshape(1,-1))
plt.yticks(())
plt.xticks((np.arange(16)),data_encoded.columns[:16],rotation=45)
plt.colorbar()
plt.xlabel("Features")
plt.title("Feature Distribution")
plt.show()
```



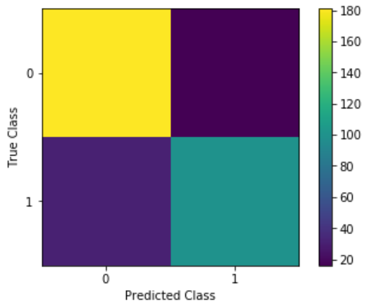
In [82]:
```
classreport(grid_rfc.best_estimator_,X_test_rfc_selected)
```

```
MODEL RESULT
============
Confusion Matrix:
 [[181  16]
 [ 31 100]]

Classification Report:
               precision    recall  f1-score   support

           1       0.85      0.92      0.89       197
           0       0.86      0.76      0.81       131

   micro avg       0.86      0.86      0.86       328
   macro avg       0.86      0.84      0.85       328
weighted avg       0.86      0.86      0.85       328
```



Clearly shows if you are male you are most likely to survive

3. Logistic Regression

In [83]:
```
param_grid={
    'C':[0.001,0.01,0.1,1,10,100,1000]
}
```

In [84]:
```
grid_lr = GridSearchCV(LogisticRegression(), param_grid, cv=5, n_jobs=-1)
grid_lr.fit(X_train,y_train)
print(
    'Best Estimator:\n',
    grid_lr.best_estimator_,
    '\nBest CV Score:\n',
    grid_lr.best_score_,
    '\nTest Score:\n',
    grid_lr.best_estimator_.score(X_test,y_test),
    '\nTrain Score:\n',
    grid_lr.best_estimator_.score(X_train,y_train)
)
```

```
Best Estimator:
 LogisticRegression(C=1, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=None, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
Best CV Score:
 0.8562691131498471
Test Score:
 0.8567073170731707
Train Score:
 0.8776758409785933
```

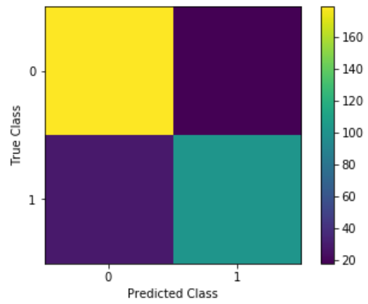
In [85]:
```
classreport(grid_lr.best_estimator_,X_test)
```

```
MODEL RESULT
============
Confusion Matrix:
 [[179  18]
 [ 29 102]]

Classification Report:
               precision    recall  f1-score   support

           1       0.86      0.91      0.88       197
           0       0.85      0.78      0.81       131

   micro avg       0.86      0.86      0.86       328
   macro avg       0.86      0.84      0.85       328
weighted avg       0.86      0.86      0.86       328
```



4. K Nearest Neighbors

In [86]:
```
param_grid={
    'n_neighbors':[1,2,3,4,5,6,7,8,9,10]
}
```

In [87]:
```
grid_knn = GridSearchCV(KNeighborsClassifier(), param_grid, cv=5, n_jobs=-1)
grid_knn.fit(X_train,y_train)
print(
    'Best Estimator:\n',
    grid_knn.best_estimator_,
    '\nBest CV Score:\n',
    grid_knn.best_score_,
    '\nTest Score:\n',
    grid_knn.best_estimator_.score(X_test,y_test),
    '\nTrain Score:\n',
    grid_knn.best_estimator_.score(X_train,y_train)
)
```

```
Best Estimator:
 KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=None, n_neighbors=5, p=2,
           weights='uniform')
Best CV Score:
 0.6962283384301733
Test Score:
 0.7073170731707317
Train Score:
 0.8032619775739042
```

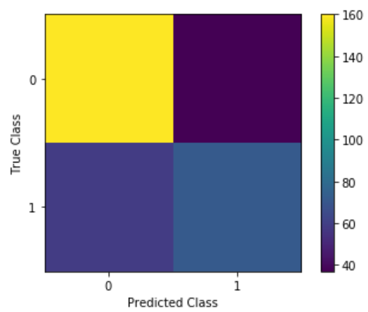
In [88]:
```
classreport(grid_knn.best_estimator_,X_test)
```

```
MODEL RESULT
============
Confusion Matrix:
 [[160  37]
 [ 59  72]]

