深 圳 大 学 实 验 报 告

课程名称：　　机器学习

实验项目名称：　　**Machine Learning Task 2**

学院：　　电子与信息工程学院

专业：　　电子信息工程

指导教师：　　欧阳乐

报告人：　　余韦藩　　学号：　　**2020285102**

班级：　　文华班

实验时间：　　**2022.4.21 ——2022.5.22**

实验报告提交时间：　　**2022.5.23**

教务处制

Aim of Experiment:

(1) Familiar with the five typical classifiers.

(2) Compare the performance for five typical classifiers.

(3) Further understand the principle of five typical classifiers.

Experiment Content:

In the experiment, the datasets we used is breast cancer datasets loaded in the sklearn.datasets. The breast cancer datasets have thirty attributes and one class label. The class label includes two type which means that there are two type of breast cancer totally in the datasets. For the classification task, we want to correctly predict the class label according to the given value of 30 attributes and the task belongs to two-classification problem. In the experiment, we perform five typical classifiers including Linear Discriminant Analysis(LDA), Logistic Regression, k-nearest neighbors(KNN), Naive Bayes and Support Vector Machine(SVM) on the breast cancer datasets and evaluate the accuracy of specific classifier. Finally, we objectively compare the performance of five classifiers.

Experiment Process:

The whole process is shown in the following figure 1. The evaluation includes three blocks: figure, statistic data and table. Figure includes confusion matrix(heatmap) and ROC curve. Statistic data includes classification accuracy, classification error, precision, recall or sensitivity, true positive rate, false positive rate, specificity and AUC. Table includes classification report. For each classifier, we perform the same procedure.

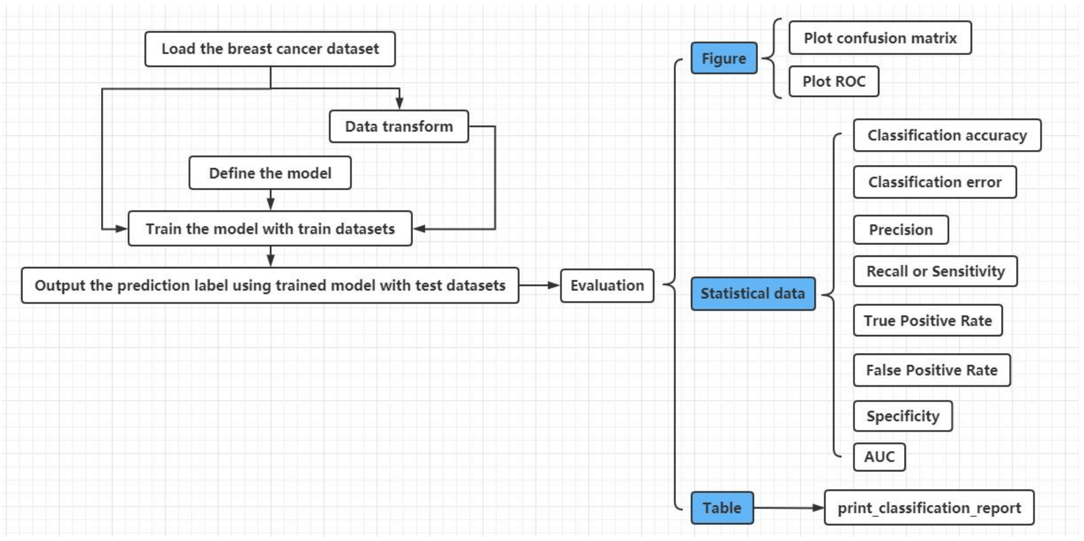


Figure 1: The whole process for each classifier

Data Logging and Processing:

The related codes have been attached to the appendix. In the experiment, we apply LDA model, KNN with k=9 model, Logistic Regression model, GaussianNB model, BernoulliNB model, MultinomialNB model, SVM with rbf kernel and c=100, SVM with rbf kernel and c=1000, SVM with linear kernel and c=1, SVM with linear kernel and c=100, SVM with linear kernel and c=1000, SVM with

polynomial kernel and c=1, SVM with polynomial kernel and c=100, SVM with sigmoid kernel and c=1 and SVM with sigmoid kernel and c=100.
At the beginning, we first briefly analyze the definition of each evaluation item.

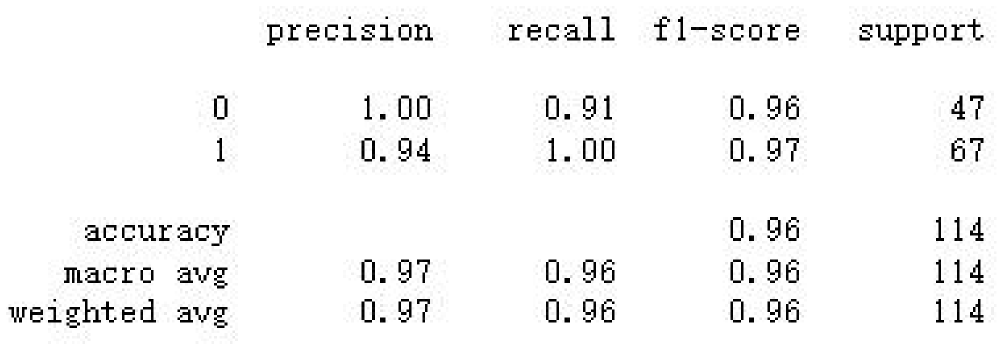
AUC: For this evaluation, we calculate the area under the ROC curve as the value of ROC. A perfect classifier will have a AUC equal to 1, whereas a purely random classifier will have a AUC equal to 0.5.
Classification accuracy: The classification accuracy is defined as (TP + TN) / (TP + TN + FP + FN)
Classification error: The classification error is defined as (FP + FN) / (TP + TN + FP + FN) or is equal to 1-classification accuracy.
Precision: Precision can be defined as the percentage of correctly predicted positive outcomes out of all the predicted positive outcomes. It can be given as the ratio of true positives (TP) to the sum of true and false positives (TP + FP).
So, Precision identifies the proportion of correctly predicted positive outcome. It is more concerned with the positive class than the negative class.
Mathematically, precision can be defined as the ratio of TP to (TP + FP).
Recall or sensitivity: Recall can be defined as the percentage of correctly predicted positive outcomes out of all the actual positive outcomes. It can be given as the ratio of true positives (TP) to the sum of true positives and false negatives (TP + FN). Recall is also called Sensitivity.
Recall identifies the proportion of correctly predicted actual positives.
Mathematically, recall can be defined as the ratio of TP to (TP + FN).
True positive rate: True Positive Rate is synonymous with Recall and is defined as the ratio TP to (TP + FN).
False positive rate: False positive rate is defined as FP / float(FP + TN).
Specificity: The specificity is defined as TN / (TN + FP)

Experimental Results and Analysis:

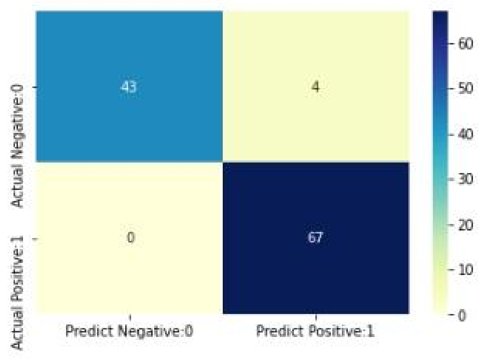
In convince to compare the statistic data with the different classifier, we summary all the statistic data into two table at last and plot the corresponding figure respectively at first.

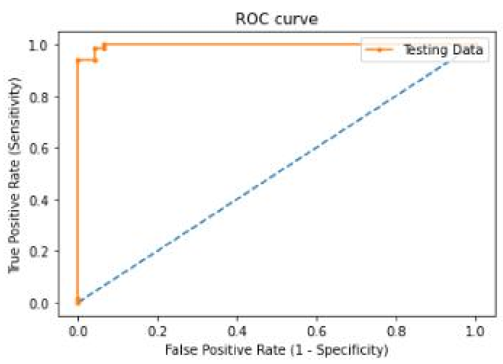
A. LDA model

The classification report, heatmap and ROC curve are shown in the following figure.



```
              precision    recall  f1-score   support

           0       1.00      0.91      0.96        47
           1       0.94      1.00      0.97        67

    accuracy                           0.96       114
   macro avg       0.97      0.96      0.96       114
weighted avg       0.97      0.96      0.96       114
```

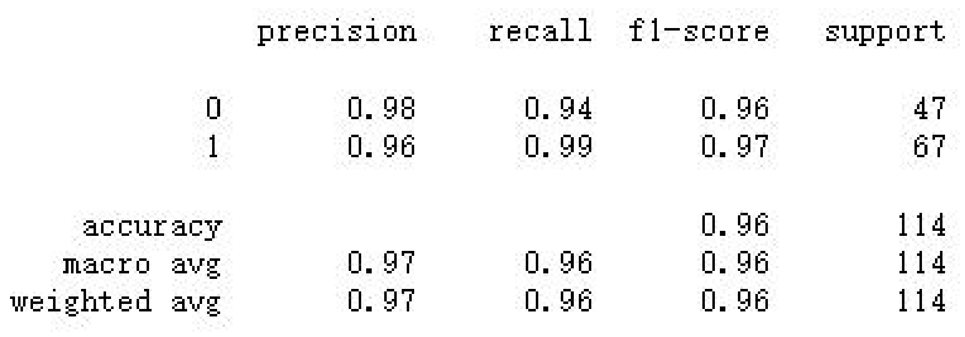




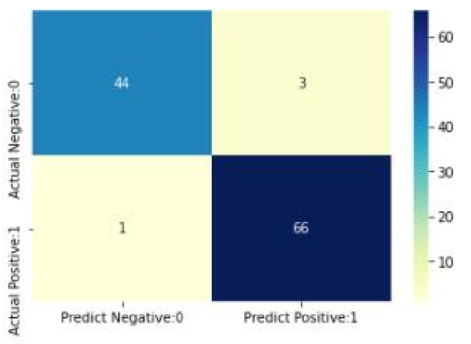
From the figure, we can calculate that TN=43, FP=4, FN=0 and TP=67.

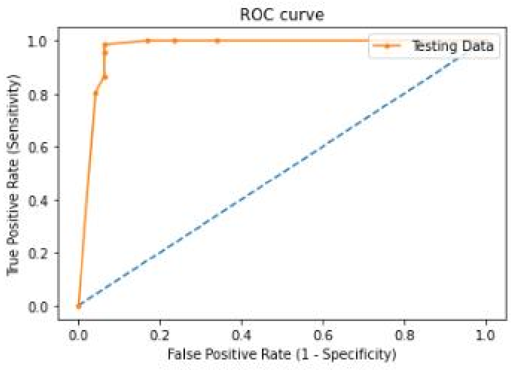
B. KNN with k=9 model

The classification report, heatmap and ROC curve are shown in the following figure.



```
              precision    recall  f1-score   support

           0       0.98      0.94      0.96        47
           1       0.96      0.99      0.97        67

    accuracy                           0.96       114
   macro avg       0.97      0.96      0.96       114
weighted avg       0.97      0.96      0.96       114
```

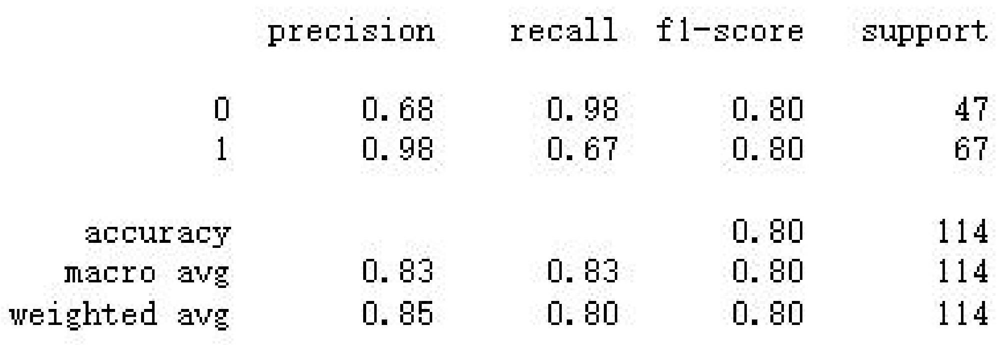


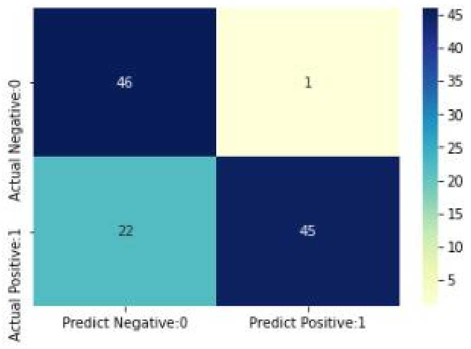


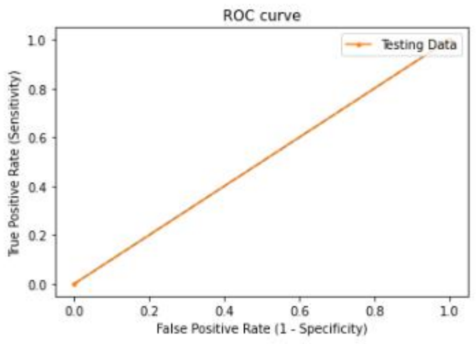
From the figure, we can calculate that TN=44, FP=3, FN=1 and TP=66.

C. Logistic Regression model

The classification report, heatmap and ROC curve are shown in the following figure.


```
              precision    recall  f1-score   support

           0       0.68      0.98      0.80        47
           1       0.98      0.67      0.80        67

    accuracy                           0.80       114
   macro avg       0.83      0.83      0.80       114
weighted avg       0.85      0.80      0.80       114
```

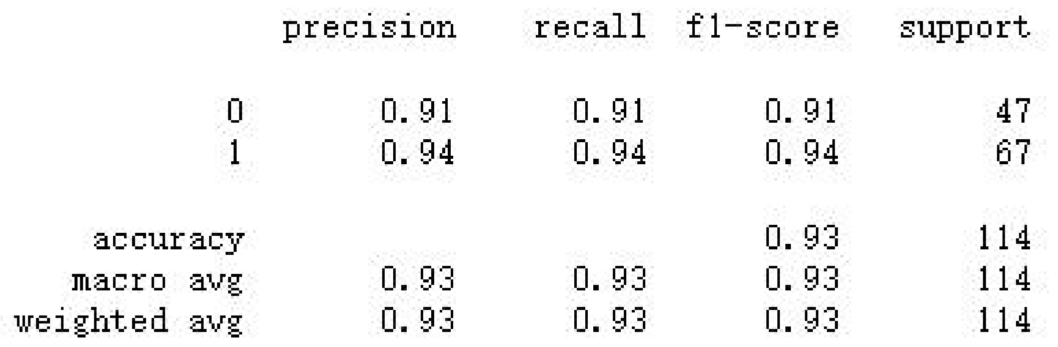




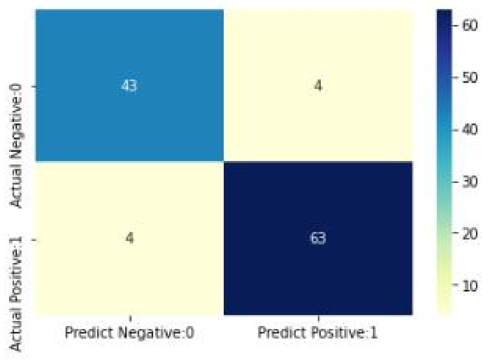
From the figure, we observe that the AUC is equal to 0.5 which means that the logistic regression model purely random predicts when applying in the breast cancer datasets. Also, we can calculate that TN=46, FP=1, FN=22 and TP=45.

