In [134]:
```
import numpy as np
import pandas as pd
import seaborn as sb
%matplotlib inline
import matplotlib.pyplot as plt
import sklearn
from pandas import Series, DataFrame
from pylab import rcParams
from sklearn import preprocessing
from sklearn.linear_model import LogisticRegression
from sklearn.cross_validation import train_test_split
from sklearn import metrics
from sklearn.metrics import classification_report
```

In [135]:
```
link = 'https://raw.githubusercontent.com/BigDataGal/Python-for-Data-S
cience/master/titanic-train.csv'
titanic = pd.read_csv(link)
titanic.columns = ['PassengerId','Survived','Pclass','Name','Sex','Age
','SibSp','Parch','Ticket','Fare','Cabin','Embarked']
titanic.head()
```

Out[135]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0

In [136]:

```
type(titanic)
```

Out[136]: pandas.core.frame.DataFrame

In [137]:
```
titanic.isnull().sum()
```

Out[137]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

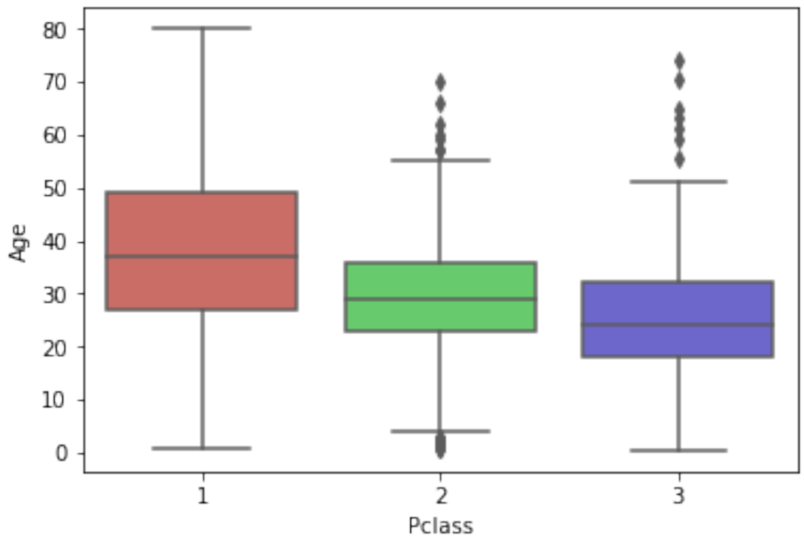
In [138]:
```
titanic = titanic.drop(['PassengerId','Name','Ticket','Cabin','SibSp']
, 1)
titanic.head()
# Since Passenger ID, Name, Ticket, cabin is not affecting the respons
e value, hence we are dropping them.
```

Out[138]:

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

In [139]:
```
sb.boxplot(x='Pclass', y='Age', data=titanic, palette='hls')
```

Out[139]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1c1875c0>



In [140]:
```
def age_approx(cols):
    Age = cols[0]
    Pclass = cols[1]
    if pd.isnull(Age):
        if Pclass == 1:
            return 37
        elif Pclass == 2:
            return 29
        else:
            return 24
    else:
            return Age
titanic['Age'] = titanic[['Age', 'Pclass']].apply(age_approx, axis=1)
titanic.isnull().sum()
```

Out[140]:
```
Survived    0
Pclass      0
Sex         0
Age         0
Parch       0
Fare        0
Embarked    2
dtype: int64
```

In [141]:
```
titanic.dropna(inplace=True)
titanic.isnull().sum()
```

Out[141]:
```
Survived    0
Pclass      0
Sex         0
Age         0
Parch       0
Fare        0
Embarked    0
dtype: int64
```

In [142]:
```
# Now Converting Categorical Variables into Dummy Variables
gender = pd.get_dummies(titanic['Sex'],drop_first=True)
gender.head()
```

Out[142]:

	male
0	1
1	0
2	0
3	0
4	1

In [143]:
```
embark = pd.get_dummies(titanic['Embarked'],drop_first=True)
embark.head()
```

Out[143]:

	Q	S
0	0	1
1	0	0
2	0	1
3	0	1
4	0	1

In [144]:
```
titanic.head()
```

Out[144]:

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

In [145]:
```
titanic.drop(['Sex', 'Embarked'],axis=1,inplace=True)
titanic.head()
```

Out[145]:

	Survived	Pclass	Age	Parch	Fare
0	0	3	22.0	0	7.2500
1	1	1	38.0	0	71.2833
2	1	3	26.0	0	7.9250
3	1	1	35.0	0	53.1000
4	0	3	35.0	0	8.0500

In [146]:
```
titanicD = pd.concat([titanic,gender,embark],axis=1)
titanicD.head()
```