Classification Report:
               precision    recall  f1-score   support

           1       0.73      0.81      0.77       197
           0       0.66      0.55      0.60       131

   micro avg       0.71      0.71      0.71       328
   macro avg       0.70      0.68      0.68       328
weighted avg       0.70      0.71      0.70       328
```



With Age Binning

In [89]:
```
data.head()
```

Out[89]:

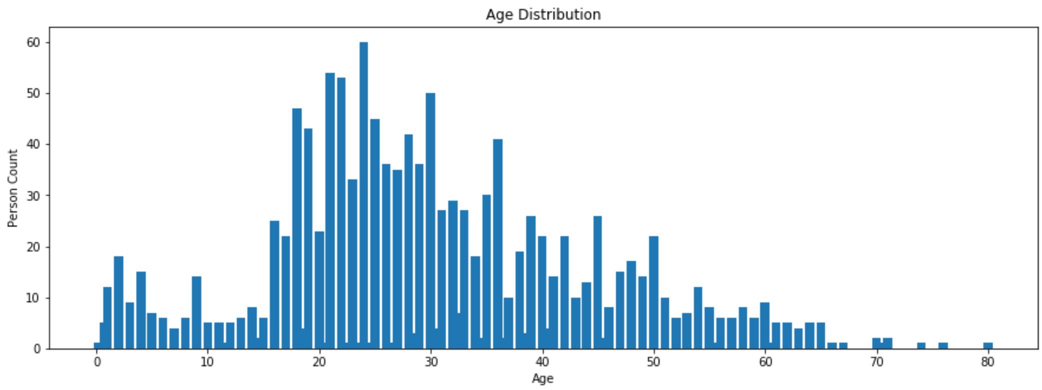
	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
0	3	male	22.0	1	0	7.2500	B45	S
1	1	female	38.0	1	0	71.2833	C85	C
2	3	female	26.0	0	0	7.9250	B45	S
3	1	female	35.0	1	0	53.1000	C123	S
4	3	male	35.0	0	0	8.0500	B45	S

In [90]:
```
ages=[]
for age in data.Age.values:
    if age<10:
        ages.append('<10')
    elif age>10 and age<=20:
        ages.append('10-20')
    elif age>20 and age<=30:
        ages.append('20-30')
    elif age>30 and age<=40:
        ages.append('30-40')
    elif age>40 and age<=50:
        ages.append('40-50')
    elif age>50 and age<=60:
        ages.append('50-60')
    elif age>60 and age<=70:
        ages.append('60-70')
    elif age>70 and age<=80:
        ages.append('70-80')
    else:
        ages.append('>80')
```

In [91]:
```
len(ages)
```

Out[91]: 1309

In [92]:
```
plt.figure(figsize=(15,5))
plt.title("Age Distribution")
plt.bar(data.Age.value_counts().keys(),data.Age.value_counts().values)
plt.xlabel("Age")
plt.ylabel("Person Count")
plt.show()
```



In [93]:
```
data.index=np.arange(1309)
target.index=np.arange(1309)
```

In [94]:
```
data_age_bin=data
```

In [95]:
```
data_age_bin.Age=(ages)
```

In [96]:
```
data_age_bin.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1309 entries, 0 to 1308
Data columns (total 8 columns):
Pclass      1309 non-null object
Sex         1309 non-null object
Age         1309 non-null object
SibSp       1309 non-null object
Parch       1309 non-null object
Fare        1309 non-null float64
Cabin       1309 non-null object
Embarked    1309 non-null object
dtypes: float64(1), object(7)
memory usage: 92.0+ KB
```

In [97]:
```
data_age_bin.Age.value_counts()
```
Out[97]:
```
20-30    452
30-40    267
10-20    197
40-50    167
<10      107
50-60     79
60-70     29
70-80      6
>80        5
Name: Age, dtype: int64
```
In [98]:
```
data_age_bin= pd.get_dummies(data_age_bin)
```
In [99]:
```
data_age_bin.head()
```
Out[99]:

	Fare	Pclass_1	Pclass_2	Pclass_3	Sex_female	Sex_male	Age_10-20	Age_20-30	Age_30-40	Age_40-50	...	Cabin_F G73	Cabin_F2	Cabin_F33	Cabin_F38	Cabin_F4	Cabin_G6	C
0	7.2500	0	0	1	0	1	0	1	0	0	...	0	0	0	0	0	0	
1	71.2833	1	0	0	1	0	0	0	1	0	...	0	0	0	0	0	0	
2	7.9250	0	0	1	1	0	0	1	0	0	...	0	0	0	0	0	0	
3	53.1000	1	0	0	1	0	0	0	1	0	...	0	0	0	0	0	0	
4	8.0500	0	0	1	0	1	0	0	1	0	...	0	0	0	0	0	0	

5 rows × 219 columns

In [100]:
```
data_age_bin.columns
```
Out[100]:
```
Index(['Fare', 'Pclass_1', 'Pclass_2', 'Pclass_3', 'Sex_female', 'Sex_male',
       'Age_10-20', 'Age_20-30', 'Age_30-40', 'Age_40-50',
       ...
       'Cabin_F G73', 'Cabin_F2', 'Cabin_F33', 'Cabin_F38', 'Cabin_F4',
       'Cabin_G6', 'Cabin_T', 'Embarked_C', 'Embarked_Q', 'Embarked_S'],
      dtype='object', length=219)
```
In [101]:
```
X=data_age_bin
```
In [102]:
```
X_train,X_test, y_train,y_test= train_test_split(X,y, test_size=0.25, random_state=45)
```
In [103]:
```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```
Out[103]:
```
((981, 219), (328, 219), (981,), (328,))
```