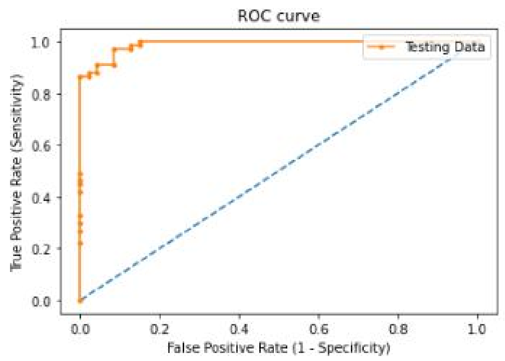
D. GaussianNB model

The classification report, heatmap and ROC curve are shown in the following figure.



```
              precision    recall  f1-score   support

           0       0.91      0.91      0.91        47
           1       0.94      0.94      0.94        67

    accuracy                           0.93       114
   macro avg       0.93      0.93      0.93       114
weighted avg       0.93      0.93      0.93       114
```

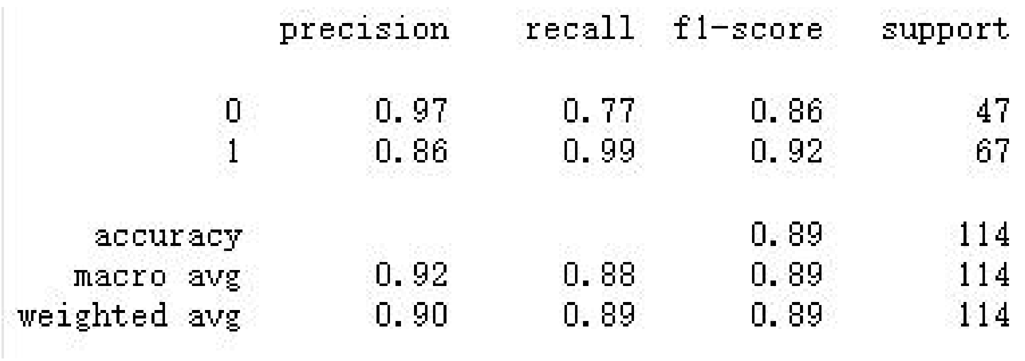




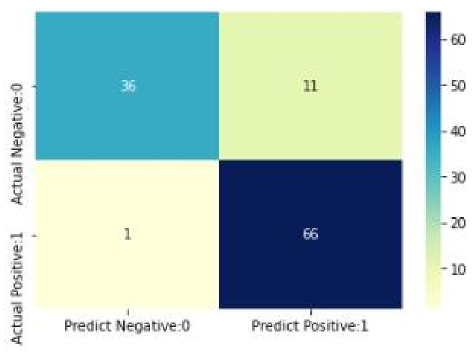
From the figure, we can calculate that TN=43, FP=4, FN=4 and TP=63.

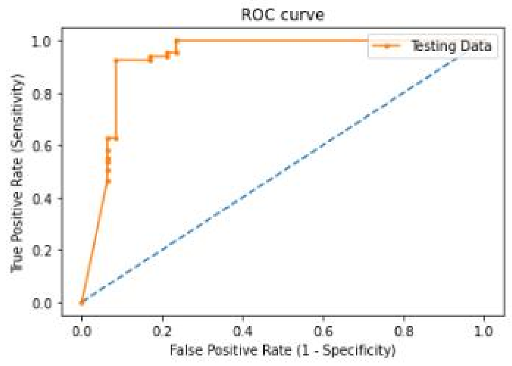
E. BernoulliNB model

The classification report, heatmap and ROC curve are shown in the following figure.



```
              precision    recall  f1-score   support

           0       0.97      0.77      0.86        47
           1       0.86      0.99      0.92        67

    accuracy                           0.89       114
   macro avg       0.92      0.88      0.89       114
weighted avg       0.90      0.89      0.89       114
```



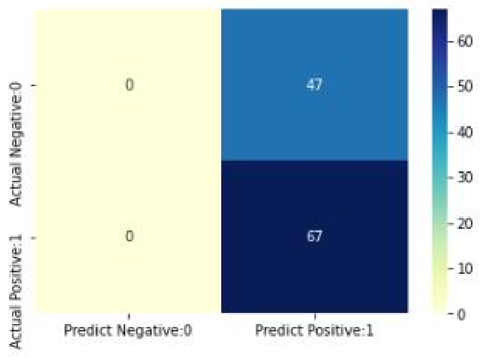


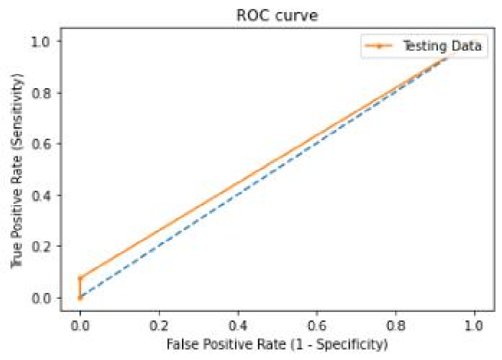
From the figure, we can calculate that TN=36, FP=11, FN=1 and TP=66.

F. MultinomialNB model

The classification report, heatmap and ROC curve are shown in the following figure.

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        47
           1       0.59      1.00      0.74        67

    accuracy                           0.59       114
   macro avg       0.29      0.50      0.37       114
weighted avg       0.35      0.59      0.44       114
```





From the figure, we can calculate that TN=0, FP=47, FN=0 and TP=67.

Notice that there exists some 0 value in the classification report and when we calculate the AUC, it raise error. The detail of the error is shown in the following figure. It reports that there is only one class present in y_pred so we print the all the value of y_pred.

```
     79         check_consistent_length(y_true, y_score, sample_weight)

~\Anaconda3\lib\site-packages\sklearn\metrics\_ranking.py in _binary_roc_auc_score(y_true, y_score, sample_weigh
t, max_fpr)
    325     """Binary roc auc score."""
    326     if len(np.unique(y_true)) != 2:
--> 327         raise ValueError("Only one class present in y_true. ROC AUC score "
    328                          "is not defined in that case.")
    329

ValueError: Only one class present in y_true. ROC AUC score is not defined in that case.
```

All the value of the y_pred is shown in the following figure. We observe that all the predicted label is equal to 1 and no 0 label which proves that there is only one class present in y_pred so raise the error when calculate the AUC.

```
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1]
```

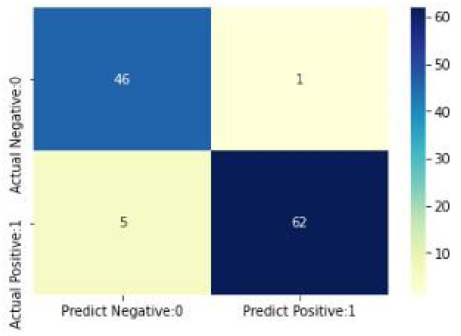
What' more, when print the classification report, in order to prevent dividing 0, some value such as precision,recall and f1-score in the report are directly set to 0.

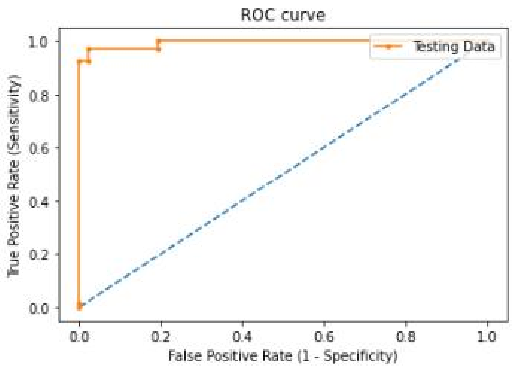
```
C:\Users\28291\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1245: UndefinedMetricWarning: Precision and F-sc
ore are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this be
havior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\28291\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1245: UndefinedMetricWarning: Precision and F-sc
ore are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this be
havior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\28291\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1245: UndefinedMetricWarning: Precision and F-sc
ore are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this be
havior.
  _warn_prf(average, modifier, msg_start, len(result))
```

G. SVM with rbf kernel and c=100

The classification report, heatmap and ROC curve are shown in the following figure.

```
              precision    recall  f1-score   support

           0       0.90      0.98      0.94        47
           1       0.98      0.93      0.95        67

    accuracy                           0.95       114
   macro avg       0.94      0.95      0.95       114
weighted avg       0.95      0.95      0.95       114
```



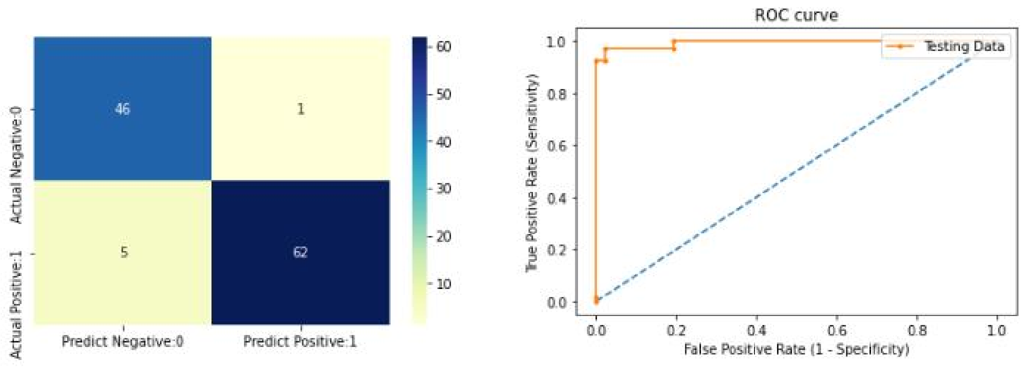


From the figure, we can calculate that TN=46, FP=1, FN=5 and TP=62.

H. SVM with rbf kernel and c=1000

The classification report, heatmap and ROC curve are shown in the following figure.

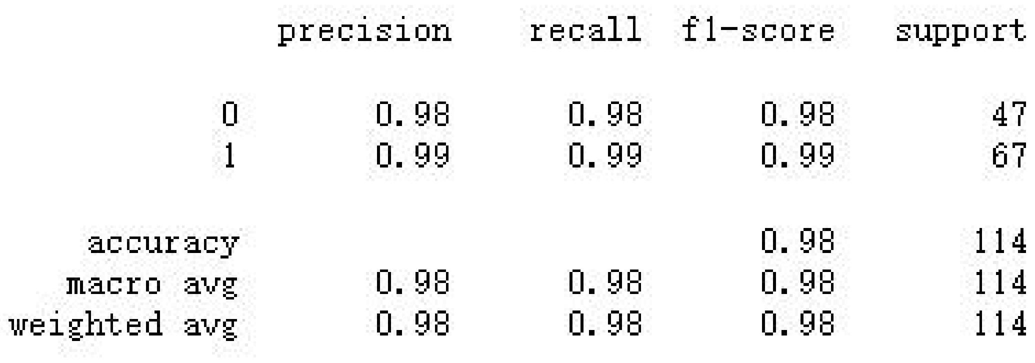
```
              precision    recall  f1-score   support

           0       0.90      0.98      0.94        47
           1       0.98      0.93      0.95        67

    accuracy                           0.95       114
   macro avg       0.94      0.95      0.95       114
weighted avg       0.95      0.95      0.95       114
```



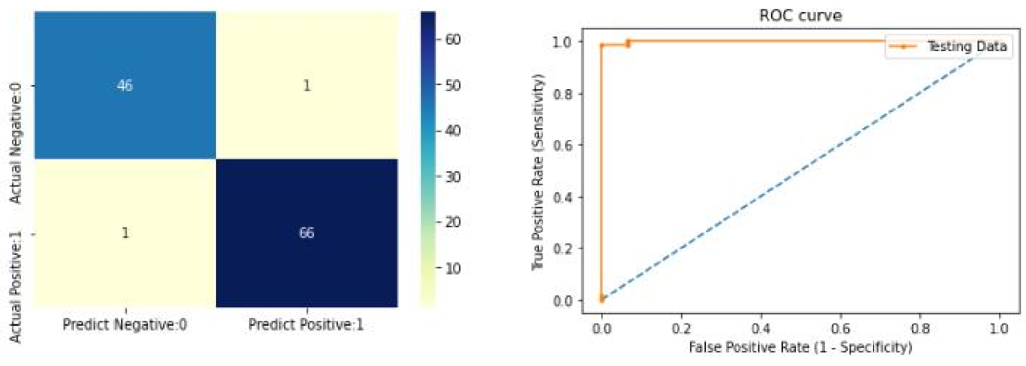
From the figure, we can calculate that TN=46, FP=1, FN=5 and TP=62.

I. SVM with linear kernel and c=1

The classification report, heatmap and ROC curve are shown in the following figure.



```
              precision    recall  f1-score   support

           0       0.98      0.98      0.98        47
           1       0.99      0.99      0.99        67

    accuracy                           0.98       114
   macro avg       0.98      0.98      0.98       114
weighted avg       0.98      0.98      0.98       114
```

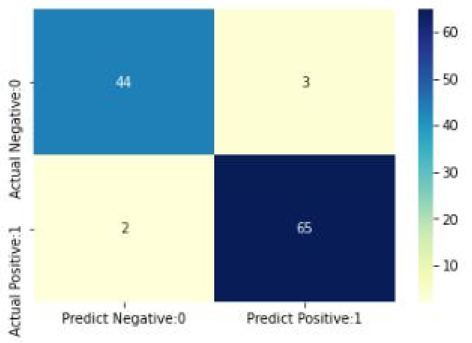


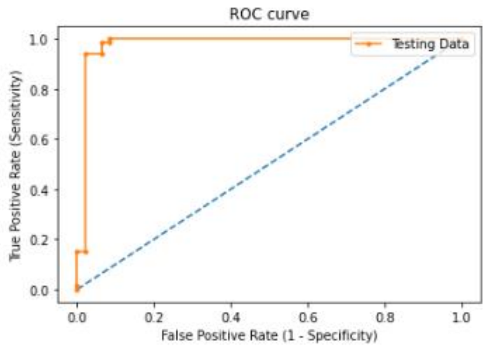
From the figure, we can calculate that TN=46, FP=1, FN=1 and TP=66.

J. SVM with linear kernel and c=100

The classification report, heatmap and ROC curve are shown in the following figure.

```
              precision    recall  f1-score   support

           0       0.96      0.94      0.95        47
           1       0.96      0.97      0.96        67

    accuracy                           0.96       114
   macro avg       0.96      0.95      0.95       114
weighted avg       0.96      0.96      0.96       114
```



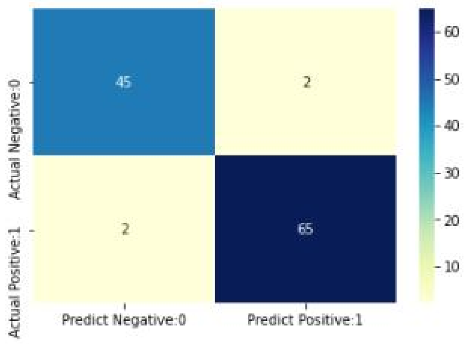


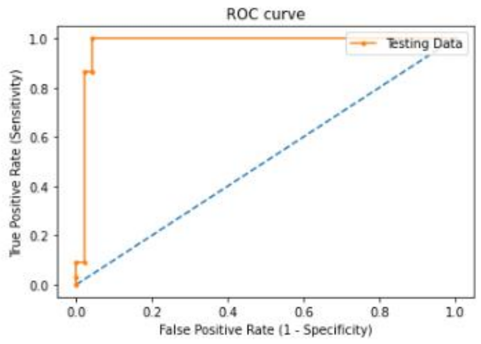
From the figure, we can calculate that TN=44, FP=3, FN=2 and TP=65.

K. SVM with linear kernel and c=1000

The classification report, heatmap and ROC curve are shown in the following figure.

```
              precision    recall  f1-score   support

           0       0.96      0.96      0.96        47
           1       0.97      0.97      0.97        67

    accuracy                           0.96       114
   macro avg       0.96      0.96      0.96       114
weighted avg       0.96      0.96      0.96       114
```



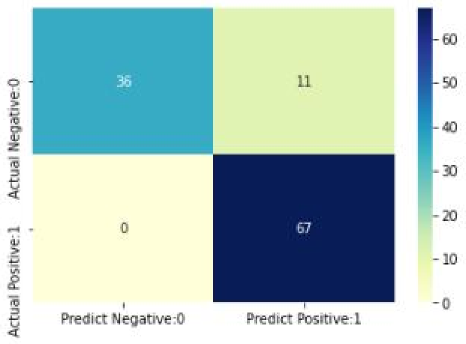


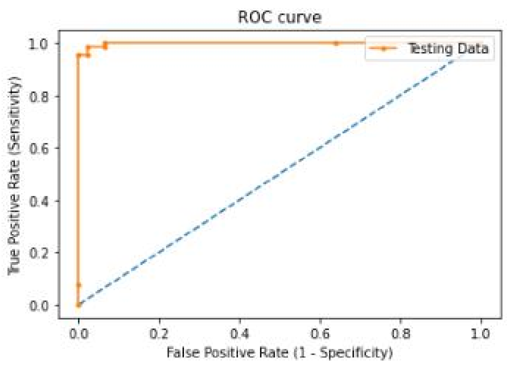
From the figure, we can calculate that TN=45, FP=2, FN=2 and TP=65.

L. SVM with polynomial kernel and c=1

The classification report, heatmap and ROC curve are shown in the following figure.

```
              precision    recall  f1-score   support

           0       1.00      0.77      0.87        47
           1       0.86      1.00      0.92        67

    accuracy                           0.90       114
   macro avg       0.93      0.88      0.90       114
weighted avg       0.92      0.90      0.90       114
```



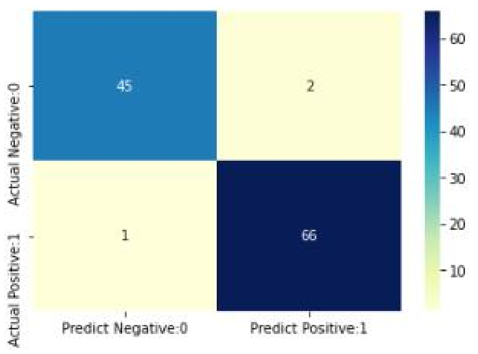


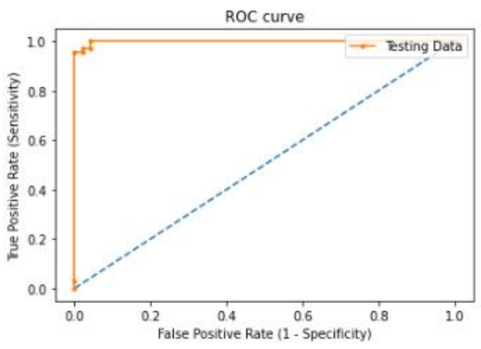
From the figure, we can calculate that TN=36, FP=11, FN=0 and TP=67.

M. SVM with polynomial kernel and c=100

The classification report, heatmap and ROC curve are shown in the following figure.