Out[146]:

	Survived	Pclass	Age	Parch	Fare	male	Q	S
0	0	3	22.0	0	7.2500	1	0	1
1	1	1	38.0	0	71.2833	0	0	0
2	1	3	26.0	0	7.9250	0	0	1
3	1	1	35.0	0	53.1000	0	0	1
4	0	3	35.0	0	8.0500	1	0	1

In [147]:
```
titanicD.corr()
```

Out[147]:

	Survived	Pclass	Age	Parch	Fare	male	Q
Survived	1.000000	-0.335549	-0.052051	0.083151	0.255290	-0.541585	0.004536
Pclass	-0.335549	1.000000	-0.405549	0.016824	-0.548193	0.127741	0.220558
Age	-0.052051	-0.405549	1.000000	-0.170089	0.120938	0.083730	-0.080875
Parch	0.083151	0.016824	-0.170089	1.000000	0.217532	-0.247508	-0.081585
Fare	0.255290	-0.548193	0.120938	0.217532	1.000000	-0.179958	-0.116684
male	-0.541585	0.127741	0.083730	-0.247508	-0.179958	1.000000	-0.075217
Q	0.004536	0.220558	-0.080875	-0.081585	-0.116684	-0.075217	1.000000
S	-0.151777	0.076466	0.013598	0.061512	-0.163758	0.121405	-0.499261

In [148]:
```
titanicD.drop(['Fare', 'Pclass'],axis=1,inplace=True)
titanicD.head()
```

Out[148]:

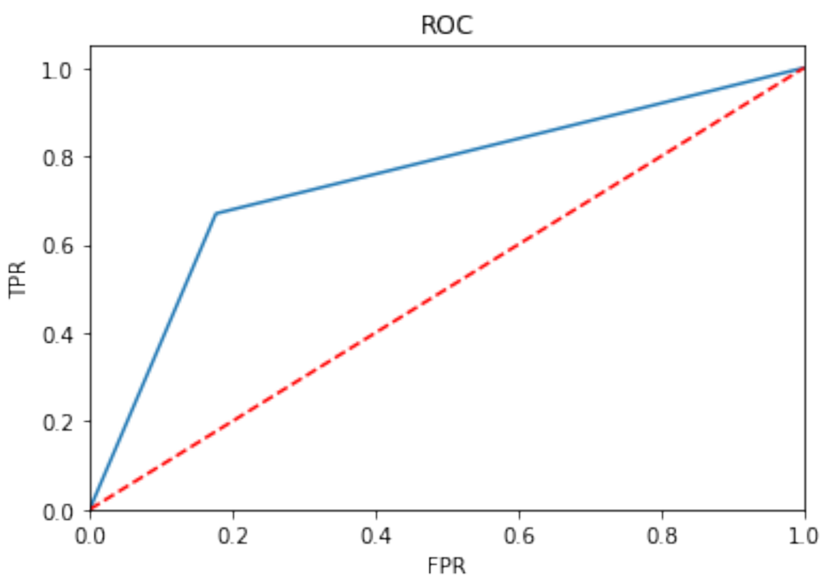
	Survived	Age	Parch	male	Q	S
0	0	22.0	0	1	0	1
1	1	38.0	0	0	0	0
2	1	26.0	0	0	0	1
3	1	35.0	0	0	0	1
4	0	35.0	0	1	0	1

In [161]:
```
X = titanicD.ix[:,(1,2,3,4,5)].values
y = titanicD.ix[:,0].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
.3, random_state=25)
```

In [172]:
```
#Using L1
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score
from sklearn.cross_validation import train_test_split,KFold
from sklearn.linear_model import LinearRegression, Lasso, Ridge
parameter1 =[0.0001,0.0002,0.0005,0.001,0.005,0.01,0.05, 0.1,0.5,1, 10
, 100,500,1000]
auc_cv=[]
for a in parameter1:
    logr=LogisticRegression(C=a,penalty="l1",class_weight="balanced",r
andom_state=2)
    k = KFold(len(X_train), n_folds=11)
    score=0
    for train, test in k:
        logr.fit(X_train[train], y_train[train])
        score+=roc_auc_score(y_train,logr.predict(X_train))
    auc_cv.append(score/10)
    print('{:.3f}\t {:.5f}\t '.format(a,score/10))
parameter1=np.array(parameter1)
auc_cv=np.array(auc_cv)
c_best=parameter1[auc_cv==max(auc_cv)][0]
print("The Value of C Best=",c_best)
y_pred= logr.predict(X_test)
from sklearn.metrics import confusion_matrix
cmp = confusion_matrix(y_test, y_pred)
print("Confusion Matrix \n",cmp)
```

```
0.000    0.55000
0.000    0.55000
0.001    0.55000
0.001    0.55000
0.005    0.55000
0.010    0.55000
0.050    0.84868
0.100    0.85166
0.500    0.85166
1.000    0.85187
10.000   0.85429
100.000  0.85506
500.000  0.85506
1000.000        0.85506
The Value of C Best= 100.0
Confusion Matrix
 [[135  29]
 [ 34  69]]
```

In [173]:
```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_curve, auc
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % max(au
c_cv))
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC')
plt.show()
print("AUC Ridge = ",max(auc_cv)*100)
```

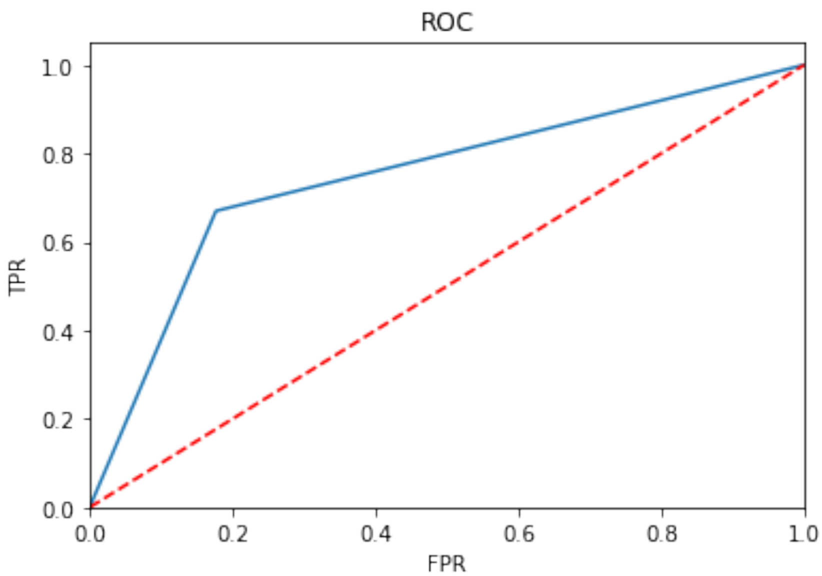



```
AUC Ridge =  85.5055071511
```

In [174]:
```
#Using L2
parameter1 =[0.0001,0.0002,0.0005,0.001,0.005,0.01,0.05, 0.1,0.5,1, 10
, 100,500,1000]
auc_cv_1=[]
for a in parameter1:
    logreg=LogisticRegression(C=a,penalty="l2",class_weight="balanced"
,random_state=2)
    k = KFold(len(X_train), n_folds=10)
    score=0
    for train, test in k:
        logreg.fit(X_train[train], y_train[train])
        score+=roc_auc_score(y_train,logreg.predict(X_train))
    auc_cv_1.append(score/10)
    print('{:.3f}\t {:.5f}\t '.format(a,score/10))
parameter1=np.array(parameter1)
auc_cv_1=np.array(auc_cv_1)
c_best=parameter1[auc_cv_1==max(auc_cv_1)][0]
print("The Value of C Best=",c_best)
y_pred= logreg.predict(X_test)
from sklearn.metrics import confusion_matrix
cmp = confusion_matrix(y_test, y_pred)
print("Confusion Matrix \n",cmp)
```

```
0.000    0.50798
0.000    0.51417
0.001    0.53075
0.001    0.58308
0.005    0.76934
0.010    0.76971
0.050    0.77216
0.100    0.77411
0.500    0.77424
1.000    0.77424
10.000   0.77463
100.000  0.77593
500.000  0.77593
1000.000        0.77593
The Value of C Best= 100.0
Confusion Matrix
 [[135  29]
 [ 34  69]]
```

In [175]:
```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_curve, auc
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % max
(auc_cv_1))
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC')
plt.show()
print("AUC Lasso = ",max(auc_cv_1)*100)
```




```
AUC Lasso =  77.5926352129
```

In [176]:
```
print(classification_report(y_test, y_pred))
```

```
             precision    recall  f1-score   support

          0       0.80      0.82      0.81       164
          1       0.70      0.67      0.69       103

avg / total       0.76      0.76      0.76       267
```

In [177]:
```
import pandas as pd
titanic3 = [['AUC Ridge ',max(auc_cv)*100],['AUC Lasso ',max(auc_cv_1)
*100]]
df = pd.DataFrame(titanic3,columns=['Loss Method','Accuracy'])
print(df)
```

```
  Loss Method   Accuracy
0  AUC Ridge   85.505507
1  AUC Lasso   77.592635
```

In [79]:
```
# From the above analysis in the table, we see that L1 Loss method has
high accuracy as compared to L2 Loss method.
```