Using Random Forests

In [104]:
```
param_grid={
    'max_depth':[2,3,4,5,10,15],
    'n_estimators':[50,100,150,200,500,1000],
}
grid_rfc_age_bin=GridSearchCV(RandomForestClassifier(), param_grid, n_jobs=-1,cv=5)
grid_rfc_age_bin.fit(X_train,y_train)
print(
    'Best Estimator:\n',
    grid_rfc_age_bin.best_estimator_,
    '\nBest CV Score:\n',
    grid_rfc_age_bin.best_score_,
    '\nTest Score:\n',
    grid_rfc_age_bin.best_estimator_.score(X_test,y_test),
    '\nTrain Score:\n',
    grid_rfc_age_bin.best_estimator_.score(X_train,y_train)
)
```
```
Best Estimator:
 RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=10, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=50, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
Best CV Score:
 0.8511722731906218
Test Score:
 0.9085365853658537
Train Score:
 0.890927624872579
```

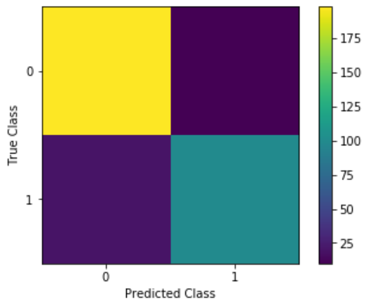
In [105]:
```
classreport(grid_rfc_age_bin.best_estimator_,X_test)
```

```
MODEL RESULT
============
Confusion Matrix:
 [[198  10]
 [ 20 100]]

Classification Report:
               precision    recall  f1-score   support

           1       0.91      0.95      0.93       208
           0       0.91      0.83      0.87       120

   micro avg       0.91      0.91      0.91       328
   macro avg       0.91      0.89      0.90       328
weighted avg       0.91      0.91      0.91       328
```



Feature Selection

In [106]:
```
select= SelectFromModel(grid_rfc_age_bin.best_estimator_)
select.fit(X_train,y_train)
```

Out[106]:
```
SelectFromModel(estimator=RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=10, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=50, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False),
        max_features=None, norm_order=1, prefit=False, threshold=None)
```

In [107]:
```
bool_array=select.get_support()
```

In [108]:
```
selected_features=[]
for i in range(219):
    if bool_array[i]==True:
        selected_features.append((data_age_bin.columns)[i])
```

In [109]:
```
selected_features
```

Out[109]:
```
['Fare',
 'Pclass_1',
 'Pclass_2',
 'Pclass_3',
 'Sex_female',
 'Sex_male',
 'Age_10-20',
 'Age_20-30',
 'Age_30-40',
 'Age_40-50',
 'Age_<10',
 'SibSp_0',
 'SibSp_1',
 'SibSp_3',
 'Parch_0',
 'Parch_1',
 'Parch_2',
 'Cabin_B45',
 'Embarked_C',
 'Embarked_Q',
 'Embarked_S']
```

In [110]:
```
X_train_age_bin_selected= select.transform(X_train)
X_test_age_bin_selected= select.transform(X_test)
```