```
              precision    recall  f1-score   support

           0       0.98      0.96      0.97        47
           1       0.97      0.99      0.98        67

    accuracy                           0.97       114
   macro avg       0.97      0.97      0.97       114
weighted avg       0.97      0.97      0.97       114
```



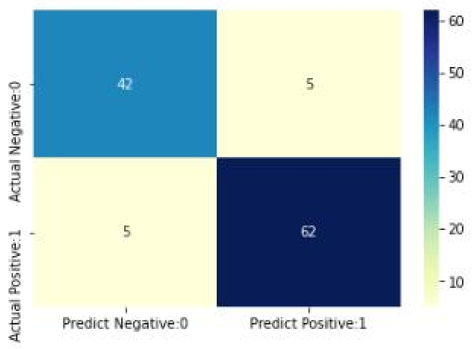


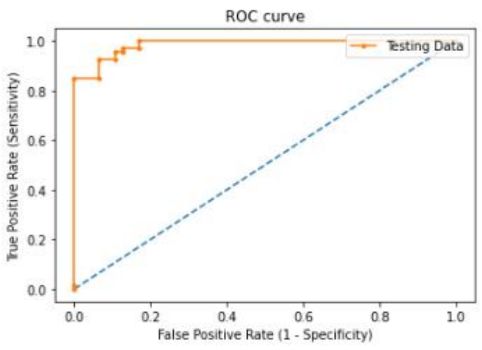
From the figure, we can calculate that TN=45, FP=2, FN=1 and TP=66.

N. SVM with sigmoid kernel and c=1

The classification report, heatmap and ROC curve are shown in the following figure.

```
              precision    recall  f1-score   support

           0       0.89      0.89      0.89        47
           1       0.93      0.93      0.93        67

    accuracy                           0.91       114
   macro avg       0.91      0.91      0.91       114
weighted avg       0.91      0.91      0.91       114
```



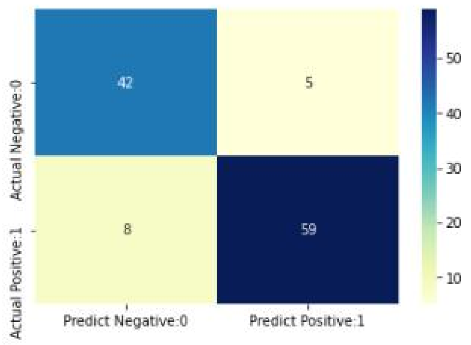


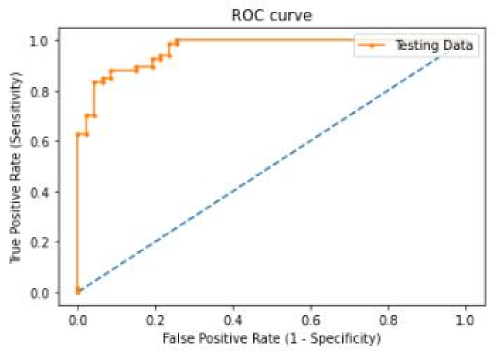
From the figure, we can calculate that TN=42, FP=5, FN=5 and TP=62.

O. SVM with sigmoid kernel and c=100

The classification report, heatmap and ROC curve are shown in the following figure.

```
              precision    recall  f1-score   support

           0       0.84      0.89      0.87        47
           1       0.92      0.88      0.90        67

    accuracy                           0.89       114
   macro avg       0.88      0.89      0.88       114
weighted avg       0.89      0.89      0.89       114
```



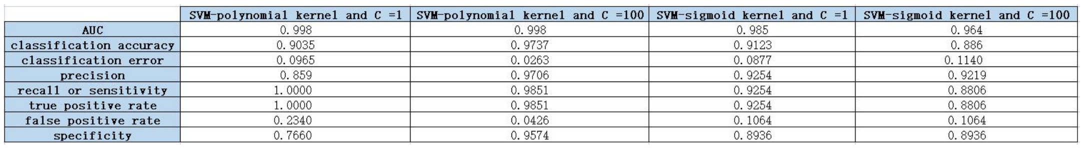


From the figure, we can calculate that TN=42, FP=5, FN=8 and TP=59.

The summary tables for the statistic data are analyzed in the following part.
In the experiment, we apply SVM with different kernel and different C. C is related to the acceptable abnormal point in the margin. If c is more higher, the less abnormal points in the margin is acceptable. So when c approaches to infinite, the margin also call hard margin which means that no abnormal point exists in the margin. In practice, correctly set the value of c can relieve the problem of overfitting. We use four kernels including rbf kernel, linear kernel, polynomial kernel and sigmoid kernel in the experiment. The result is shown in the following figure.
From the figure, when it comes to the rbf kernel, we observe that the result of SVM with rgf kernel,c=100 and SVM with rbf kernel,c=1000 is the same whose classification accuracy reaches to 0.9474.
When it comes to the linear kernel, we observe that the classification accuracy of SVM with linear kernel, c=1 reaches to 0.9825, the classification accuracy of SVM with linear kernel, c=100 reaches to 0.9561 and the classification accuracy of SVM with linear kernel, c=1000 reaches to 0.9649. In terms of SVM with linear kernel, c=1 achieves the best performance among three different c, the c=1000 achieves the second performance and c=100 achieves the relative poor performance. The result indicates that for cancer datasets, the soft margin may be appropriate to the linear kernel. High C will cause overfitting.

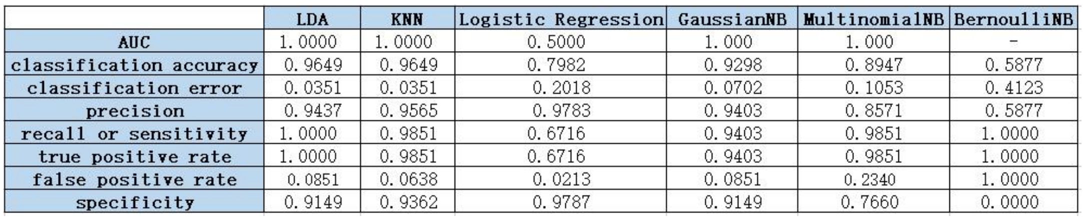
	SVM-rbf kernel-C =100.0	SVM-rbf kernel-C =1000.0	SVM-linear kernel-C =1	SVM-linear kernel-C =100	SVM-linear kernel-C =1000
AUC	0.993	0.993	0.999	0.979	0.978
classification accuracy	0.9474	0.9474	0.9825	0.9561	0.9649
classification error	0.0526	0.0526	0.0175	0.0439	0.0351
precision	0.9841	0.9841	0.9851	0.9559	0.9701
recall or sensitivity	0.9254	0.9254	0.9851	0.9701	0.9701
true positive rate	0.9254	0.9254	0.9851	0.9701	0.9701
false positive rate	0.0213	0.0213	0.0213	0.0638	0.0426
specificity	0.9787	0.9787	0.9787	0.9362	0.9574

When it comes to polynomial kernel, we observe that the classification accuracy of SVM with polynomial kernel, c=1 reaches to 0.9035 and the classification accuracy of SVM with polynomial kernel, c=100 reaches to 0.9737. In term of polynomial kernel, c=100 achieves the best performance among different c which indicates that the hard margin may be appropriate to polynomial kernel.
When it come to sigmoid kernel, we observe that the classification accuracy of SVM with sigmoid kernel, c=1 reaches to 0.9123 and the classification accuracy of SVM with sigmoid kernel, c=100 reaches to 0.886. In term of sigmoid kernel, c=1 achieves the best performance among different c which indicates that the soft margin may be appropriate to sigmoid kernel.
Compare the result of three different kernels, the rbf kernel may be appropriate to the breast cancer datasets classification when using SVM model.



	SVM-polynomial kernel and C =1	SVM-polynomial kernel and C =100	SVM-sigmoid kernel and C =1	SVM-sigmoid kernel and C =100
AUC	0.998	0.998	0.985	0.964
classification accuracy	0.9035	0.9737	0.9123	0.886
classification error	0.0965	0.0263	0.0877	0.1140
precision	0.859	0.9706	0.9254	0.9219
recall or sensitivity	1.0000	0.9851	0.9254	0.8806
true positive rate	1.0000	0.9851	0.9254	0.8806
false positive rate	0.2340	0.0426	0.1064	0.1064
specificity	0.7660	0.9574	0.8936	0.8936

When it comes to NB model, the classification accuracy of GaussianNB reaches to 0.9298, the classification accuracy of MultinomialNB reaches to 0.8947 and the classification accuracy of BernoulliNB reaches to 0.5877. In terms of NB model, the GaussianNB achieves the best performance, the MultinomialNB achieves the second best performance and the BernoulliNB achieves the poor performance.
When it comes to LDA, the classification accuracy reaches to 0.9649. When it comes to KNN, the classification accuracy reaches to 0.9649. When it comes to Logistic Regression, the classification accuracy reaches to 0.7982.



	LDA	KNN	Logistic Regression	GaussianNB	MultinomialNB	BernoulliNB
AUC	1.0000	1.0000	0.5000	1.000	1.000	-
classification accuracy	0.9649	0.9649	0.7982	0.9298	0.8947	0.5877
classification error	0.0351	0.0351	0.2018	0.0702	0.1053	0.4123
precision	0.9437	0.9565	0.9783	0.9403	0.8571	0.5877
recall or sensitivity	1.0000	0.9851	0.6716	0.9403	0.9851	1.0000
true positive rate	1.0000	0.9851	0.6716	0.9403	0.9851	1.0000
false positive rate	0.0851	0.0638	0.0213	0.0851	0.2340	1.0000
specificity	0.9149	0.9362	0.9787	0.9149	0.7660	0.0000

Compare all the classifiers, in terms of classification accuracy, the SVM with linear kernel and c=1 achieves the best performance and its classification accuracy reaches to 0.9825; the BernoulliNB achieve the poorest performance and it classification accuracy reaches to 0.5877.
In addition, the SVM with linear kernel and c=100, the SVM with linear kernel and c=1000, the SVM with polynomial kernel and c=100, the LDA and KNN perform well in classifying breast cancer datasets and the classification accuracy of them all up to 0.96.

指导教师批阅意见： 成绩评定： 指导教师签字： 年　　月　　日
备注：

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。

comparison between different discriminators

May 23, 2022

```
[61]: import pandas as pd
      from sklearn.model_selection import train_test_split
      from sklearn.preprocessing import StandardScaler
      import numpy as np
      from sklearn.metrics import accuracy_score
      from sklearn.model_selection import cross_val_score
      from sklearn.metrics import roc_auc_score, roc_curve, classification_report
      import matplotlib.pyplot as plt
      from sklearn.datasets import load_breast_cancer
      from sklearn.metrics import confusion_matrix
      import seaborn as sns
```

```
[102]: cancer = load_breast_cancer()
       df_cancer = pd.DataFrame(data=cancer.data, columns=cancer.feature_names)
       df_cancer['class'] = cancer.target
```

```
[41]: df_breast.head()
```

```
[41]:    Clump Thickness  Uniformity of Cell Size  Uniformity of Cell Shape  \
      0                5                        1                         1
      1                5                        4                         4
      2                3                        1                         1
      3                6                        8                         8
      4                4                        1                         1

         Marginal Adhesion  Single Epithelial Cell Size Bare Nuclei  \
      0                  1                            2           1
      1                  5                            7          10
      2                  1                            2           2
      3                  1                            3           4
      4                  3                            2           1

         Bland Chromatin  Normal Nucleoli  Mitoses
      0                3                1        1
      1                3                2        1
      2                3                1        1
      3                3                7        1
      4                3                1        1
```

[122]:
```
def plot_confusion_matrix(y_test, y_pred_test):
    cm = confusion_matrix(y_test, y_pred_test)
    cm_matrix = pd.DataFrame(data=cm, columns=['Predict Negative:0', 'Predict␣
↪Positive:1'], index=['Actual Negative:0', 'Actual Positive:1'])
    sns.heatmap(cm_matrix, annot=True, fmt='d', cmap='YlGnBu')
    TP = cm[1,1]
    TN = cm[0,0]
    FP = cm[0,1]
    FN = cm[1,0]
    classification_accuracy = (TP + TN) / float(TP + TN + FP + FN)
    print('Classification accuracy : {0:0.4f}'.format(classification_accuracy))
    classification_error = (FP + FN) / float(TP + TN + FP + FN)
    print('Classification error : {0:0.4f}'.format(classification_error))
    precision = TP / float(TP + FP)
    print('Precision : {0:0.4f}'.format(precision))
    recall = TP / float(TP + FN)
    print('Recall or Sensitivity : {0:0.4f}'.format(recall))
    true_positive_rate = TP / float(TP + FN)
    print('True Positive Rate : {0:0.4f}'.format(true_positive_rate))
    false_positive_rate = FP / float(FP + TN)
    print('False Positive Rate : {0:0.4f}'.format(false_positive_rate))
    specificity = TN / (TN + FP)
    print('Specificity : {0:0.4f}'.format(specificity))
```

[28]:
```
def print_classification_report(y_test, y_pred_test):
    print(classification_report(y_test, y_pred_test))
```

[132]:
```
# Receiver Operating Characteristic Curve
def calculate_AUC(x_test, y_test, model):
    pred_prob_test = model.predict_proba(x_test)
    auc_test = roc_auc_score(y_test,pred_prob_test[:,1])
    print('AUC for the Testing Data: %.3f' % auc_test)
```

[141]:
```
def plot_ROC(x_test, y_test, model):
    pred_prob_test = model.predict_proba(x_test)
    fpr_test, tpr_test, thresholds_test = roc_curve(y_test, pred_prob_test[:,1])
    plt.plot([0, 1], [0, 1], linestyle='--')
    plt.plot(fpr_test, tpr_test, marker='.',label = 'Testing Data')
    plt.title('ROC curve')
    plt.xlabel('False Positive Rate (1 - Specificity)')
    plt.ylabel('True Positive Rate (Sensitivity)')
    plt.legend(loc=1)
```

SVC

[153]:
```
from sklearn.svm import SVC

sample_x = df_cancer.iloc[:, 0:-1]
```

```
sample_y = df_cancer.iloc[:, -1]
x_train, x_test, y_train, y_test = train_test_split(sample_x, sample_y,␣
 ↪random_state=0, test_size=0.2)

scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)

# Run SVM with rbf kernel and C=100.0
svc_rbf100 = SVC(C=100.0,probability=True)
svc_rbf100.fit(x_train, y_train)
y_pred_rbf100 = svc_rbf100.predict(x_test)
print('Model accuracy with rbf kernel and C =100.0 : {0:0.4f}'.
 ↪format(accuracy_score(y_test,y_pred_rbf100)))
calculate_AUC(x_test, y_test, svc_rbf100)
plt.figure(1)
plot_confusion_matrix(y_test, y_pred_rbf100)
plt.figure(2)
plot_ROC(x_test, y_test, svc_rbf100)
print_classification_report(y_test, y_pred_rbf100)

# Run SVM with rbf kernel and C=1000.0
svc_rbf1000 = SVC(C=1000.0,probability=True)
svc_rbf1000.fit(x_train, y_train)
y_pred_rbf1000 = svc_rbf1000.predict(x_test)
print('Model accuracy with rbf kernel and C =1000.0 : {0:0.4f}'.
 ↪format(accuracy_score(y_test,y_pred_rbf1000)))
calculate_AUC(x_test, y_test, svc_rbf1000)
plt.figure(3)
plot_confusion_matrix(y_test, y_pred_rbf1000)
plt.figure(4)
plot_ROC(x_test, y_test, svc_rbf1000)
print_classification_report(y_test, y_pred_rbf1000)

# Run SVM with linear kernel and C=1.0
svc_linear1 = SVC(kernel='linear', C=1.0,probability=True)
svc_linear1.fit(x_train, y_train)
y_pred_linear1 = svc_linear1.predict(x_test)
print('Model accuracy with linear kernel and C =1.0 : {0:0.4f}'.
 ↪format(accuracy_score(y_test,y_pred_linear1)))
calculate_AUC(x_test, y_test, svc_linear1)
plt.figure(5)
plot_confusion_matrix(y_test, y_pred_linear1)
plt.figure(6)
plot_ROC(x_test, y_test, svc_linear1)
print_classification_report(y_test, y_pred_linear1)
```

