Machine Learning - Logistic Regression, Linear Discriminant and Quadratic Discriminant Analysis (Titanic data set)

In [1]:
```
#Import of packages
import numpy as np
```

In [2]:
```
import pandas as pd
```

In [3]:
```
import matplotlib.pyplot as plt #for graphing
```

In [4]:
```
import seaborn as sb #for graphing
```

In [5]:
```
# Import of Scikit-Learn
import sklearn
```

In [6]:
```
from sklearn import preprocessing
```

In [ ]:
```
from sklearn.model_selection import train_test_split
```

In [7]:
```
from sklearn.linear_model import LogisticRegression
```

In [9]:
```
from sklearn import metrics
```

In [10]:
```
from sklearn.metrics import classification_report
```

In [11]:
```
from sklearn.metrics import accuracy_score
```

In [12]:
```
%matplotlib inline
```

In [ ]:
```

```

Read-in data, analyse data

In [13]:
```
train = pd.read_csv('C:/Bilder/train.csv')
```

In [14]:
```
train.head()
```

Out[14]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Ca
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	N
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	(
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	N
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	N

In [15]:
```
# Rows and columns
train.shape
```

Out[15]: (891, 12)

In [16]:
```
# Descriptive statistics
train.describe()
```

Out[16]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [17]:
```
train.dtypes
```

Out[17]:
```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
dtype: object
```

In [18]:
```
# Find out how many missing values are there
train.isnull().sum()
```

Out[18]:
```
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
```

Pre-processing

Survived is y-variable

In [19]:
```
train.drop(['PassengerId', 'Name', 'Ticket', 'Cabin'], axis=1, inplace=True)
train.head()
```

Out[19]:

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

In [20]:
```
train.Pclass.value_counts()
```

Out[20]:
```
Pclass
3    491
1    216
2    184
Name: count, dtype: int64
```

In [21]:
```
train.Pclass = train.Pclass.apply(str)
```

In [22]:
```
train.dtypes
```

Out[22]:
```
Survived      int64
Pclass       object
Sex          object
Age         float64
SibSp         int64
Parch         int64
Fare        float64
Embarked     object
dtype: object
```

In [23]:
```
train = pd.get_dummies(train, prefix_sep='_', drop_first=True, dtype=float)
```

In [24]:
```
train.head()
```

Out[24]:

	Survived	Age	SibSp	Parch	Fare	Pclass_2	Pclass_3	Sex_male	Embarked_Q	Embarke
0	0	22.0	1	0	7.2500	0.0	1.0	1.0	0.0	
1	1	38.0	1	0	71.2833	0.0	0.0	0.0	0.0	
2	1	26.0	0	0	7.9250	0.0	1.0	0.0	0.0	
3	1	35.0	1	0	53.1000	0.0	0.0	0.0	0.0	
4	0	35.0	0	0	8.0500	0.0	1.0	1.0	0.0	

In [25]:
```
train.isnull().sum()
```

Out[25]:
```
Survived        0
Age           177
SibSp           0
Parch           0
Fare            0
Pclass_2        0
Pclass_3        0
Sex_male        0
Embarked_Q      0
Embarked_S      0
dtype: int64
```