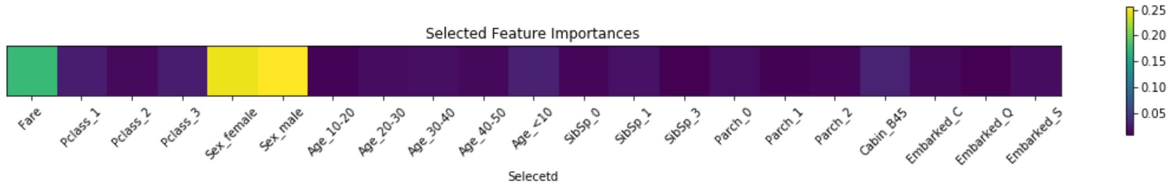
In [111]:
```
grid_rfc_age_bin.best_estimator_.fit(X_train_age_bin_selected,y_train)
print(
    'Test Score:\n',
    grid_rfc_age_bin.best_estimator_.score(X_test_age_bin_selected,y_test),
    '\nTrain Score:\n',
    grid_rfc_age_bin.best_estimator_.score(X_train_age_bin_selected,y_train)
)
```

```
Test Score:
 0.8780487804878049
Train Score:
 0.9327217125382263
```

In [112]:
```
grid_rfc_age_bin.best_estimator_.feature_importances_.reshape(1,-1)
```

Out[112]:
```
array([[0.17532787, 0.02815255, 0.01493118, 0.02835921, 0.24888015,
        0.25653292, 0.01121432, 0.01669611, 0.0189162 , 0.01390441,
        0.03094476, 0.01285323, 0.02083382, 0.00854847, 0.01814951,
        0.01025495, 0.01266504, 0.03290229, 0.01352123, 0.0092341 ,
        0.0171777 ]])
```

In [113]:
```
plt.figure(figsize=(20,2))
plt.title("Selected Feature Importances")
plt.imshow(grid_rfc_age_bin.best_estimator_.feature_importances_.reshape(1,-1))
plt.xticks((range(21)),selected_features,rotation=45)
plt.yticks(())
plt.xlabel("Selecetd")
plt.colorbar()
plt.show()
```



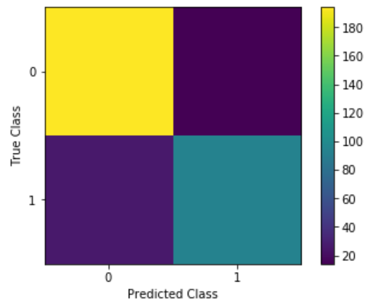
In [114]:
```
classreport(grid_rfc_age_bin.best_estimator_,X_test_age_bin_selected)
```

```
MODEL RESULT
============
Confusion Matrix:
 [[194  14]
 [ 26  94]]

Classification Report:
               precision    recall  f1-score   support

           1       0.88      0.93      0.91       208
           0       0.87      0.78      0.82       120

   micro avg       0.88      0.88      0.88       328
   macro avg       0.88      0.86      0.87       328
weighted avg       0.88      0.88      0.88       328
```



With Age and Fare Binning

In [115]:
```
data.head()
```

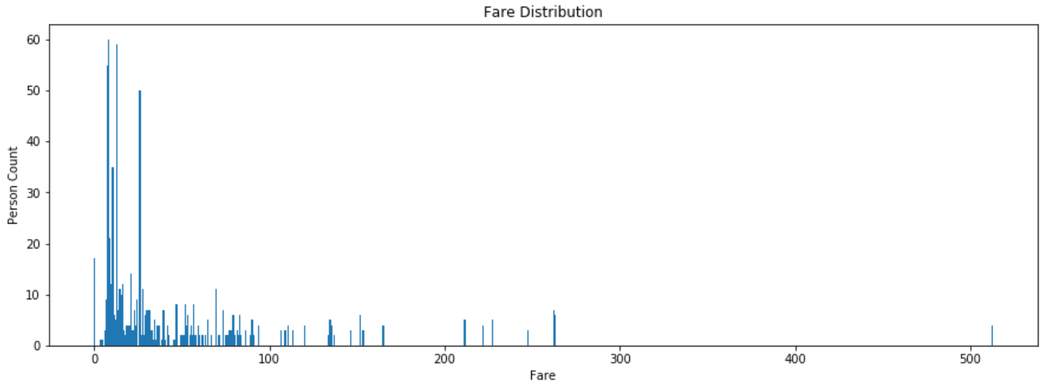
Out[115]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
0	3	male	20-30	1	0	7.2500	B45	S
1	1	female	30-40	1	0	71.2833	C85	C
2	3	female	20-30	0	0	7.9250	B45	S
3	1	female	30-40	1	0	53.1000	C123	S
4	3	male	30-40	0	0	8.0500	B45	S

In [116]:
```
ages[:6]
```

Out[116]:
```
['20-30', '30-40', '20-30', '30-40', '30-40', '30-40']
```

In [117]:
```
plt.figure(figsize=(15,5))
plt.title("Fare Distribution")
plt.bar(data.Fare.value_counts().keys(),data.Fare.value_counts().values)
plt.xlabel("Fare")
plt.ylabel("Person Count")
plt.show()
```



In [118]:
```
data.Fare.value_counts()
```

Out[118]:
```
8.0500      60
13.0000     59
7.7500      55
26.0000     50
7.8958      50
10.5000     35
7.7750      26
7.2292      24
7.9250      23
26.5500     22
8.6625      21
7.8542      21
7.2250      21
7.2500      18
0.0000      17
21.0000     14
9.5000      12
16.1000     12
14.5000     11
69.5500     11
27.7208     11
7.7958      10
14.4542     10
7.8792      10
15.5000     10
24.1500      9
15.2458      9
7.0500       9
56.4958      8
46.9000      8
            ..
45.5000      1
8.4333       1
8.1583       1
8.0292       1
12.7375      1
31.6833      1
42.5000      1
34.0208      1
25.9250      1
12.6500      1
7.0458       1
49.5000      1
7.7208       1
32.3208      1
6.4500       1
8.3000       1
7.5208       1
28.7125      1
25.5875      1
7.7292       1
9.8458       1
8.6542       1
25.7000      1
10.1708      1
7.3125       1
33.5000      1
7.8000       1
26.3875      1
15.5792      1
7.1417       1
Name: Fare, Length: 281, dtype: int64
```

In [119]:
```python
fares=[]
for fare in data.Fare.values:
    if fare<33.3:
        fares.append('CHEAP')
    elif fare>33.3 and fare<66.6:
        fares.append('AVERAGE')
    elif fare>66.6 and fare<100:
        fares.append('EXPENSIVE')
    else:
        fares.append('VERY EXPENSIVE')
```

In [120]:
```python
len(fares)
```

Out[120]:
```
1309
```

In [121]:
```python
data.Fare.values
```

Out[121]:
```
array([ 7.25  , 71.2833,  7.925 , ...,  7.25  ,  8.05  , 22.3583])
```

In [122]:
```python
data_age_bin.head()
```

Out[122]:

	Fare	Pclass_1	Pclass_2	Pclass_3	Sex_female	Sex_male	Age_10-20	Age_20-30	Age_30-40	Age_40-50	...	Cabin_FG73	Cabin_F2	Cabin_F33	Cabin_F38	Cabin_F4	Cabin_G6	C
0	7.2500	0	0	1	0	1	0	1	0	0	...	0	0	0	0	0	0	
1	71.2833	1	0	0	1	0	0	0	1	0	...	0	0	0	0	0	0	
2	7.9250	0	0	1	1	0	0	1	0	0	...	0	0	0	0	0	0	
3	53.1000	1	0	0	1	0	0	0	1	0	...	0	0	0	0	0	0	
4	8.0500	0	0	1	0	1	0	0	1	0	...	0	0	0	0	0	0	

5 rows × 219 columns

In [123]:
```python
data_age_fare_bin=data_age_bin
```

In [124]:
```python
data_age_fare_bin.Fare=fares
```

In [125]:
```python
data_age_fare_bin.head()
```

Out[125]:

	Fare	Pclass_1	Pclass_2	Pclass_3	Sex_female	Sex_male	Age_10-20	Age_20-30	Age_30-40	Age_40-50	...	Cabin_FG73	Cabin_F2	Cabin_F33	Cabin_F38	Cabin_F4	Cabin_G
0	CHEAP	0	0	1	0	1	0	1	0	0	...	0	0	0	0	0	
1	EXPENSIVE	1	0	0	1	0	0	0	1	0	...	0	0	0	0	0	
2	CHEAP	0	0	1	1	0	0	1	0	0	...	0	0	0	0	0	
3	AVERAGE	1	0	0	1	0	0	0	1	0	...	0	0	0	0	0	
4	CHEAP	0	0	1	0	1	0	0	1	0	...	0	0	0	0	0	

5 rows × 219 columns

In [126]:
```python
fare_value_counts=data_age_fare_bin.Fare.value_counts()
```

In [127]:
```python
fare_value_counts
```

Out[127]:
```
CHEAP             1005
AVERAGE            133
VERY EXPENSIVE      86
EXPENSIVE           85
Name: Fare, dtype: int64
```

In [128]:
```python
data_age_fare_bin.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1309 entries, 0 to 1308
Columns: 219 entries, Fare to Embarked_S
dtypes: object(1), uint8(218)
memory usage: 299.1+ KB
```

In [129]:
```python
data_age_fare_bin=pd.get_dummies(data_age_fare_bin)
```

In [130]:
```python
data_age_fare_bin.head()
```

Out[130]:

	Pclass_1	Pclass_2	Pclass_3	Sex_female	Sex_male	Age_10-20	Age_20-30	Age_30-40	Age_40-50	Age_50-60	...	Cabin_F4	Cabin_G6	Cabin_T	Embarked_C	Embarked_Q	Embark
0	0	0	1	0	1	0	1	0	0	0	...	0	0	0	0	0	
1	1	0	0	1	0	0	0	1	0	0	...	0	0	0	1	0	
2	0	0	1	1	0	0	1	0	0	0	...	0	0	0	0	0	
3	1	0	0	1	0	0	0	1	0	0	...	0	0	0	0	0	
4	0	0	1	0	1	0	0	1	0	0	...	0	0	0	0	0	

5 rows × 222 columns

In [131]:
```python
X=data_age_fare_bin.values
```

In [132]:
```python
X_train,X_test,y_train,y_test=train_test_split(X,y,random_state=32,test_size=0.25)
```

In [133]:
```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```
Out[133]:
```
((981, 222), (328, 222), (981,), (328,))
```

Using Random Forests

In [134]:
```
param_grid={
    'max_depth':[2,3,4,5,10,15],
    'n_estimators':[50,100,150,200,500,1000],
}
grid_rfc_age_fare_bin=GridSearchCV(RandomForestClassifier(), param_grid, n_jobs=-1,cv=5)
grid_rfc_age_fare_bin.fit(X_train,y_train)
print(
    'Best Estimator:\n',
    grid_rfc_age_fare_bin.best_estimator_,
    '\nBest CV Score:\n',
    grid_rfc_age_fare_bin.best_score_,
    '\nTest Score:\n',
    grid_rfc_age_fare_bin.best_estimator_.score(X_test,y_test),
    '\nTrain Score:\n',
    grid_rfc_age_fare_bin.best_estimator_.score(X_train,y_train)
)
```