```
# Run SVM with linear kernel and C=100.0
svc_linear100 = SVC(kernel='linear', C=100.0,probability=True)
svc_linear100.fit(x_train, y_train)
y_pred_linear100 = svc_linear100.predict(x_test)
print('Model accuracy with linear kernel and C =100.0 : {0:0.4f}'.
 ↪format(accuracy_score(y_test,y_pred_linear100)))
calculate_AUC(x_test, y_test, svc_linear100)
plt.figure(7)
plot_confusion_matrix(y_test, y_pred_linear100)
plt.figure(8)
plot_ROC(x_test, y_test, svc_linear100)
print_classification_report(y_test, y_pred_linear100)

# Run SVM with linear kernel and C=1000.0
svc_linear1000 = SVC(kernel='linear', C=1000.0,probability=True)
svc_linear1000.fit(x_train, y_train)
y_pred_linear1000 = svc_linear1000.predict(x_test)
print('Model accuracy with linear kernel and C =1000.0 : {0:0.4f}'.
 ↪format(accuracy_score(y_test,y_pred_linear1000)))
calculate_AUC(x_test, y_test, svc_linear1000)
plt.figure(9)
plot_confusion_matrix(y_test, y_pred_linear1000)
plt.figure(10)
plot_ROC(x_test, y_test, svc_linear1000)
print_classification_report(y_test, y_pred_linear1000)

# Run SVM with polynomial kernel and C=1.0
svc_polynomial1 = SVC(kernel='poly', C=1.0,probability=True)
svc_polynomial1.fit(x_train, y_train)
y_pred_polynomial1 = svc_polynomial1.predict(x_test)
print('Model accuracy with polynomial kernel and C =1.0 : {0:0.4f}'.
 ↪format(accuracy_score(y_test,y_pred_polynomial1)))
calculate_AUC(x_test, y_test, svc_polynomial1)
plt.figure(11)
plot_confusion_matrix(y_test, y_pred_polynomial1)
plt.figure(12)
plot_ROC(x_test, y_test, svc_polynomial1)
print_classification_report(y_test, y_pred_polynomial1)

# Run SVM with polynomial kernel and C=100.0
svc_polynomial100 = SVC(kernel='poly', C=100.0,probability=True)
svc_polynomial100.fit(x_train, y_train)
y_pred_polynomial100 = svc_polynomial100.predict(x_test)
print('Model accuracy with polynomial kernel and C =100.0 : {0:0.4f}'.
 ↪format(accuracy_score(y_test,y_pred_polynomial100)))
calculate_AUC(x_test, y_test, svc_polynomial100)
plt.figure(13)
```

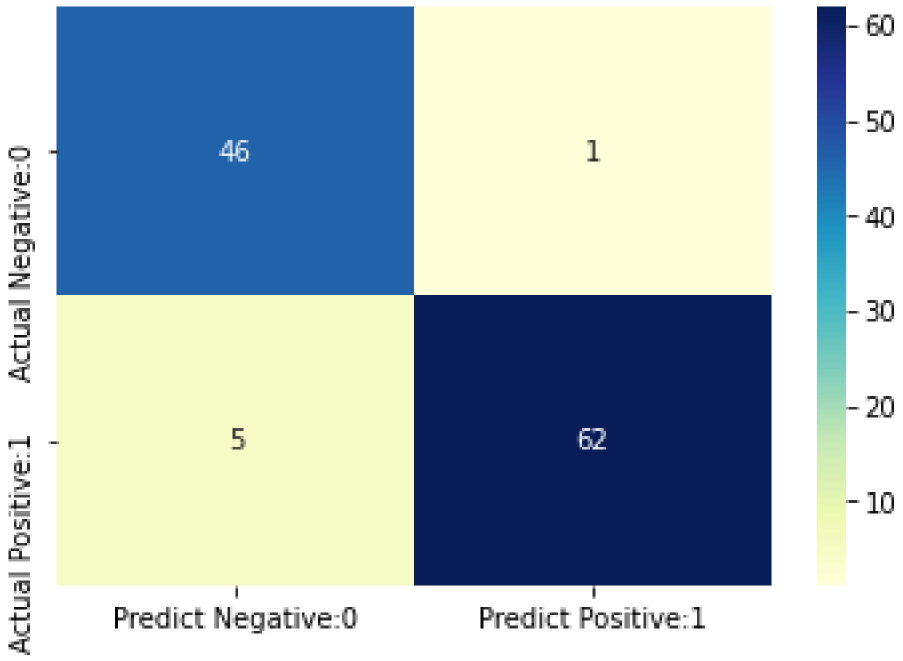
```
plot_confusion_matrix(y_test, y_pred_polynomial100)
plt.figure(14)
plot_ROC(x_test, y_test, svc_polynomial100)
print_classification_report(y_test, y_pred_polynomial100)

# Run SVM with sigmoid kernel and C=1.0
svc_sigmoid1 = SVC(kernel='sigmoid', C=1.0,probability=True)
svc_sigmoid1.fit(x_train, y_train)
y_pred_sigmoid1 = svc_sigmoid1.predict(x_test)
print('Model accuracy with sigmoid kernel and C =1.0 : {0:0.4f}'.
 ↪format(accuracy_score(y_test,y_pred_sigmoid1)))
calculate_AUC(x_test, y_test, svc_sigmoid1)
plt.figure(15)
plot_confusion_matrix(y_test, y_pred_sigmoid1)
plt.figure(16)
plot_ROC(x_test, y_test, svc_sigmoid1)
print_classification_report(y_test, y_pred_sigmoid1)

# Run SVM with sigmoid kernel and C=100.0
svc_sigmoid100 = SVC(kernel='sigmoid', C=100.0,probability=True)
svc_sigmoid100.fit(x_train, y_train)
y_pred_sigmoid100 = svc_sigmoid100.predict(x_test)
print('Model accuracy with sigmoid kernel and C =100.0 : {0:0.4f}'.
 ↪format(accuracy_score(y_test,y_pred_sigmoid100)))
calculate_AUC(x_test, y_test, svc_sigmoid100)
plt.figure(17)
plot_confusion_matrix(y_test, y_pred_sigmoid100)
plt.figure(18)
plot_ROC(x_test, y_test, svc_sigmoid100)
print_classification_report(y_test, y_pred_sigmoid100)
```

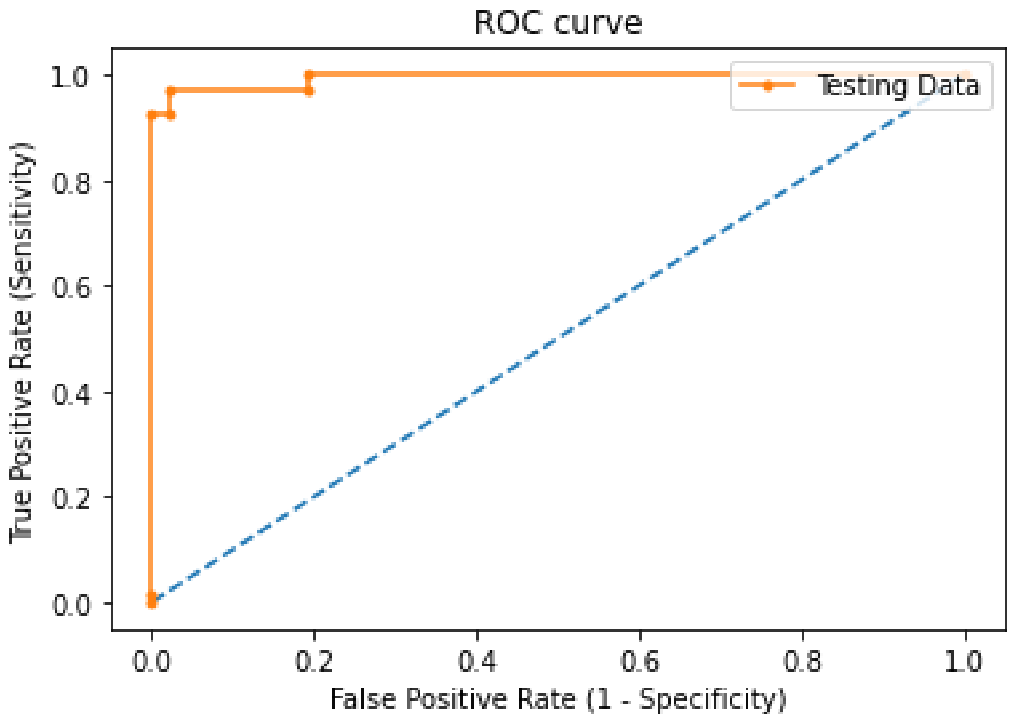
```
Model accuracy with rbf kernel and C =100.0 : 0.9474
AUC for the Testing Data: 0.993
Classification accuracy : 0.9474
Classification error : 0.0526
Precision : 0.9841
Recall or Sensitivity : 0.9254
True Positive Rate : 0.9254
False Positive Rate : 0.0213
Specificity : 0.9787
              precision    recall  f1-score   support

           0       0.90      0.98      0.94        47
           1       0.98      0.93      0.95        67

    accuracy                           0.95       114
   macro avg       0.94      0.95      0.95       114
```

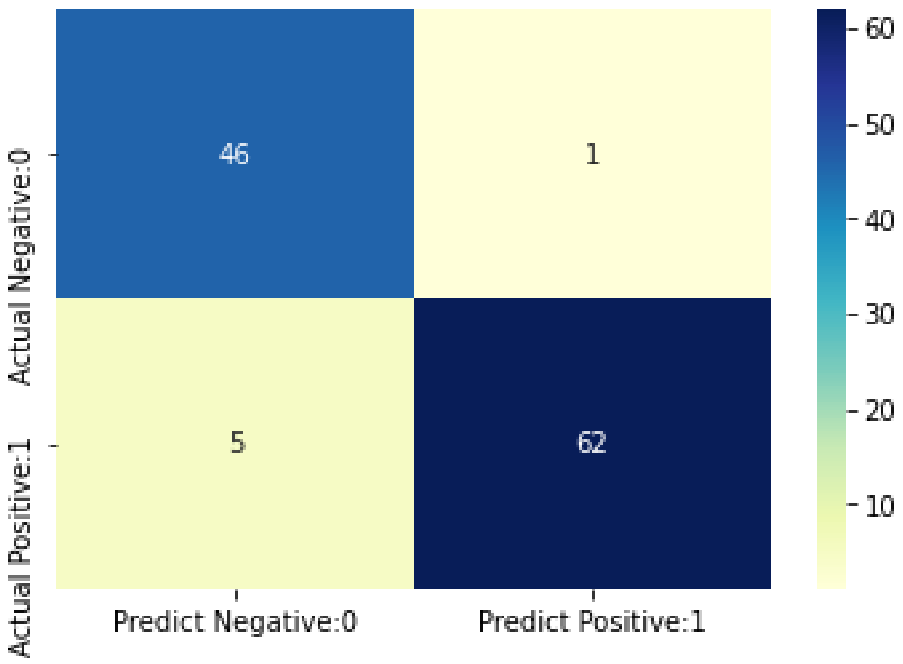
```
weighted avg       0.95      0.95      0.95       114

Model accuracy with rbf kernel and C =1000.0 : 0.9474
AUC for the Testing Data: 0.993
Classification accuracy : 0.9474
Classification error : 0.0526
Precision : 0.9841
Recall or Sensitivity : 0.9254
True Positive Rate : 0.9254
False Positive Rate : 0.0213
Specificity : 0.9787
              precision    recall  f1-score   support

           0       0.90      0.98      0.94        47
           1       0.98      0.93      0.95        67

    accuracy                           0.95       114
   macro avg       0.94      0.95      0.95       114
weighted avg       0.95      0.95      0.95       114

Model accuracy with linear kernel and C =1.0 : 0.9825
AUC for the Testing Data: 0.999
Classification accuracy : 0.9825
Classification error : 0.0175
Precision : 0.9851
Recall or Sensitivity : 0.9851
True Positive Rate : 0.9851
False Positive Rate : 0.0213
Specificity : 0.9787
              precision    recall  f1-score   support

           0       0.98      0.98      0.98        47
           1       0.99      0.99      0.99        67

    accuracy                           0.98       114
   macro avg       0.98      0.98      0.98       114
weighted avg       0.98      0.98      0.98       114

Model accuracy with linear kernel and C =100.0 : 0.9561
AUC for the Testing Data: 0.979
Classification accuracy : 0.9561
Classification error : 0.0439
Precision : 0.9559
Recall or Sensitivity : 0.9701
True Positive Rate : 0.9701
False Positive Rate : 0.0638
Specificity : 0.9362
              precision    recall  f1-score   support
           0       0.96      0.94      0.95        47
           1       0.96      0.97      0.96        67

    accuracy                           0.96       114
   macro avg       0.96      0.95      0.95       114
weighted avg       0.96      0.96      0.96       114

Model accuracy with linear kernel and C =1000.0 : 0.9649
AUC for the Testing Data: 0.978
Classification accuracy : 0.9649
Classification error : 0.0351
Precision : 0.9701
Recall or Sensitivity : 0.9701
True Positive Rate : 0.9701
False Positive Rate : 0.0426
Specificity : 0.9574
              precision    recall  f1-score   support

           0       0.96      0.96      0.96        47
           1       0.97      0.97      0.97        67

    accuracy                           0.96       114
   macro avg       0.96      0.96      0.96       114
weighted avg       0.96      0.96      0.96       114

Model accuracy with polynomial kernel and C =1.0 : 0.9035
AUC for the Testing Data: 0.998
Classification accuracy : 0.9035
Classification error : 0.0965
Precision : 0.8590
Recall or Sensitivity : 1.0000
True Positive Rate : 1.0000
False Positive Rate : 0.2340
Specificity : 0.7660
              precision    recall  f1-score   support

           0       1.00      0.77      0.87        47
           1       0.86      1.00      0.92        67

    accuracy                           0.90       114
   macro avg       0.93      0.88      0.90       114
weighted avg       0.92      0.90      0.90       114

Model accuracy with polynomial kernel and C =100.0 : 0.9737
AUC for the Testing Data: 0.998
Classification accuracy : 0.9737
Classification error : 0.0263
```

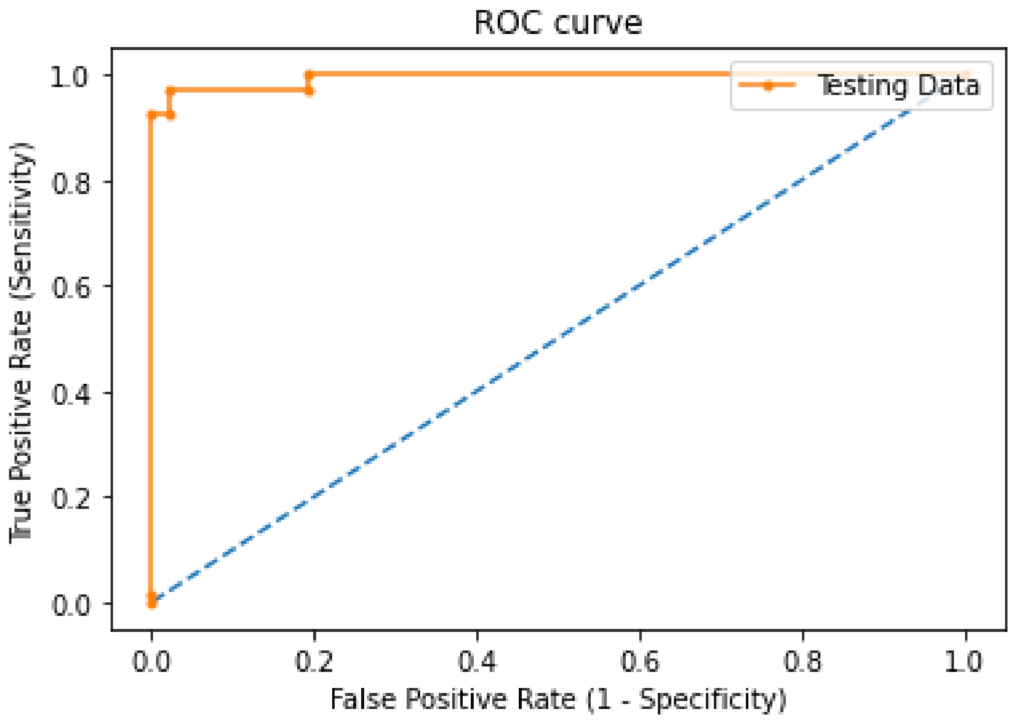
```
Precision : 0.9706
Recall or Sensitivity : 0.9851
True Positive Rate : 0.9851
False Positive Rate : 0.0426
Specificity : 0.9574
              precision    recall  f1-score   support

           0       0.98      0.96      0.97        47
           1       0.97      0.99      0.98        67

    accuracy                           0.97       114
   macro avg       0.97      0.97      0.97       114
weighted avg       0.97      0.97      0.97       114

Model accuracy with sigmoid kernel and C =1.0 : 0.9123
AUC for the Testing Data: 0.985
Classification accuracy : 0.9123
Classification error : 0.0877
Precision : 0.9254
Recall or Sensitivity : 0.9254
True Positive Rate : 0.9254
False Positive Rate : 0.1064
Specificity : 0.8936
              precision    recall  f1-score   support