In [26]:
```python
train.dtypes
```

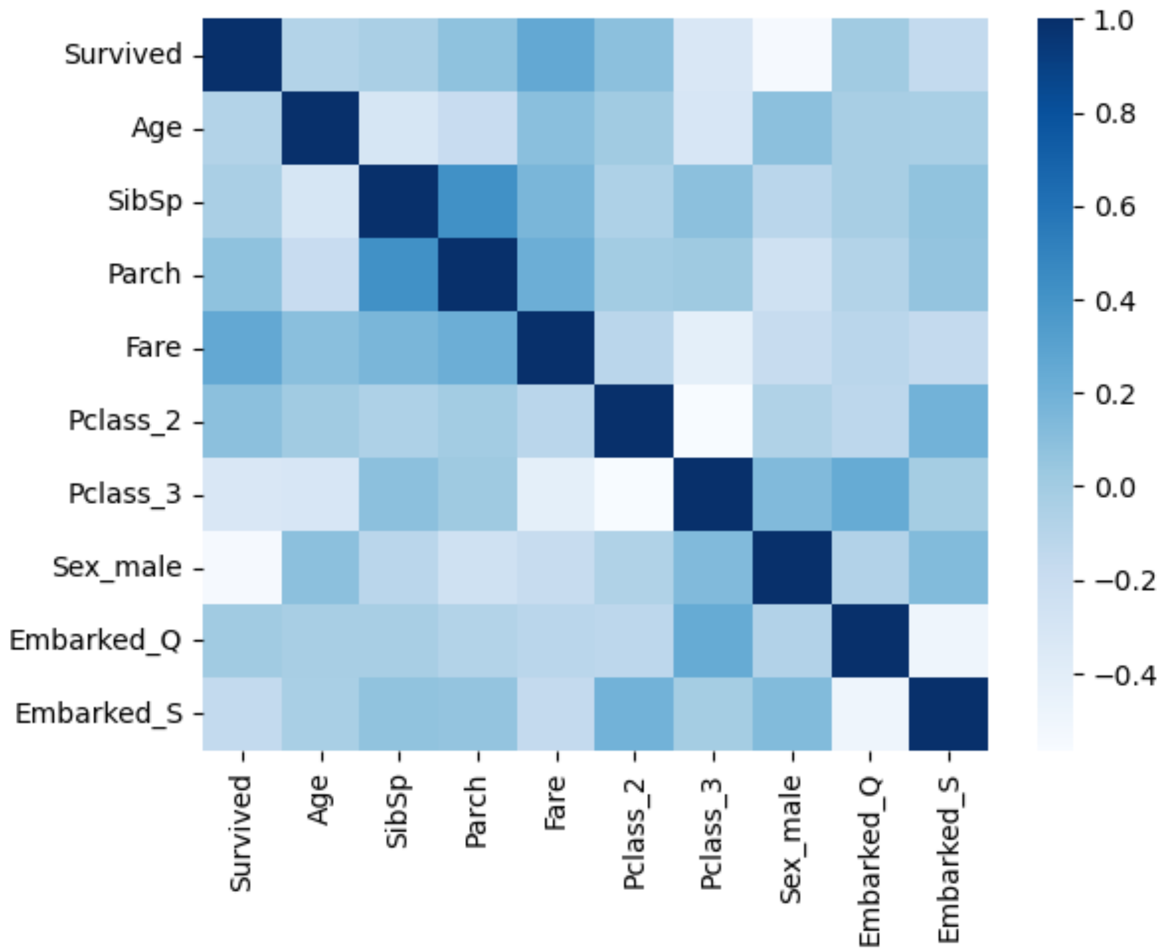
Out[26]:
```
Survived        int64
Age           float64
SibSp           int64
Parch           int64
Fare          float64
Pclass_2      float64
Pclass_3      float64
Sex_male      float64
Embarked_Q    float64
Embarked_S    float64
dtype: object
```

In [27]:
```python
train.corr()
```

Out[27]:

	Survived	Age	SibSp	Parch	Fare	Pclass_2	Pclass_3	Sex_mal
Survived	1.000000	-0.077221	-0.035322	0.081629	0.257307	0.093349	-0.322308	-0.54335
Age	-0.077221	1.000000	-0.308247	-0.189119	0.096067	0.006954	-0.312271	0.09325
SibSp	-0.035322	-0.308247	1.000000	0.414838	0.159651	-0.055932	0.092548	-0.11463
Parch	0.081629	-0.189119	0.414838	1.000000	0.216225	-0.000734	0.015790	-0.24548
Fare	0.257307	0.096067	0.159651	0.216225	1.000000	-0.118557	-0.413333	-0.18233
Pclass_2	0.093349	0.006954	-0.055932	-0.000734	-0.118557	1.000000	-0.565210	-0.06474
Pclass_3	-0.322308	-0.312271	0.092548	0.015790	-0.413333	-0.565210	1.000000	0.13714
Sex_male	-0.543351	0.093254	-0.114631	-0.245489	-0.182333	-0.064746	0.137143	1.00000
Embarked_Q	0.003650	-0.022405	-0.026354	-0.081228	-0.117216	-0.127301	0.237449	-0.07411
Embarked_S	-0.155660	-0.032523	0.070941	0.063036	-0.166603	0.192061	-0.009511	0.12572

In [28]:
```
sb.heatmap(train.corr(), cmap='Blues')
```
Out[28]: <Axes: >



Logistic Regression

In [ ]:

In [29]:
```

X = train
X = train.dropna()
```

In [30]:
```python
X.isnull().sum()
```

Out[30]:
```
Survived      0
Age           0
SibSp         0
Parch         0
Fare          0
Pclass_2      0
Pclass_3      0
Sex_male      0
Embarked_Q    0
Embarked_S    0
dtype: int64
```

In [31]:
```python

Y = X['Survived']
```

In [32]:
```python
Y.head()
```

Out[32]:
```
0    0
1    1
2    1
3    1
4    0
Name: Survived, dtype: int64
```

In [33]:
```python

X.pop('Survived')
X.columns
```

Out[33]:
```
Index(['Age', 'SibSp', 'Parch', 'Fare', 'Pclass_2', 'Pclass_3', 'Sex_male',
       'Embarked_Q', 'Embarked_S'],
      dtype='object')
```

In [34]:
```python
X.shape
```

Out[34]:
```
(714, 9)
```

In [35]:
```python

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = .25, ran
```

In [36]:
```python
Y_train.shape
```

Out[36]:
```
(535,)
```

In [37]:
```python
Y_test.shape
```

Out[37]:
```
(179,)
```

In [38]:
```python
X_train.shape
```

Out[38]:
```
(535, 9)
```

In [39]:
```
X_test.shape
```
Out[39]:
```
(179, 9)
```

In [77]:
```
# Model of Logistic Regression

logit = LogisticRegression()

#logit = LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercep
#intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
#penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
#verbose=0, warm_start=False)
```

In [78]:

```
# Fit the model to the training data
logit.fit(X_train, Y_train)
```

Out[78]:

```
▸ LogisticRegression
```

Predictions

In [43]:
```
Y_pred = logit.predict(X_test)
```

In [44]:
```
Y_pred
```

Out[44]:
```
array([1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1,
       0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0,
       1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1,
       0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0,
       0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 1], dtype=int64)
```

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

Out[45]:
```
(179,)
```

In [ ]:
```

```

In [46]:
```
Y_test.head(20) # true y's from test data sample
```

Out[46]:
```
501    0
315    1
679    1
866    1
119    0
254    0
885    0
155    0
169    0
683    0
515    0
192    1
724    1
393    1
523    1
467    0
659    0
663    0
777    1
805    0
Name: Survived, dtype: int64
```

For model evaluation: compare Y_test with Y_pred !

In [ ]:

In [47]:
```
logit.predict_proba(X_test)
```

Out[47]:
```
array([[0.32446883, 0.67553117],
       [0.45052784, 0.54947216],
       [0.20637139, 0.79362861],
       [0.21110215, 0.78889785],
       [0.56510698, 0.43489302],
       [0.52947364, 0.47052636],
       [0.35379372, 0.64620628],
       [0.53068132, 0.46931868],
       [0.87966909, 0.12033091],
       [0.96267455, 0.03732545],
       [0.60539672, 0.39460328],
       [0.48480578, 0.51519422],
       [0.51800229, 0.48199771],
       [0.06234054, 0.93765946],
       [0.08831463, 0.91168537],
       [0.68086671, 0.31913329],
       [0.54184576, 0.45815424],
       [0.91461323, 0.08538677],
       [0.28068212, 0.71931788],
```