```
Best Estimator:
 RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=10, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=500, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
Best CV Score:
 0.8593272171253823
Test Score:
 0.8689024390243902
Train Score:
 0.8888888888888888
```

In [135]:
```
grid_rfc_age_fare_bin.best_estimator_.feature_importances_
```
Out[135]:
```
array([1.92320150e-02, 1.27658361e-02, 2.99281381e-02, 2.68690847e-01,
       2.94087416e-01, 7.66639724e-03, 8.64525164e-03, 1.05238154e-02,
       7.47044700e-03, 4.20907339e-03, 2.81498407e-03, 1.05232969e-03,
       1.72485117e-02, 1.61588655e-03, 1.53647607e-02, 2.28233439e-02,
       4.39121019e-03, 7.42522078e-03, 6.89314026e-03, 1.07756386e-03,
       3.81308507e-03, 2.56867438e-02, 1.39341885e-02, 1.12727800e-02,
       1.34437169e-03, 1.88091268e-03, 2.95717951e-03, 9.49611031e-04,
       6.74626679e-04, 2.78543153e-04, 3.16040478e-04, 0.00000000e+00,
       1.34369130e-04, 1.09982833e-04, 6.65574927e-05, 1.27189259e-03,
       5.23708819e-05, 9.88423224e-04, 0.00000000e+00, 1.01355343e-03,
       0.00000000e+00, 0.00000000e+00, 2.62854198e-04, 1.01549645e-03,
       1.07919742e-04, 2.86793867e-04, 1.15227021e-03, 1.69729955e-04,
       0.00000000e+00, 1.72014336e-04, 7.14268985e-04, 1.08799997e-04,
       0.00000000e+00, 1.19068456e-04, 1.48790451e-04, 1.45625816e-03,
       2.76311479e-04, 2.07095097e-04, 2.86340634e-04, 2.31735934e-04,
       0.00000000e+00, 2.74015697e-04, 1.17541942e-04, 1.33502034e-04,
       9.60341009e-05, 6.38668618e-05, 8.73271091e-05, 3.86953314e-04,
       1.18160096e-03, 3.21704204e-04, 2.73359886e-02, 1.31927095e-03,
       9.96980078e-05, 1.04557399e-03, 1.29026419e-03, 1.09008965e-04,
       7.01785247e-04, 3.12489164e-04, 1.03224501e-04, 0.00000000e+00,
       3.19722945e-04, 0.00000000e+00, 7.29326624e-04, 1.41477725e-04,
       2.65038747e-04, 0.00000000e+00, 2.48249969e-04, 0.00000000e+00,
       0.00000000e+00, 1.55378997e-03, 2.01578574e-04, 0.00000000e+00,
       0.00000000e+00, 0.00000000e+00, 7.33771197e-04, 0.00000000e+00,
       1.04666785e-04, 0.00000000e+00, 0.00000000e+00, 4.03303235e-04,
       2.68614128e-04, 3.47334316e-04, 1.18217884e-03, 5.98641158e-05,
       0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 2.29438887e-04,
       2.40664809e-03, 7.23878603e-04, 0.00000000e+00, 0.00000000e+00,
       1.98464034e-04, 1.05063982e-04, 1.56277670e-04, 5.97941705e-05,
       3.02687168e-04, 1.18057170e-03, 1.43435616e-03, 4.98935387e-05,
       0.00000000e+00, 2.84430618e-03, 2.07466683e-04, 1.39395619e-04,
       1.80260791e-04, 8.90448851e-04, 3.49147302e-04, 4.10314589e-05,
       4.59445484e-04, 3.31805450e-04, 9.71647759e-04, 6.37846742e-04,
       0.00000000e+00, 4.12305308e-04, 2.00596202e-04, 0.00000000e+00,
       1.57727260e-04, 7.82778193e-05, 1.76382236e-04, 2.76481772e-04,
       5.61838417e-04, 1.77741045e-04, 2.30207417e-04, 0.00000000e+00,
       1.04372804e-04, 2.68180281e-04, 5.49483252e-04, 1.74408915e-04,
       0.00000000e+00, 0.00000000e+00, 1.04147241e-03, 1.39274440e-03,
       3.26936207e-04, 0.00000000e+00, 4.17958540e-05, 6.92234690e-04,
       3.92819096e-04, 2.21918905e-04, 1.24287122e-03, 1.32281229e-04,
       1.34287452e-03, 2.57734617e-04, 2.52874993e-04, 1.57779786e-04,
       2.25733089e-04, 9.03967382e-05, 1.49082912e-03, 7.59018236e-05,
       0.00000000e+00, 4.00916495e-04, 1.17595068e-03, 6.48611230e-05,
       1.61504299e-03, 8.80301886e-05, 0.00000000e+00, 2.54989282e-04,
       1.63625032e-03, 3.83271298e-04, 0.00000000e+00, 2.21952454e-03,
       1.23082487e-03, 1.32453155e-03, 1.13588314e-03, 1.68721515e-04,
       3.00531275e-04, 8.36335106e-04, 0.00000000e+00, 4.79212972e-05,
       1.39160343e-04, 1.89433128e-04, 1.71069540e-04, 9.69897974e-05,
       1.45496133e-04, 1.26034625e-04, 9.11923418e-05, 4.43061472e-04,
       4.19629830e-05, 0.00000000e+00, 1.00991665e-04, 2.73488932e-04,
       2.11139320e-04, 1.61547089e-03, 1.27659195e-03, 6.33237230e-05,
       8.69276633e-05, 6.12951294e-05, 4.13399931e-04, 1.13957110e-04,
       2.52341844e-04, 9.69393546e-04, 3.15389288e-03, 0.00000000e+00,
       1.48335961e-03, 1.27610585e-03, 1.48399363e-04, 1.22851021e-02,
       9.27576963e-03, 1.46444604e-02, 8.46683973e-03, 2.29087110e-02,
       5.69086453e-03, 1.53271744e-02])
```

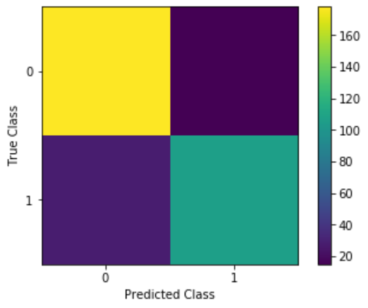
In [136]:
```
classreport(grid_rfc_age_fare_bin.best_estimator_,X_test)
```

```
MODEL RESULT
============
Confusion Matrix:
 [[178  15]
 [ 28 107]]

Classification Report:
               precision    recall  f1-score   support

           1       0.86      0.92      0.89       193
           0       0.88      0.79      0.83       135

   micro avg       0.87      0.87      0.87       328
   macro avg       0.87      0.86      0.86       328
weighted avg       0.87      0.87      0.87       328
```



Feature Selection

In [137]:
```
select=SelectFromModel(grid_rfc_age_fare_bin.best_estimator_)
select.fit(X_train,y_train)
```

Out[137]:
```
SelectFromModel(estimator=RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=10, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=500, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False),
        max_features=None, norm_order=1, prefit=False, threshold=None)
```

In [138]:
```
bool_mask=select.get_support()
```

In [139]:
```
data_age_fare_bin.columns
```

Out[139]:
```
Index(['Pclass_1', 'Pclass_2', 'Pclass_3', 'Sex_female', 'Sex_male',
       'Age_10-20', 'Age_20-30', 'Age_30-40', 'Age_40-50', 'Age_50-60',
       ...
       'Cabin_F4', 'Cabin_G6', 'Cabin_T', 'Embarked_C', 'Embarked_Q',
       'Embarked_S', 'Fare_AVERAGE', 'Fare_CHEAP', 'Fare_EXPENSIVE',
       'Fare_VERY EXPENSIVE'],
      dtype='object', length=222)
```

In [140]:
```
selected_features=[]
for i in range(222):
    if bool_mask[i]==True:
        selected_features.append(data_age_fare_bin.columns[i])
```

In [141]:
```
selected_features
```

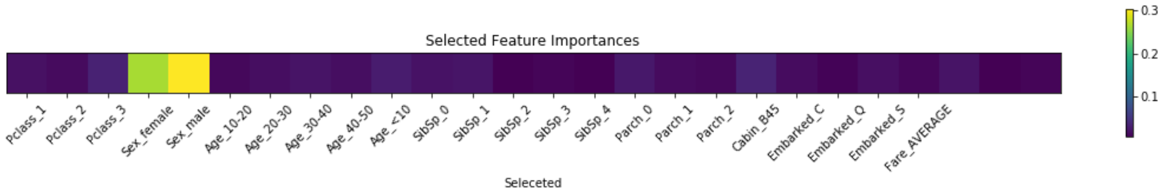
Out[141]:
```
['Pclass_1',
 'Pclass_2',
 'Pclass_3',
 'Sex_female',
 'Sex_male',
 'Age_10-20',
 'Age_20-30',
 'Age_30-40',
 'Age_40-50',
 'Age_<10',
 'SibSp_0',
 'SibSp_1',
 'SibSp_2',
 'SibSp_3',
 'SibSp_4',
 'Parch_0',
 'Parch_1',
 'Parch_2',
 'Cabin_B45',
 'Embarked_C',
 'Embarked_Q',
 'Embarked_S',
 'Fare_AVERAGE',
 'Fare_CHEAP',
 'Fare_EXPENSIVE',
 'Fare_VERY EXPENSIVE']
```

In [142]:
```
X_train_rfc_age_fare_bin=select.transform(X_train)
X_test_rfc_age_fare_bin=select.transform(X_test)
```

In [143]:
```python
grid_rfc_age_fare_bin.best_estimator_.fit(X_train_rfc_age_fare_bin,y_train)
print(
    'Test Score:\n',
    grid_rfc_age_fare_bin.best_estimator_.score(X_test_rfc_age_fare_bin,y_test),
    '\nTrain Score:\n',
    grid_rfc_age_fare_bin.best_estimator_.score(X_train_rfc_age_fare_bin,y_train)
)
```

```
Test Score:
 0.8536585365853658
Train Score:
 0.9143730886850153
```

In [144]:
```python
plt.figure(figsize=(20,2))
plt.title("Selected Feature Importances")
plt.imshow(grid_rfc_age_fare_bin.best_estimator_.feature_importances_.reshape(1,-1))
plt.xticks((range(23)),selected_features,rotation=45)
plt.yticks(())
plt.xlabel("Seleceted")
plt.colorbar()
plt.show()
```



In [145]:
```python
classreport(grid_rfc_age_fare_bin.best_estimator_,X_test_rfc_age_fare_bin)
```

```
MODEL RESULT
============
Confusion Matrix:
 [[172  21]
 [ 27 108]]

Classification Report:
               precision    recall  f1-score   support

           1       0.86      0.89      0.88       193
           0       0.84      0.80      0.82       135

   micro avg       0.85      0.85      0.85       328
   macro avg       0.85      0.85      0.85       328
weighted avg       0.85      0.85      0.85       328
```