           0       0.89      0.89      0.89        47
           1       0.93      0.93      0.93        67

    accuracy                           0.91       114
   macro avg       0.91      0.91      0.91       114
weighted avg       0.91      0.91      0.91       114

Model accuracy with sigmoid kernel and C =100.0 : 0.8860
AUC for the Testing Data: 0.964
Classification accuracy : 0.8860
Classification error : 0.1140
Precision : 0.9219
Recall or Sensitivity : 0.8806
True Positive Rate : 0.8806
False Positive Rate : 0.1064
Specificity : 0.8936
              precision    recall  f1-score   support

           0       0.84      0.89      0.87        47
           1       0.92      0.88      0.90        67

    accuracy                           0.89       114
   macro avg       0.88      0.89      0.88       114
```

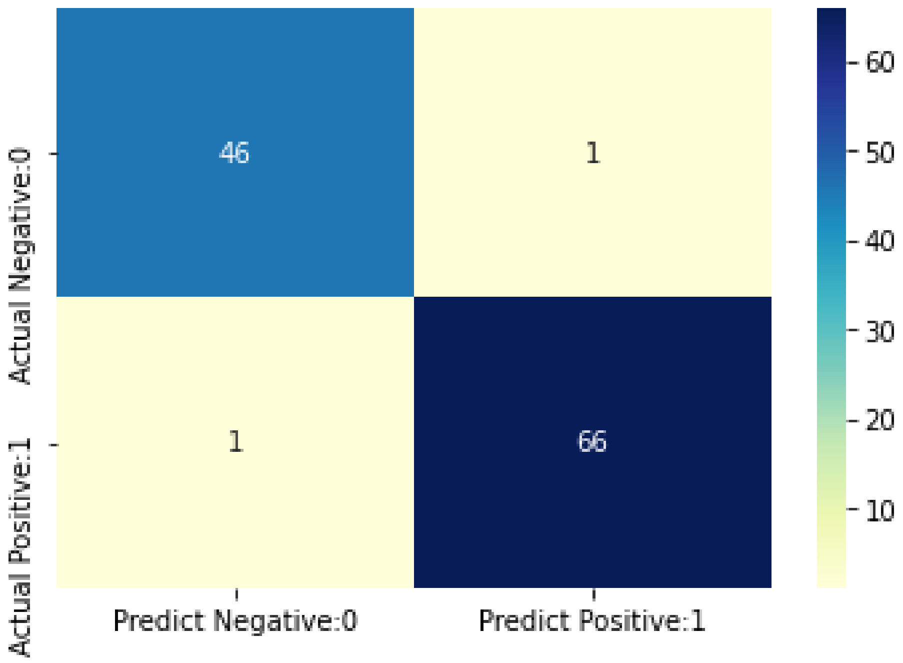
```
weighted avg       0.89      0.89      0.89       114
```



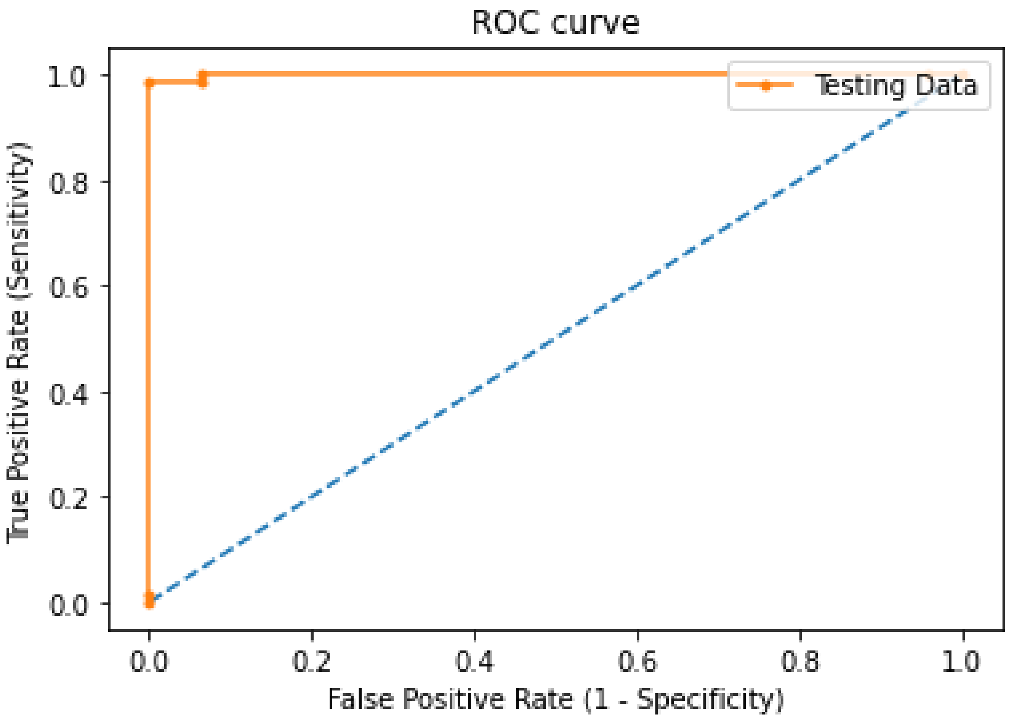



46
1
5
62
Actual Negative:0
Actual Positive:1
Predict Negative:0
Predict Positive:1
60
50
40
30
20
10

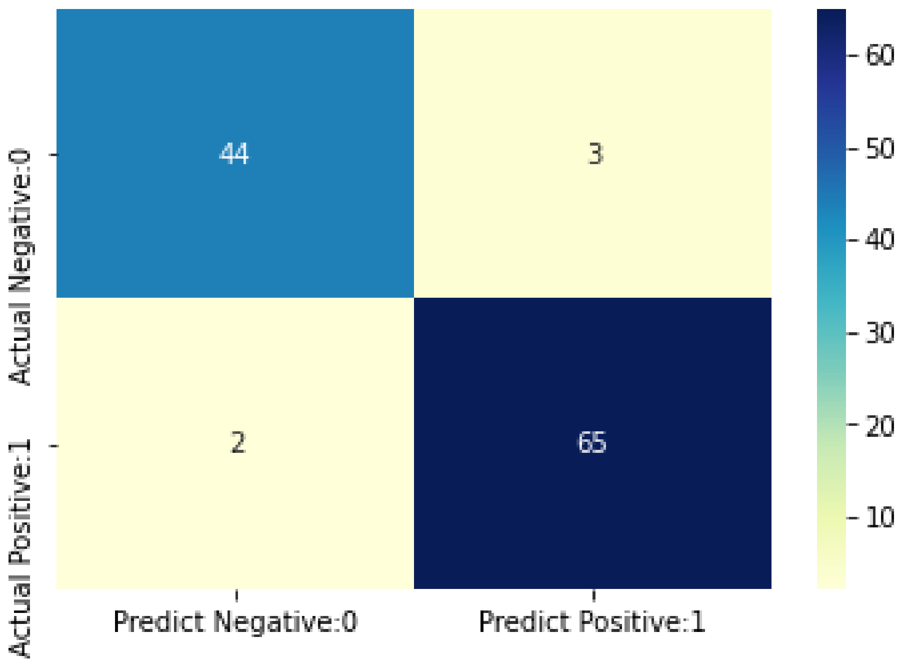

ROC curve
Testing Data
True Positive Rate (Sensitivity)
False Positive Rate (1 - Specificity)
0.0
0.2
0.4
0.6
0.8
1.0

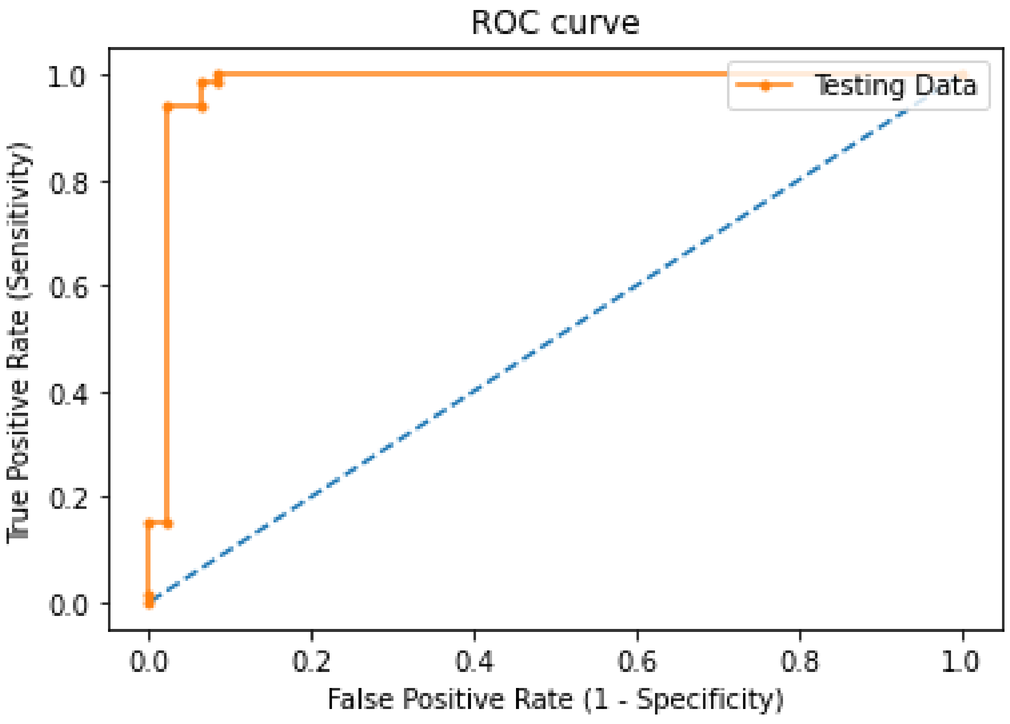


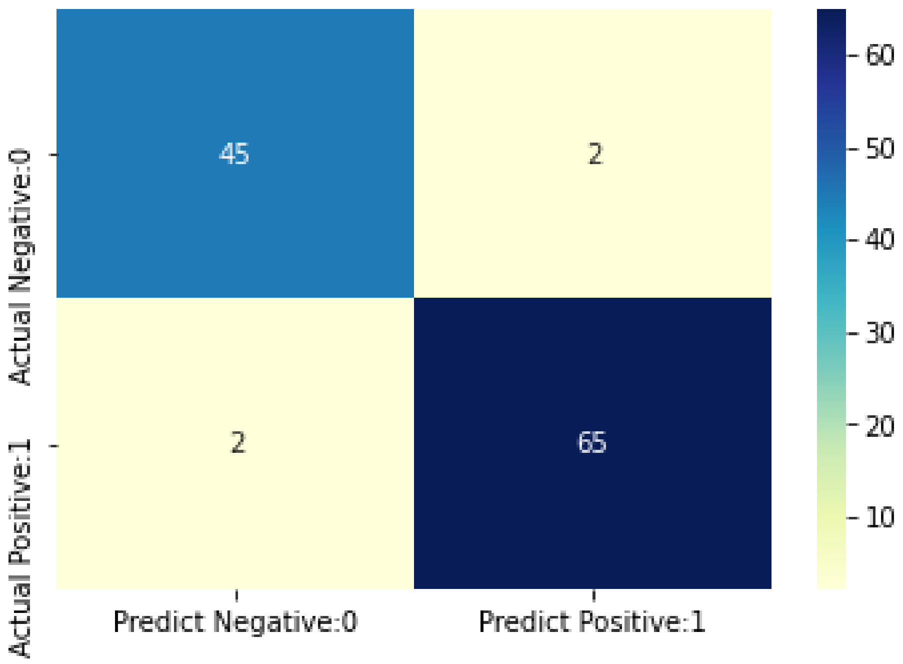
Actual Negative:0
Actual Positive:1
Predict Negative:0
Predict Positive:1
46
1
1
66
60
50
40
30
20
10

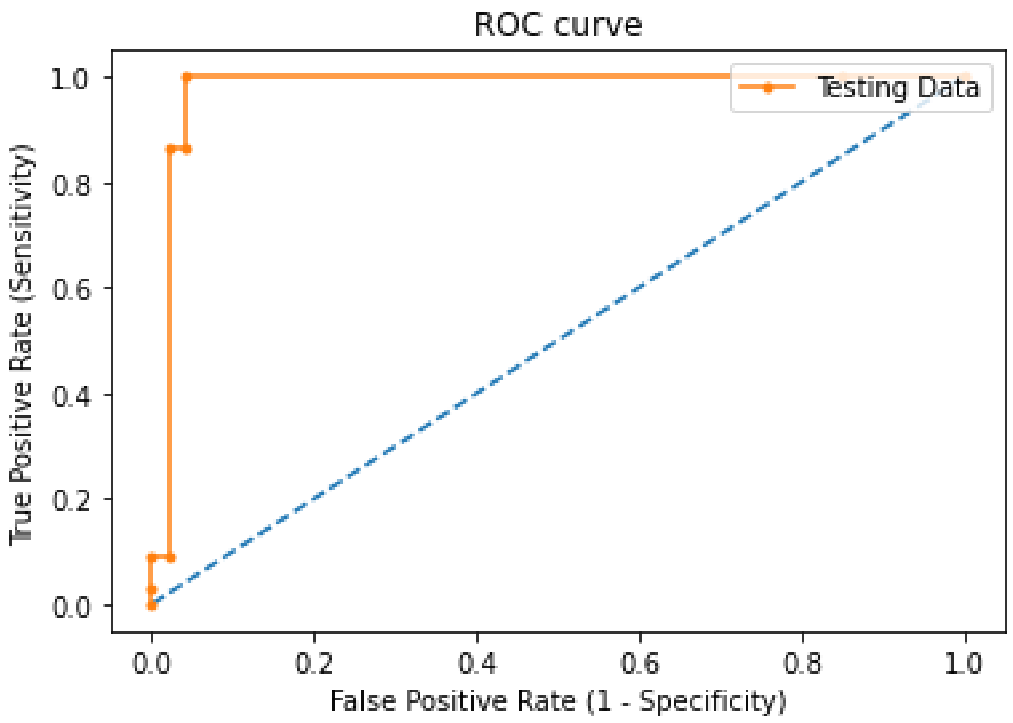


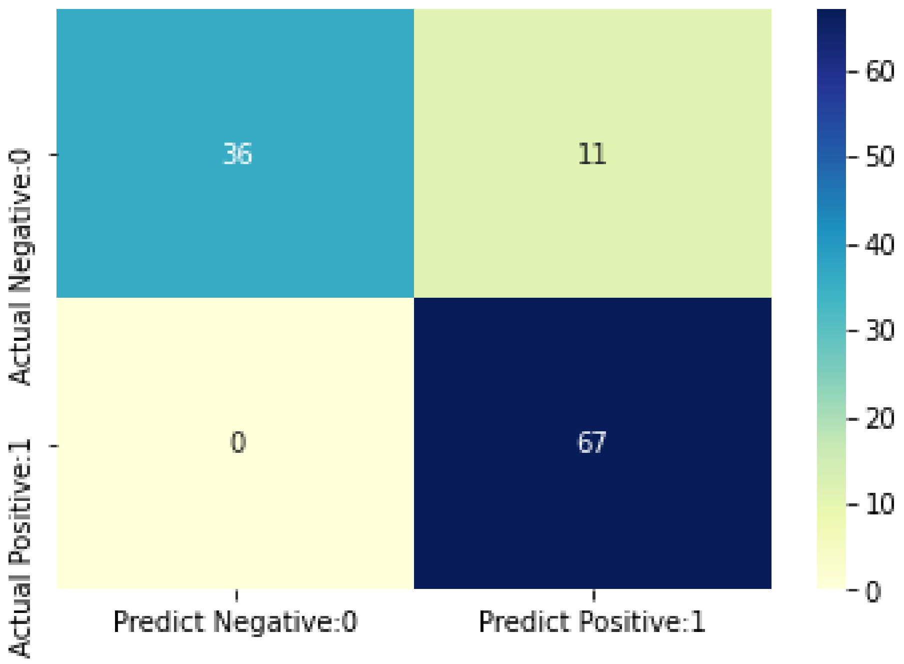
ROC curve
Testing Data
True Positive Rate (Sensitivity)
False Positive Rate (1 - Specificity)
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

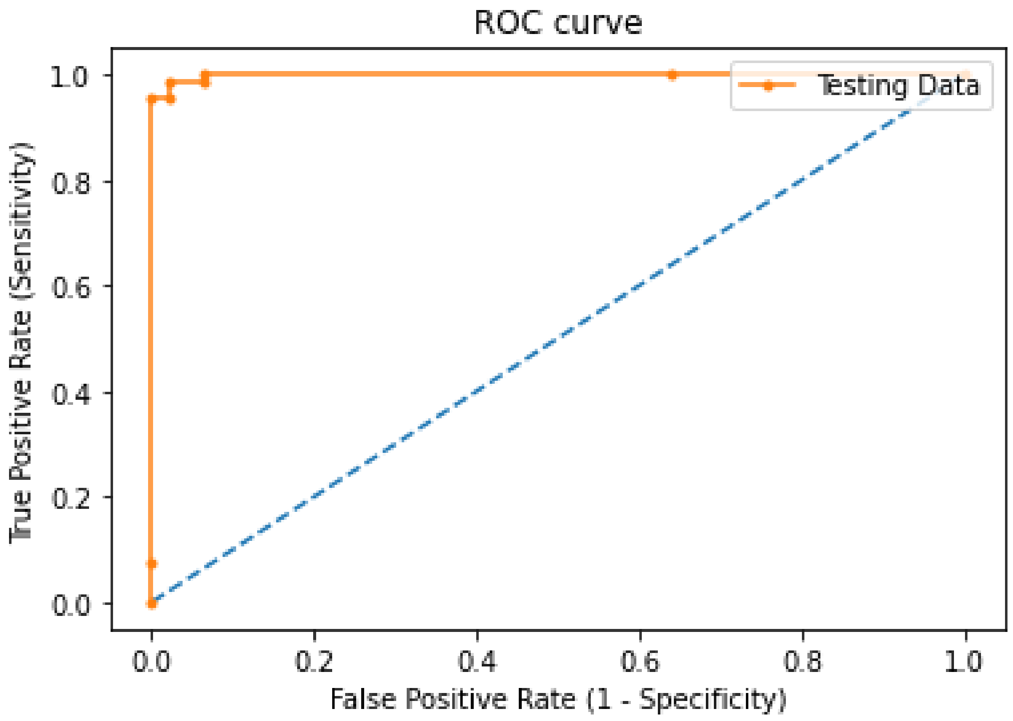

44
3
2
65
Actual Negative:0
Actual Positive:1
Predict Negative:0
Predict Positive:1
60
50
40
30
20
10

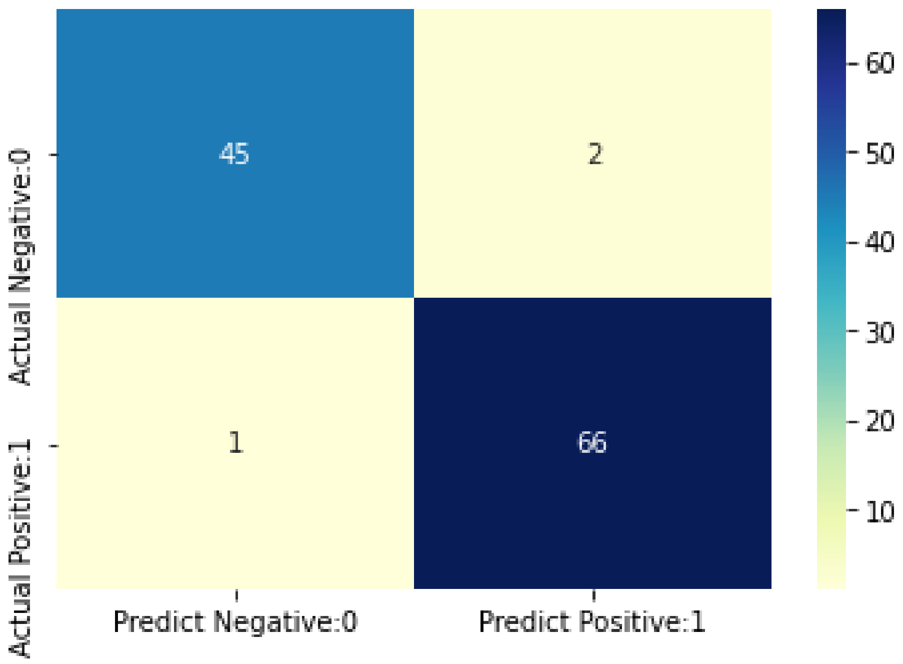

ROC curve
Testing Data
True Positive Rate (Sensitivity)
False Positive Rate (1 - Specificity)
0.0
0.2
0.4
0.6
0.8
1.0

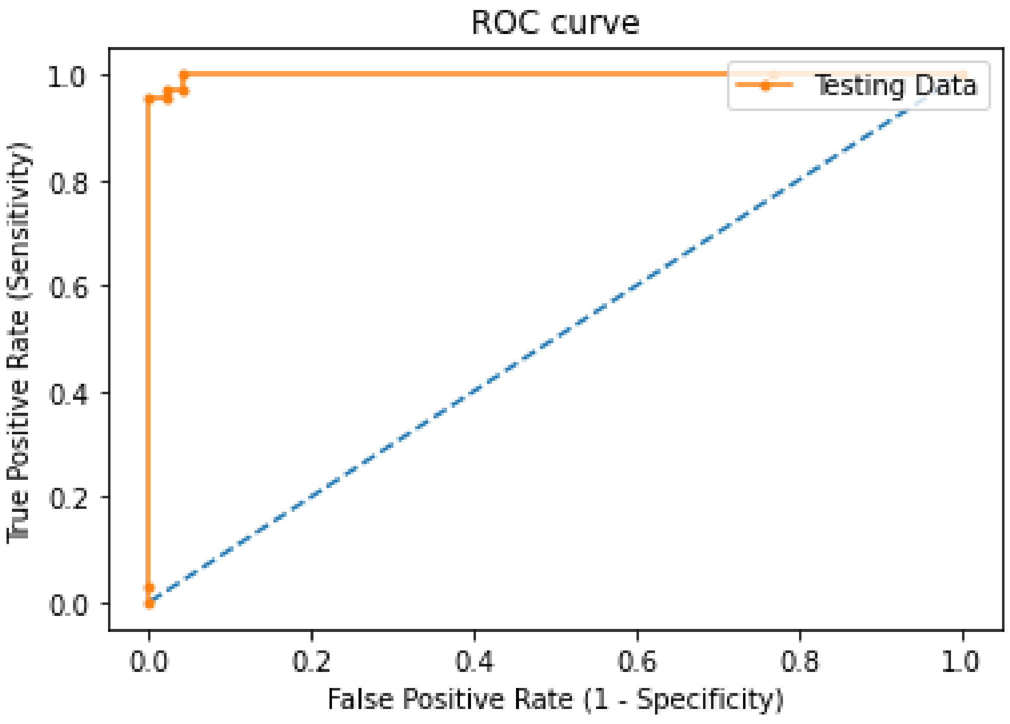

45
2
2
65
Actual Negative:0
Actual Positive:1
Predict Negative:0
Predict Positive:1
60
50
40
30
20
10

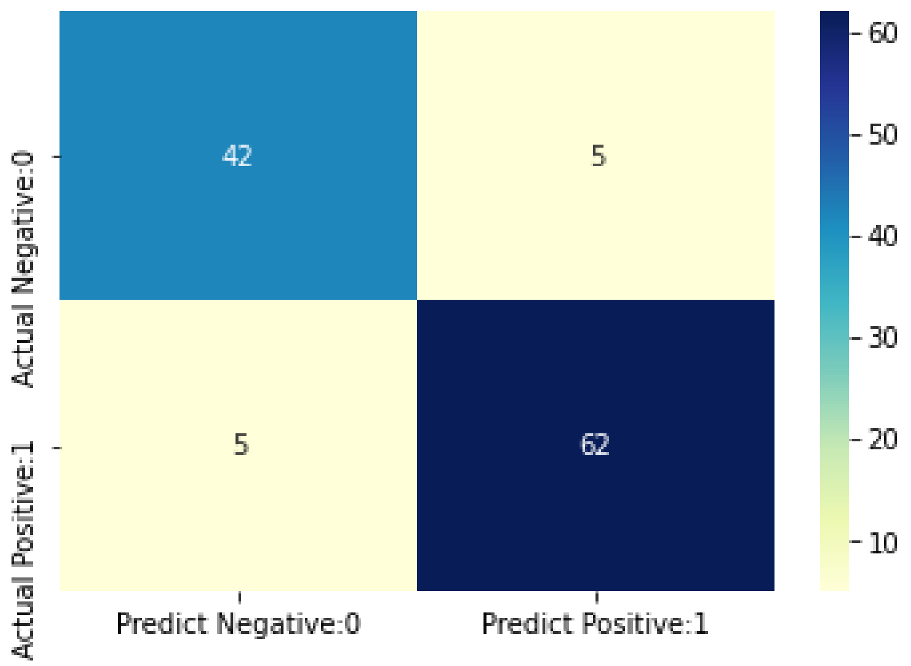

ROC curve
Testing Data
True Positive Rate (Sensitivity)
False Positive Rate (1 - Specificity)
0.0
0.2
0.4
0.6
0.8
1.0

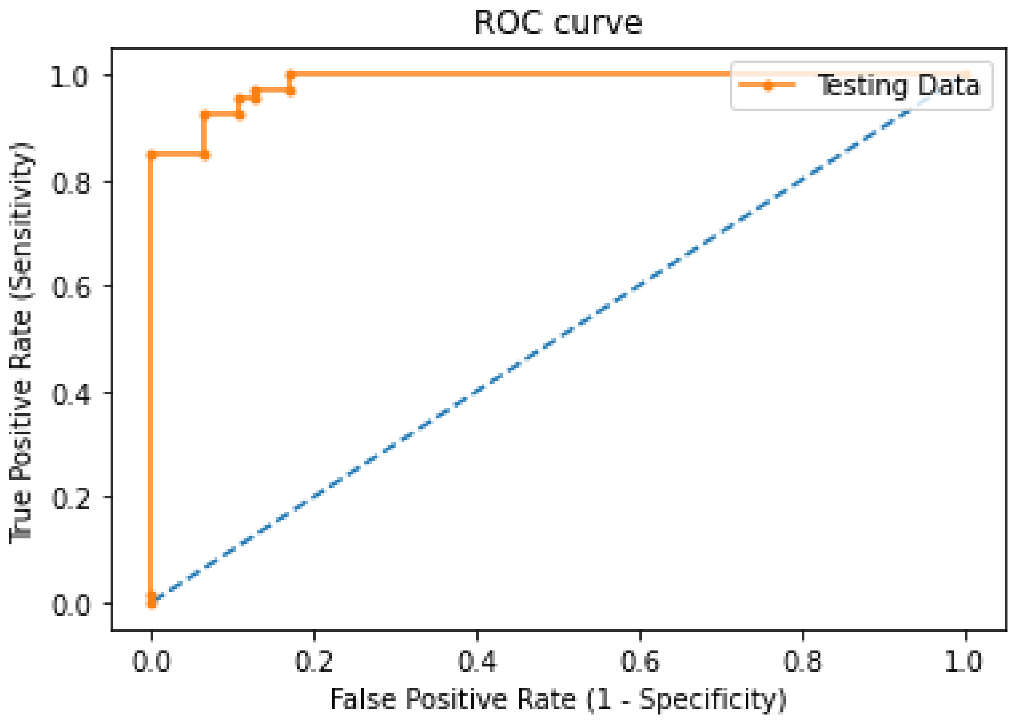

36
11
0
67
Actual Negative:0
Actual Positive:1
Predict Negative:0
Predict Positive:1
60
50
40
30
20
10
0

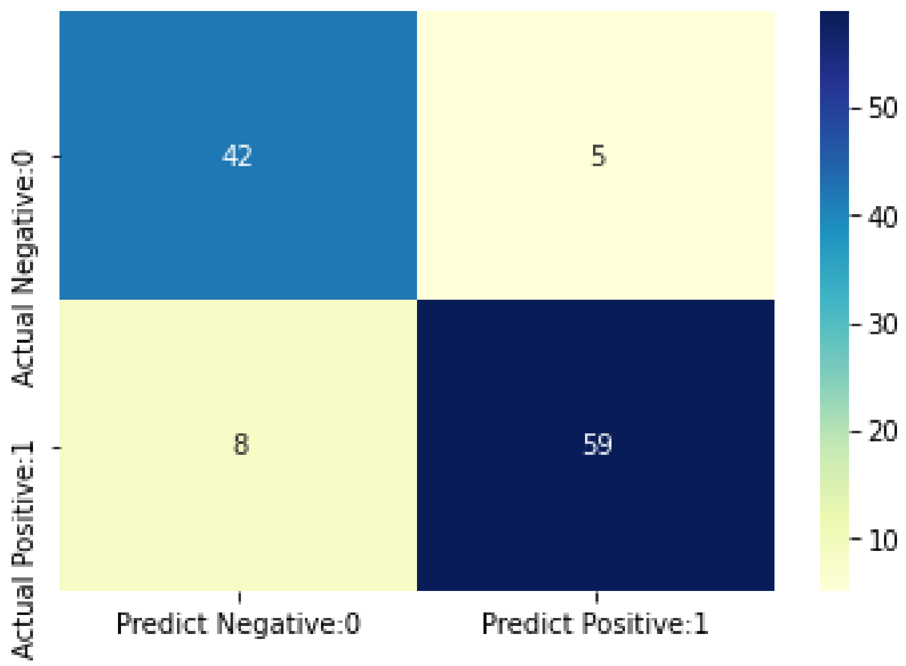

ROC curve
Testing Data
True Positive Rate (Sensitivity)
False Positive Rate (1 - Specificity)
0.0
0.2
0.4
0.6
0.8
1.0

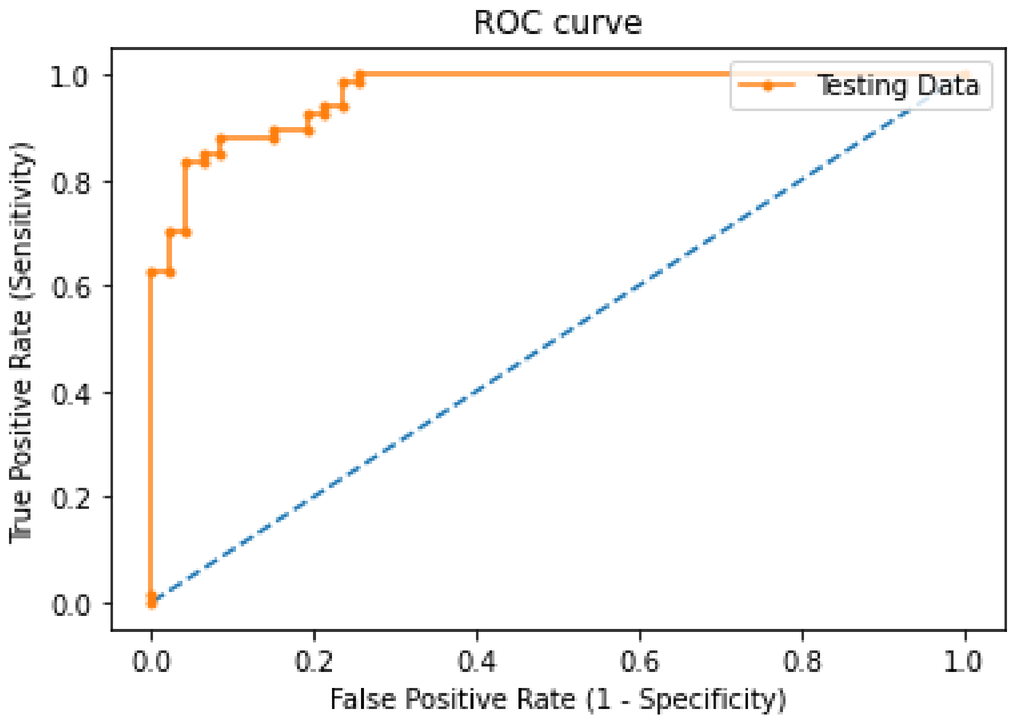

45
2
1
66
Actual Negative:0
Actual Positive:1
Predict Negative:0
Predict Positive:1
60
50
40
30
20
10


ROC curve
Testing Data
True Positive Rate (Sensitivity)
False Positive Rate (1 - Specificity)
0.0
0.2
0.4
0.6
0.8
1.0


42
5
5
62
Actual Negative:0
Actual Positive:1
Predict Negative:0
Predict Positive:1
60
50
40
30
20
10


ROC curve
Testing Data
True Positive Rate (Sensitivity)
False Positive Rate (1 - Specificity)
0.0
0.2
0.4
0.6
0.8
1.0

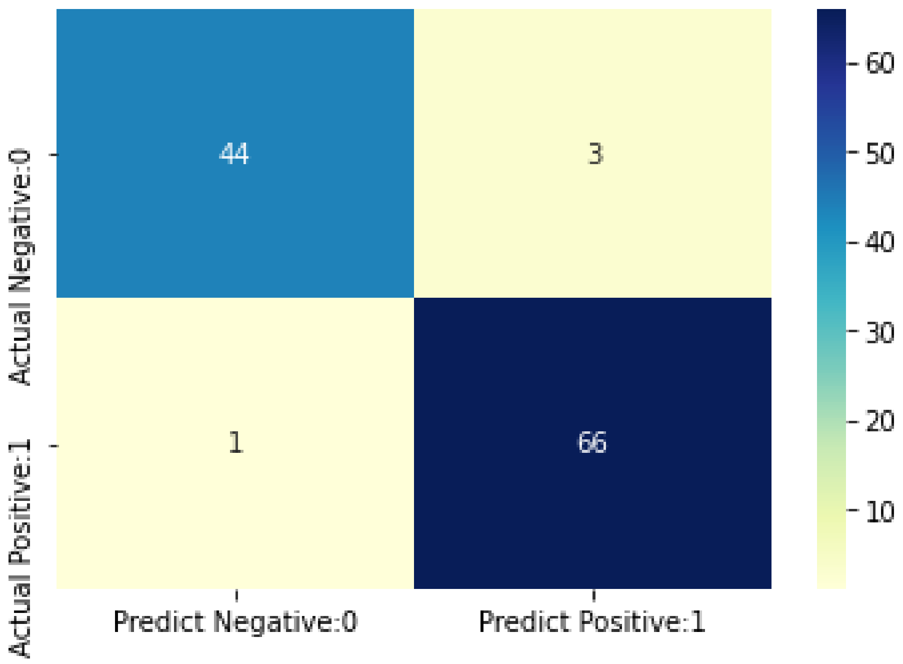

42
5
8
59
Actual Negative:0
Actual Positive:1
Predict Negative:0
Predict Positive:1
50
40
30
20
10

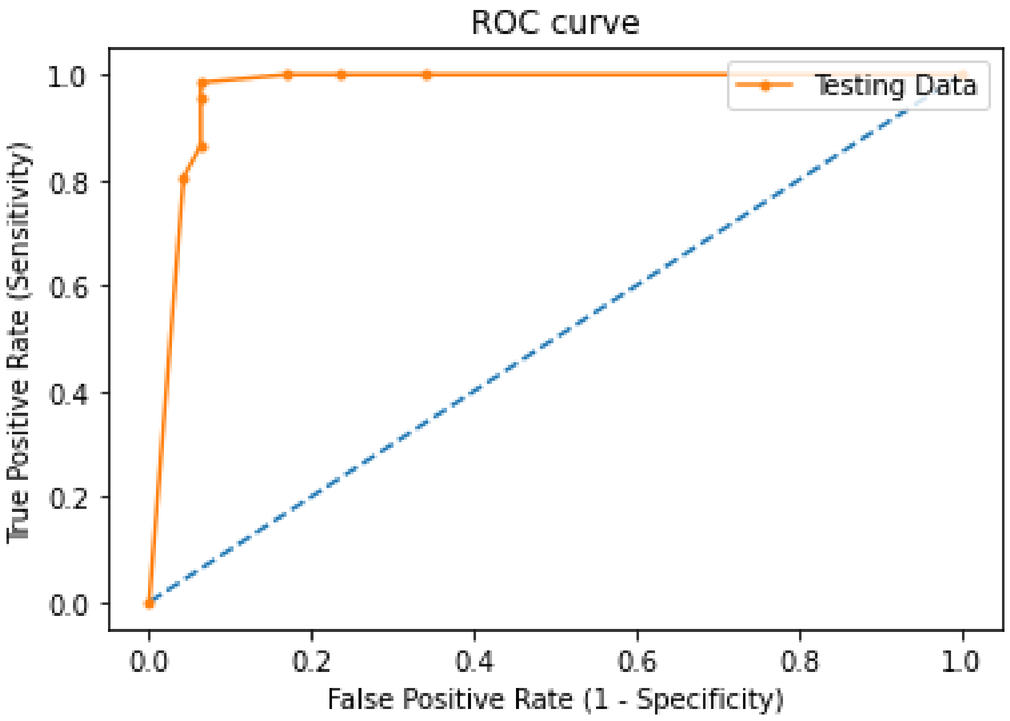

ROC curve
Testing Data
True Positive Rate (Sensitivity)
False Positive Rate (1 - Specificity)
0.0
0.2
0.4
0.6
0.8
1.0

KNN

[143]:
```
from sklearn.neighbors import KNeighborsClassifier as KNN

sample_x = df_cancer.iloc[:, 0:-1]
sample_y = df_cancer.iloc[:, -1]
x_train, x_test, y_train, y_test = train_test_split(sample_x, sample_y,␣
 ↪test_size=0.2, random_state=0)

ks = [1, 5, 9, 13, 15]
scores = []
for k in ks:
    knn = KNN(n_neighbors=k)
    score = cross_val_score(knn, x_train, y_train, cv=5)
    scores.append(1 - score.mean())
best_k = ks[np.argmin(scores)]
knn = KNN(n_neighbors=best_k)
knn.fit(x_train, y_train)
y_pred_knn = knn.predict(x_test)


calculate_AUC(x_test, y_pred_knn, knn)
plt.figure(1)
plot_confusion_matrix(y_test, y_pred_knn)
plt.figure(2)
plot_ROC(x_test, y_test, knn)
print_classification_report(y_test, y_pred_knn)
```

```
AUC for the Testing Data: 1.000
Classification accuracy : 0.9649
Classification error : 0.0351
Precision : 0.9565
Recall or Sensitivity : 0.9851
True Positive Rate : 0.9851
False Positive Rate : 0.0638
Specificity : 0.9362
              precision    recall  f1-score   support

           0       0.98      0.94      0.96        47
           1       0.96      0.99      0.97        67

    accuracy                           0.96       114
   macro avg       0.97      0.96      0.96       114
weighted avg       0.97      0.96      0.96       114
```

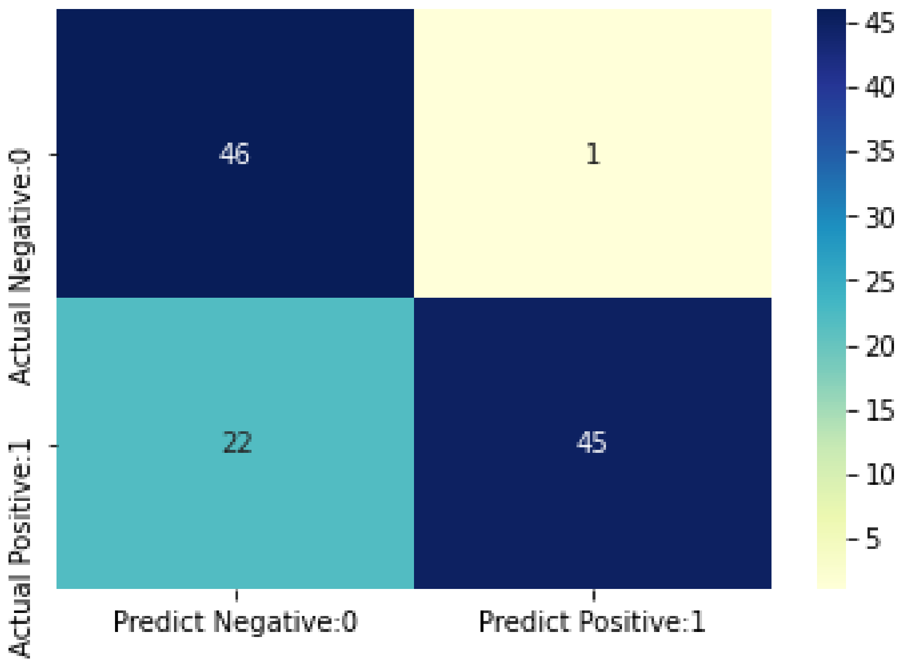


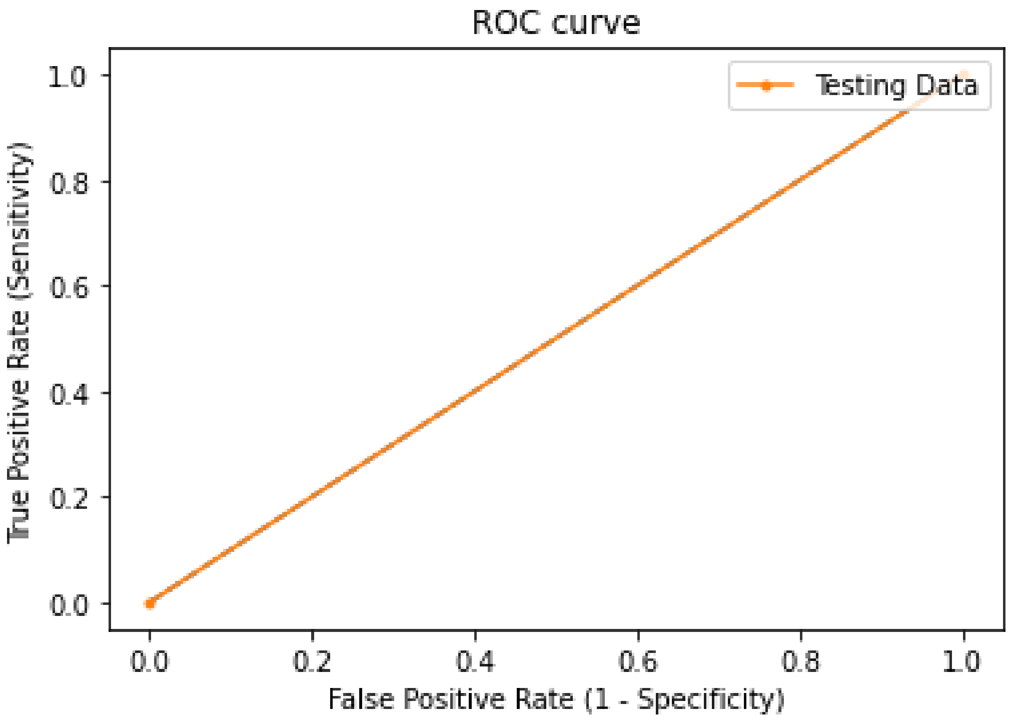


Logistic Regression

[172]:
```python
from sklearn import preprocessing

sample_x = df_cancer.iloc[:, 0:-1]
sample_y = df_cancer.iloc[:, -1]
x_train, x_test, y_train, y_test = train_test_split(sample_x, sample_y,␣
 ↪random_state=0, test_size=0.2)

minmax = preprocessing.MinMaxScaler()
minmax_x_train = minmax.fit_transform(x_train)
minmax_x_test = minmax.fit_transform(x_test)

from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression(solver='lbfgs', max_iter=1000,␣
 ↪multi_class='multinomial', C=1000)
log_reg.fit(minmax_x_train, y_train)
y_pred_log_reg = log_reg.predict(minmax_x_test)

calculate_AUC(x_test, y_pred_log_reg, log_reg)
plt.figure(1)
plot_confusion_matrix(y_test, y_pred_log_reg)
plt.figure(2)
plot_ROC(x_test, y_test, log_reg)
print_classification_report(y_test, y_pred_log_reg)
```

```
AUC for the Testing Data: 0.500
Classification accuracy : 0.7982
Classification error : 0.2018
Precision : 0.9783
Recall or Sensitivity : 0.6716
True Positive Rate : 0.6716
False Positive Rate : 0.0213
Specificity : 0.9787
              precision    recall  f1-score   support

           0       0.68      0.98      0.80        47
           1       0.98      0.67      0.80        67

    accuracy                           0.80       114
   macro avg       0.83      0.83      0.80       114
weighted avg       0.85      0.80      0.80       114
```

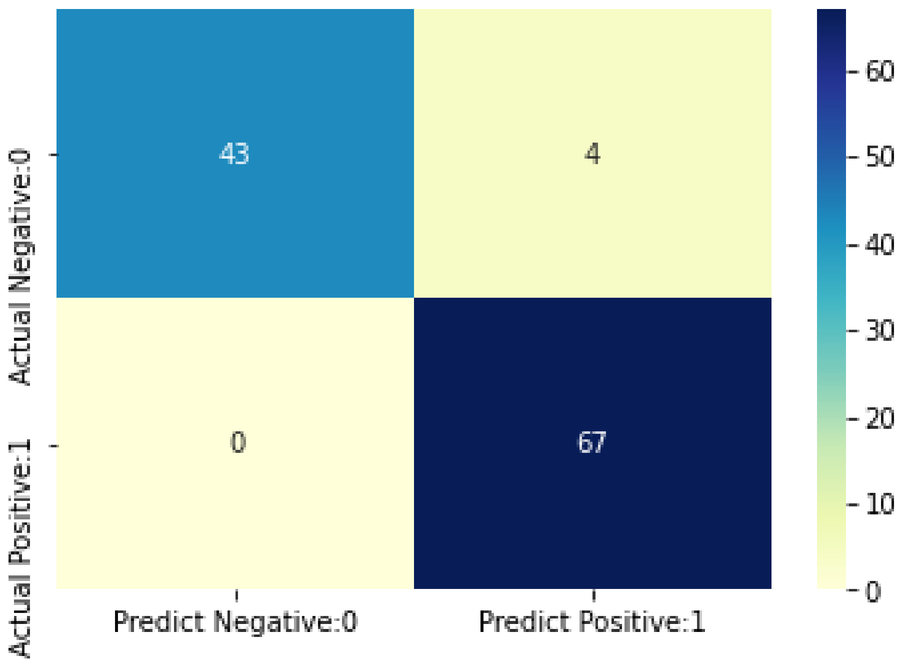




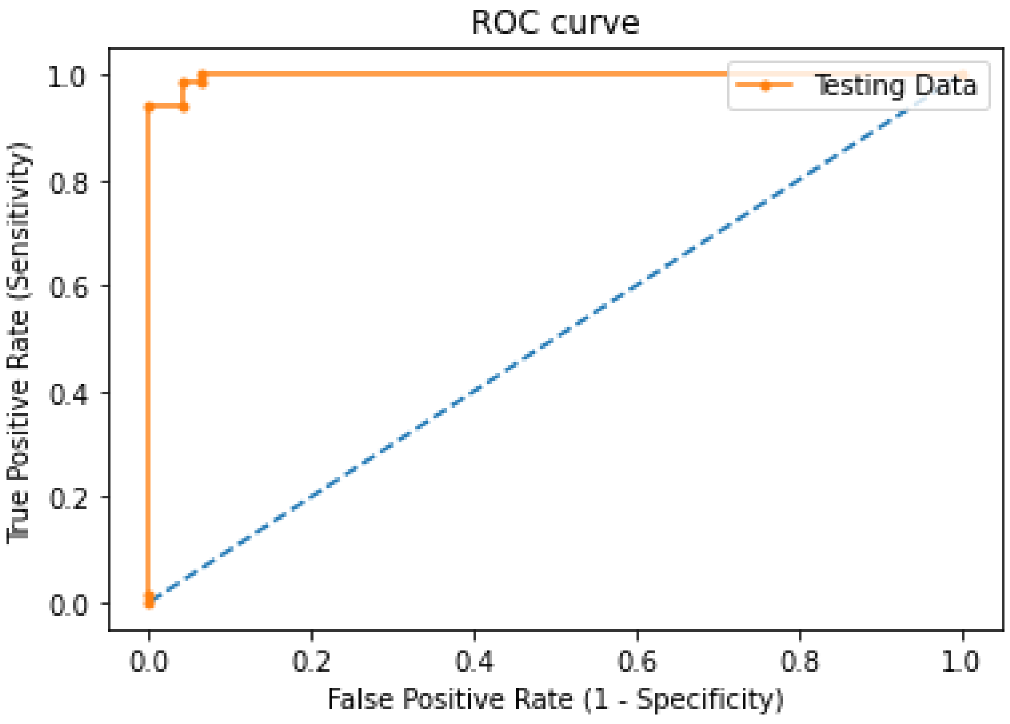
Linear Discriminant Analysis

[145]:
```python
#   LDA
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

sample_x = df_cancer.iloc[:, 0:-1]
sample_y = df_cancer.iloc[:, -1]
x_train, x_test, y_train, y_test = train_test_split(sample_x, sample_y,␣
 ↪random_state=0, test_size=0.2)

LDA = LinearDiscriminantAnalysis(n_components=1)
LDA = LDA.fit(x_train, y_train)
y_test_LDA = LDA.predict(x_test)

calculate_AUC(x_test, y_test_LDA, LDA)
plt.figure(1)
plot_confusion_matrix(y_test, y_test_LDA)
plt.figure(2)
plot_ROC(x_test, y_test, LDA)
print_classification_report(y_test, y_test_LDA)
```

```
AUC for the Testing Data: 1.000
Classification accuracy : 0.9649
Classification error : 0.0351
Precision : 0.9437
Recall or Sensitivity : 1.0000
True Positive Rate : 1.0000
False Positive Rate : 0.0851
Specificity : 0.9149
              precision    recall  f1-score   support

           0       1.00      0.91      0.96        47
           1       0.94      1.00      0.97        67

    accuracy                           0.96       114
   macro avg       0.97      0.96      0.96       114
weighted avg       0.97      0.96      0.96       114
```



Actual Negative:0
Actual Positive:1
Predict Negative:0
Predict Positive:1
43
4
0
67
0
10
20
30
40
50
60



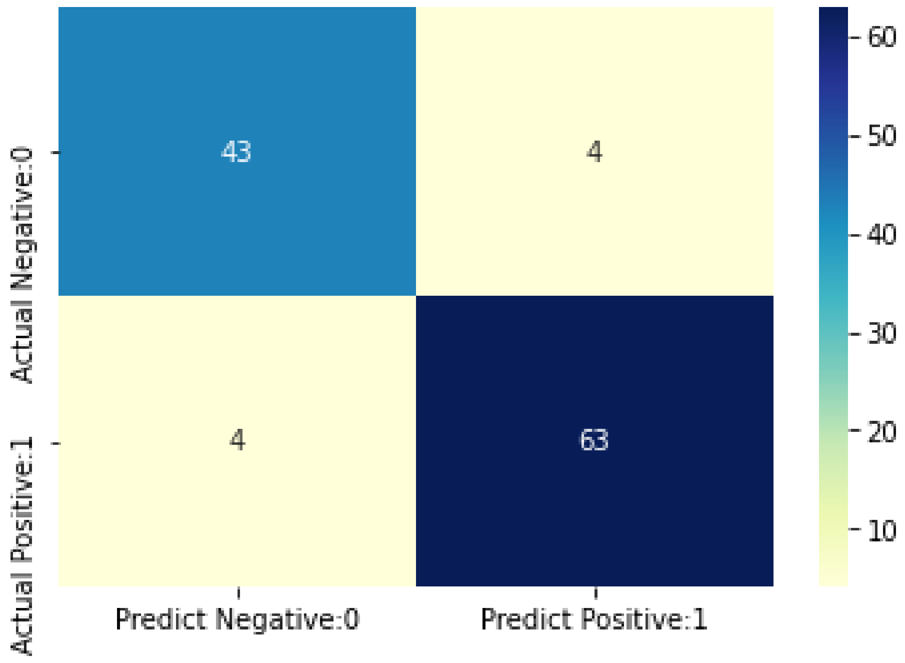
ROC curve
Testing Data
True Positive Rate (Sensitivity)
False Positive Rate (1 - Specificity)
0.0
0.2
0.4
0.6
0.8
1.0

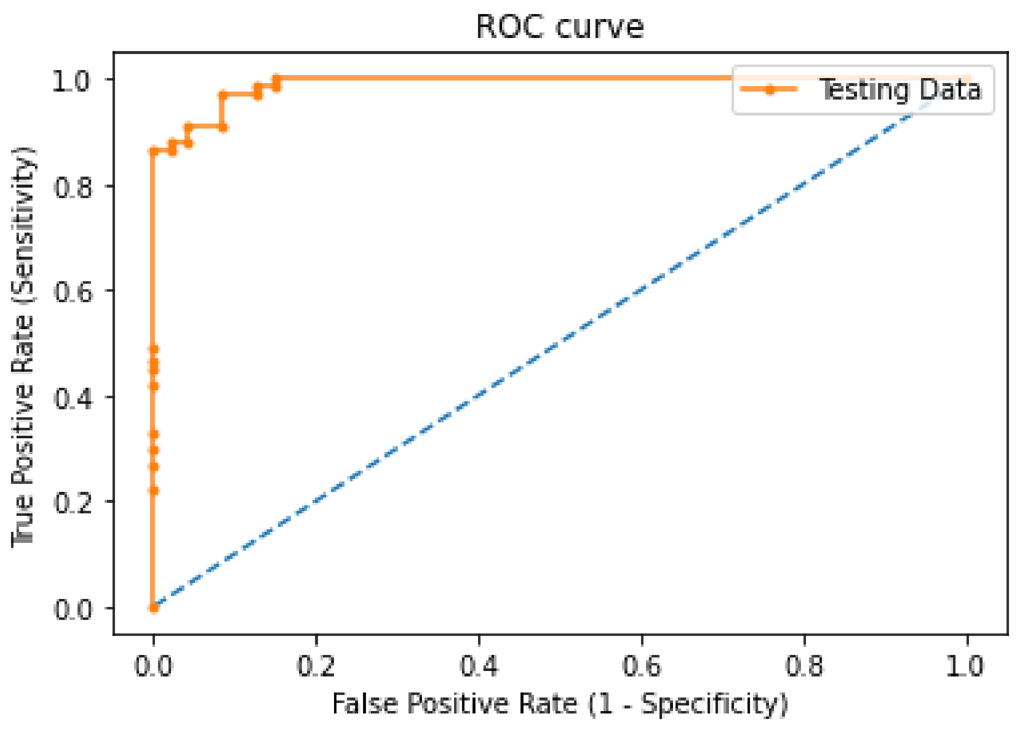
```
[174]: from sklearn.naive_bayes import GaussianNB
       from sklearn.naive_bayes import MultinomialNB
       from sklearn.naive_bayes import BernoulliNB

       sample_x = df_cancer.iloc[:, 0:-1]
       sample_y = df_cancer.iloc[:, -1]
       x_train, x_test, y_train, y_test = train_test_split(sample_x, sample_y,␣
        ↪random_state=0, test_size=0.2)

       gnb = GaussianNB()
       y_pred_gnb = gnb.fit(x_train, y_train).predict(x_test)
       calculate_AUC(x_test, y_pred_gnb, gnb)
       plt.figure(1)
       plot_confusion_matrix(y_test, y_pred_gnb)
       plt.figure(2)
       plot_ROC(x_test, y_test, gnb)
       print_classification_report(y_test, y_pred_gnb)

       clf = MultinomialNB()
       y_pred_clf = clf.fit(x_train, y_train).predict(x_test)
       calculate_AUC(x_test, y_pred_clf, clf)
       plt.figure(3)
       plot_confusion_matrix(y_test, y_pred_clf)
       plt.figure(4)
       plot_ROC(x_test, y_test, clf)
       print_classification_report(y_test, y_pred_clf)

       clf2 = BernoulliNB()
       y_pred_clf2 = clf2.fit(x_train, y_train).predict(x_test)
       print(y_test)
       print(y_pred_clf2)
       accuracy_score_clf2 = accuracy_score(y_test, y_pred_clf2)
       #calculate_AUC(x_test, y_pred_clf2, clf2)
       plt.figure(5)
       plot_confusion_matrix(y_test, y_pred_clf2)
       plt.figure(6)
       plot_ROC(x_test, y_test, clf2)
       print_classification_report(y_test, y_pred_clf2)
```

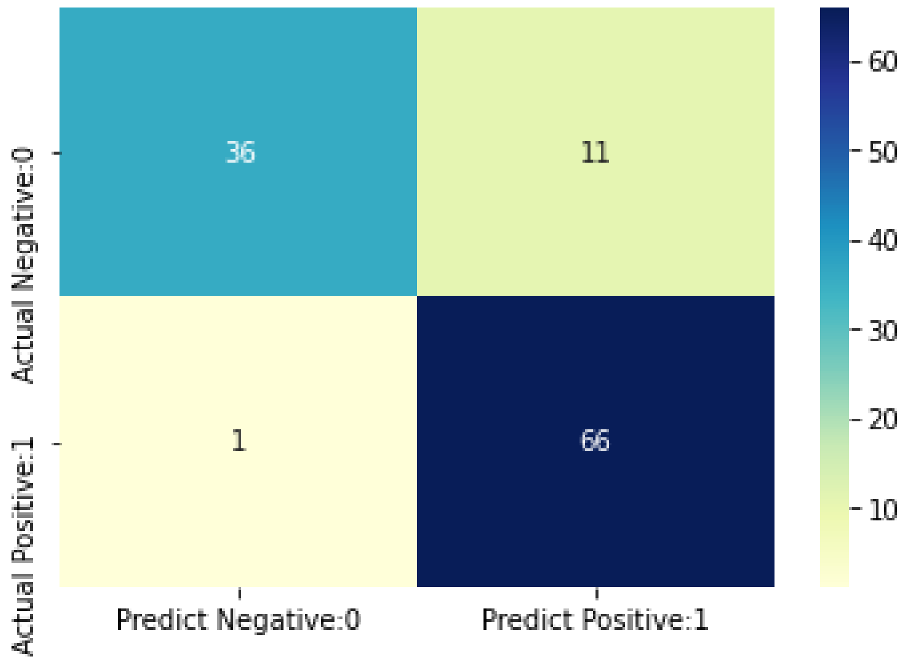
```
AUC for the Testing Data: 1.000
Classification accuracy : 0.9298
Classification error : 0.0702
Precision : 0.9403
Recall or Sensitivity : 0.9403
True Positive Rate : 0.9403
False Positive Rate : 0.0851
Specificity : 0.9149
              precision    recall  f1-score   support
```

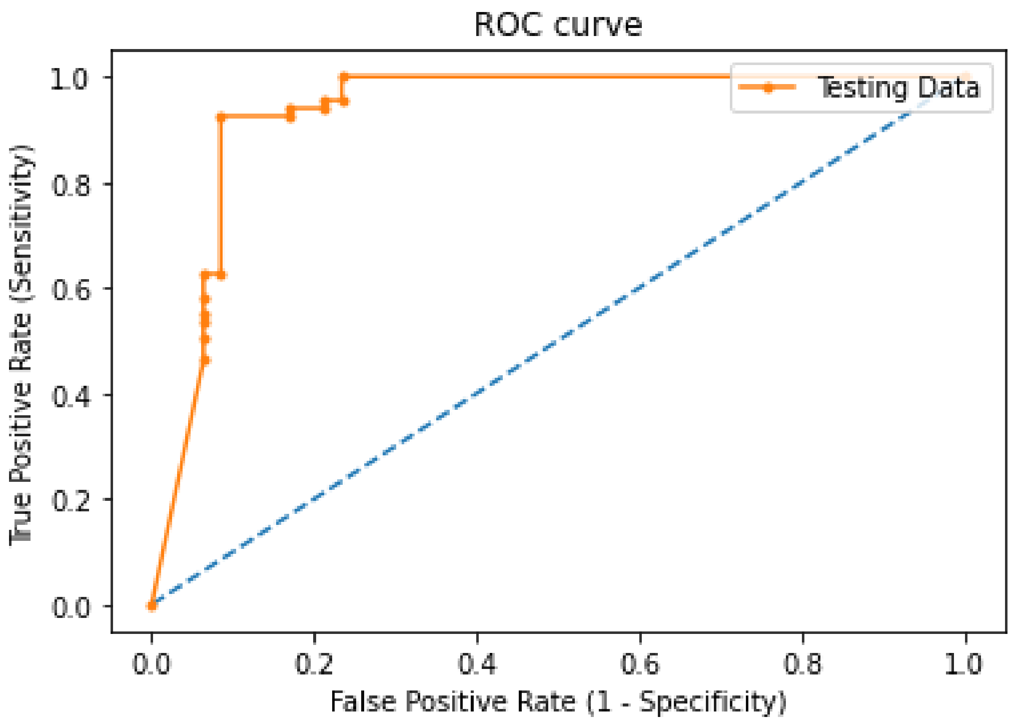
```
           0       0.91      0.91      0.91        47
           1       0.94      0.94      0.94        67

    accuracy                           0.93       114
   macro avg       0.93      0.93      0.93       114
weighted avg       0.93      0.93      0.93       114

AUC for the Testing Data: 1.000
Classification accuracy : 0.8947
Classification error : 0.1053
Precision : 0.8571
Recall or Sensitivity : 0.9851
True Positive Rate : 0.9851
False Positive Rate : 0.2340
Specificity : 0.7660
              precision    recall  f1-score   support

           0       0.97      0.77      0.86        47
           1       0.86      0.99      0.92        67

    accuracy                           0.89       114
   macro avg       0.92      0.88      0.89       114
weighted avg       0.90      0.89      0.89       114

512    0
457    1
439    1
298    1
37     1
      ..
213    0
519    1
432    0
516    0
500    1
Name: class, Length: 114, dtype: int32
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1]
Classification accuracy : 0.5877
Classification error : 0.4123
Precision : 0.5877
Recall or Sensitivity : 1.0000
True Positive Rate : 1.0000
False Positive Rate : 1.0000
Specificity : 0.0000
```

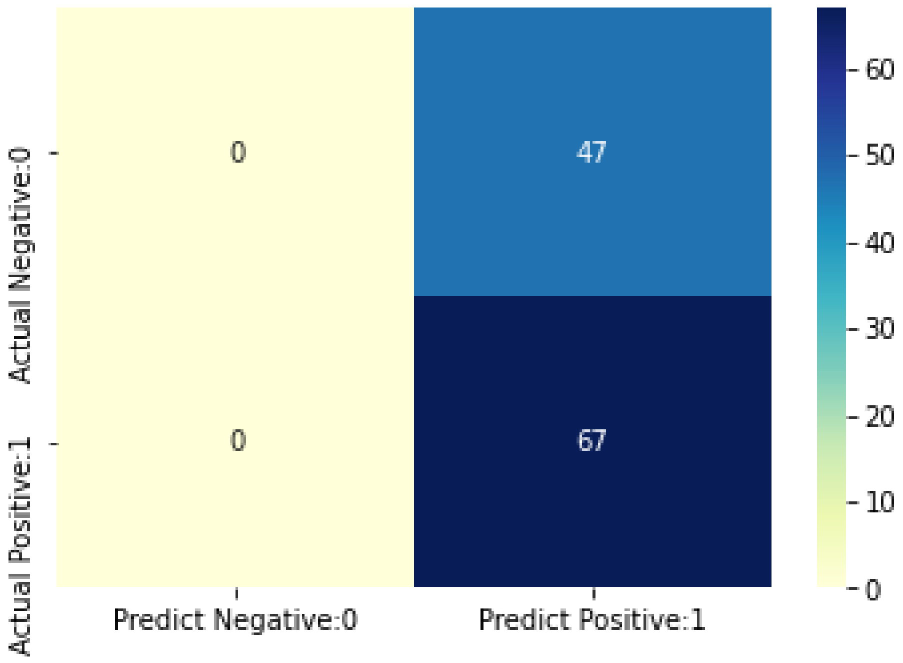
```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        47
           1       0.59      1.00      0.74        67

    accuracy                           0.59       114
   macro avg       0.29      0.50      0.37       114
weighted avg       0.35      0.59      0.44       114


C:\Users\28291\Anaconda3\lib\site-
packages\sklearn\metrics\_classification.py:1245: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\28291\Anaconda3\lib\site-
packages\sklearn\metrics\_classification.py:1245: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\28291\Anaconda3\lib\site-
packages\sklearn\metrics\_classification.py:1245: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

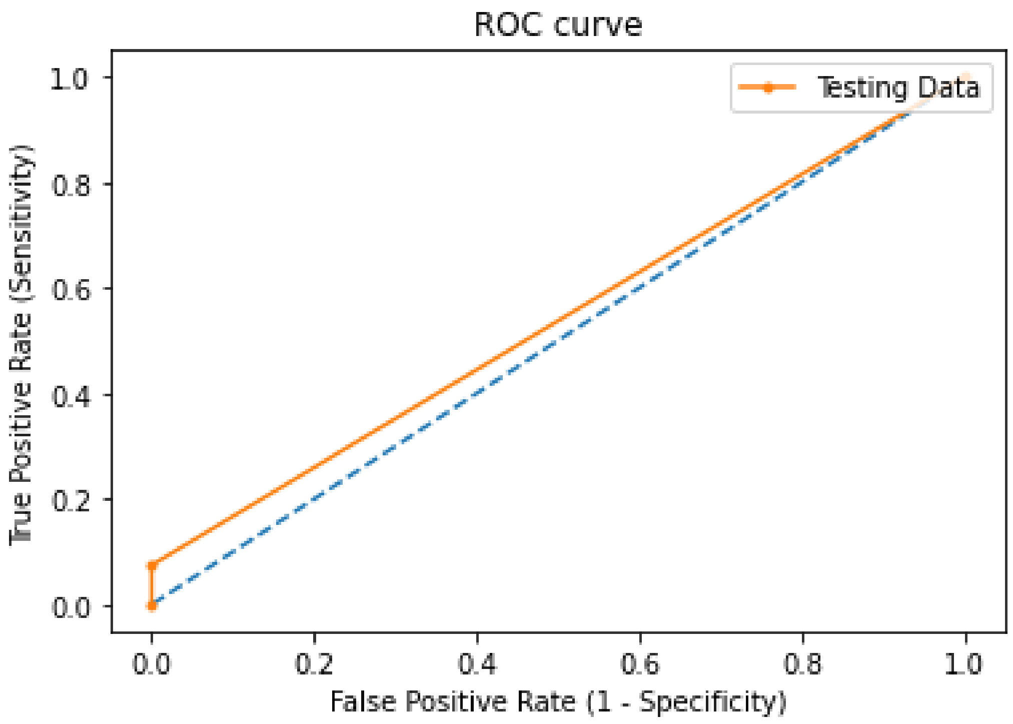



ROC curve
Testing Data
True Positive Rate (Sensitivity)
False Positive Rate (1 - Specificity)
0.0
0.2
0.4
0.6
0.8
1.0


36
11
1
66
Actual Negative:0
Actual Positive:1
Predict Negative:0
Predict Positive:1
60
50
40
30
20
10


ROC curve
Testing Data
True Positive Rate (Sensitivity)
False Positive Rate (1 - Specificity)
0.0
0.2
0.4
0.6
0.8
1.0


0
47
0
67
Actual Negative:0
Actual Positive:1
Predict Negative:0
Predict Positive:1
60
50
40
30
20
10
0


ROC curve
Testing Data
True Positive Rate (Sensitivity)
False Positive Rate (1 - Specificity)
0.0
0.2
0.4
0.6
0.8
1.0

```
[ ]:
```