Model Metrics

In [ ]:

In [48]:
```
Y_test.value_counts()
```

Out[48]:
```
Survived
0    103
1     76
Name: count, dtype: int64
```

Survived 0 103 1 76 Name: count, dtype: int64

In [49]:
```
Y_pred.shape
```

Out[49]:
```
(179,)
```

In [50]:
```
confusion_matrix = metrics.confusion_matrix(Y_test, Y_pred) # Confusion Matrix
print(confusion_matrix)
```

```
[[87 16]
 [21 55]]
```

In [51]:
```
print(metrics.classification_report(Y_test, Y_pred))
```

```
              precision    recall  f1-score   support

           0       0.81      0.84      0.82       103
           1       0.77      0.72      0.75        76

    accuracy                           0.79       179
   macro avg       0.79      0.78      0.79       179
weighted avg       0.79      0.79      0.79       179
```

In [52]:
```
print(metrics.accuracy_score(Y_test, Y_pred))
```

```
0.7932960893854749
```

Plotting the ROC:

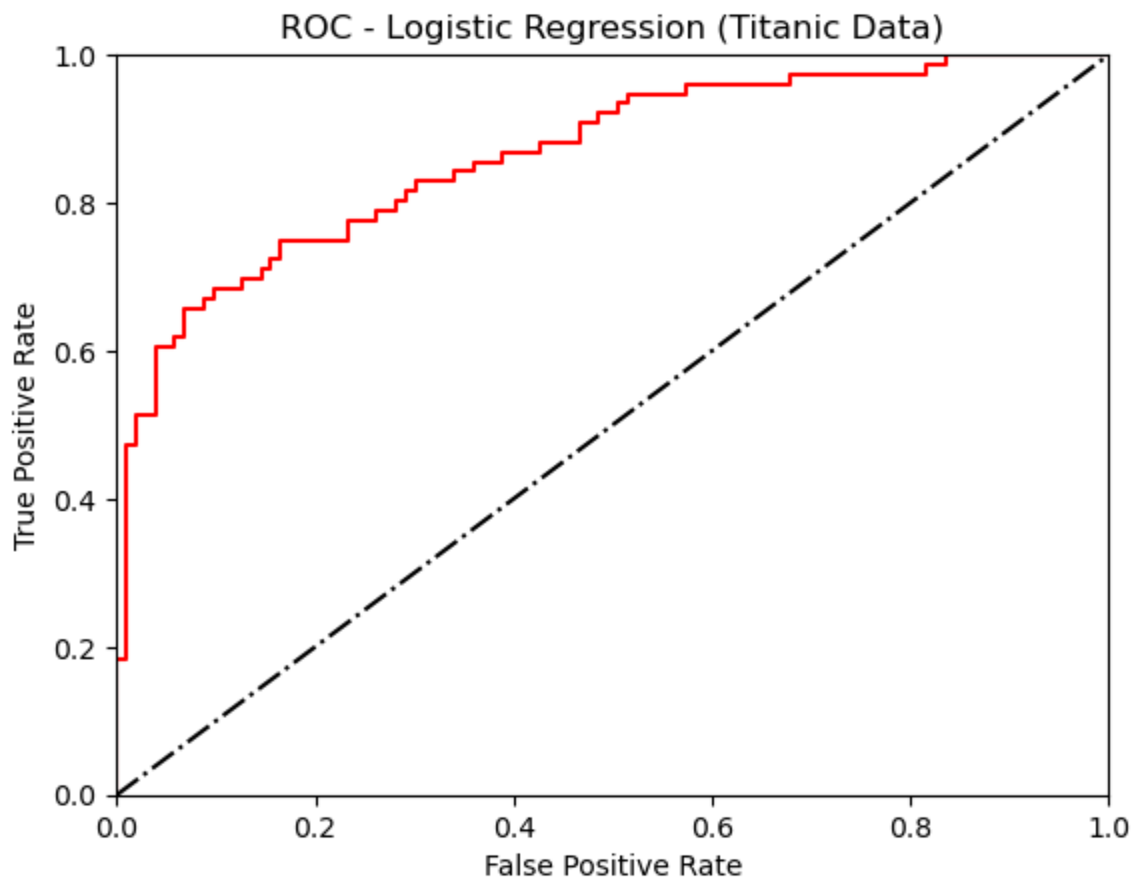
In [53]:
```
prob = logit.predict_proba(X_test)
```

In [54]:
```

pred = prob[:,1]
```

In [55]:
```
fpr, tpr, threshold = metrics.roc_curve(Y_test, pred)
```

In [57]:
```
plt.title('ROC - Logistic Regression (Titanic Data)')
plt.plot(fpr, tpr, 'r')
plt.plot([0, 1], [0, 1],'k-.')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```



In [56]:
```
metrics.roc_auc_score(Y_test, pred)
```

Out[56]: 0.8657383750638733

In [ ]:

Linear Discriminant Analysis

In [58]:
```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
```

In [59]:
```
lda = LinearDiscriminantAnalysis()
```

In [60]:
```
lda.fit(X_train, Y_train)
```
Out[60]:
```
▾ LinearDiscriminantAnalysis
LinearDiscriminantAnalysis()
```

In [61]:
```
Y_pred = lda.predict(X_test)
```

In [62]:
```
Y_pred
```
Out[62]:
```
array([1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1,
       0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0,
       1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1,
       0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0,
       0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 1], dtype=int64)
```

In [63]:
```
Y_pred.shape
```
Out[63]:
```
(179,)
```

In [64]:
```
confusion_matrix = metrics.confusion_matrix(Y_test, Y_pred) # Confusion Matrix
print(confusion_matrix)
```
```
[[82 21]
 [20 56]]
```

In [65]:
```
print(metrics.classification_report(Y_test, Y_pred))
```
```
              precision    recall  f1-score   support

           0       0.80      0.80      0.80       103
           1       0.73      0.74      0.73        76

    accuracy                           0.77       179
   macro avg       0.77      0.77      0.77       179
weighted avg       0.77      0.77      0.77       179
```

In [66]:
```
print(metrics.accuracy_score(Y_test, Y_pred))
```
```
0.770949720670391
```

In [ ]:
```

```

Quadratic Discriminant Analysis

In [67]:
```
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
```

In [68]:
```
qda = QuadraticDiscriminantAnalysis()
```

In [69]:
```
qda.fit(X_train, Y_train)
```

Out[69]:
```
▾ QuadraticDiscriminantAnalysis
QuadraticDiscriminantAnalysis()
```

In [70]:
```
Y_pred = qda.predict(X_test)
```

In [71]:
```
Y_pred
```

Out[71]:
```
array([1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1,
       0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0,
       1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
       0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0,
       0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 1], dtype=int64)
```

In [72]:
```
Y_pred.shape
```

Out[72]:
```
(179,)
```

In [73]:
```
confusion_matrix = metrics.confusion_matrix(Y_test, Y_pred) # Confusion Matrix
print(confusion_matrix)
```

```
[[83 20]
 [24 52]]
```

In [74]:
```
print(metrics.classification_report(Y_test, Y_pred))
```

```
              precision    recall  f1-score   support

           0       0.78      0.81      0.79       103
           1       0.72      0.68      0.70        76

    accuracy                           0.75       179
   macro avg       0.75      0.75      0.75       179
weighted avg       0.75      0.75      0.75       179
```

In [75]:
```
print(metrics.accuracy_score(Y_test, Y_pred))
```

```
0.7541899441340782
```

In [ ]:
```

```

In [ ]:

In [ ]: