iris-assignment-2

August 13, 2023

0.0.1 Iris Dataset classification using SVM, MLP and Random Forest classifier

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

[ ]:
```
from sklearn.datasets import load_iris
iris = load_iris()
df = pd.DataFrame(iris.data, columns = iris.feature_names)
df.head()
```

[ ]:
```
X = df.iloc[:][:]
y = iris["target"]
dict_svm = {}
dict_mlp = {}
dict_rfr = {}
RocAucSvm = {}
RocAucMlp = {}
RocAucRfr = {}
print(X, y)
```

0.0.2 Used for ploting confusion matrix

[4]:
```
def plot(y_test, y_pred):
  from sklearn.metrics import confusion_matrix
  import seaborn as sns

  print("Confusion Matrix : ")
  cf_matrix = confusion_matrix(y_test, y_pred)
  group_counts = ["{0:0.0f}".format(value) for value in
                  cf_matrix.flatten()]
  group_percentages = ["{0:.2%}".format(value) for value in
                       cf_matrix.flatten()/np.sum(cf_matrix)]
  labels = [f"{v1}\n{v2}" for v1, v2 in
            zip(group_counts,group_percentages)]
  labels = np.asarray(labels).reshape(3,3)
  plt.figure(figsize=(6, 4))
```

```
  sns.heatmap(cf_matrix, annot=labels, fmt='', cmap='Blues', xticklabels = iris.
 ↪target_names, yticklabels=iris.target_names)
  plt.xlabel('Predicted')
  plt.ylabel('Actual')
  plt.title('Confusion Matrix')
  plt.show()
  print("*********************************************")
```

[5]:
```
def reports(y_test, y_pred):
  from sklearn.metrics import classification_report
  plot(y_test, y_pred)
  print("*********************************************")
  print("Classification Evaluation : ")
  print(classification_report(y_test, y_pred, zero_division = 0))
```

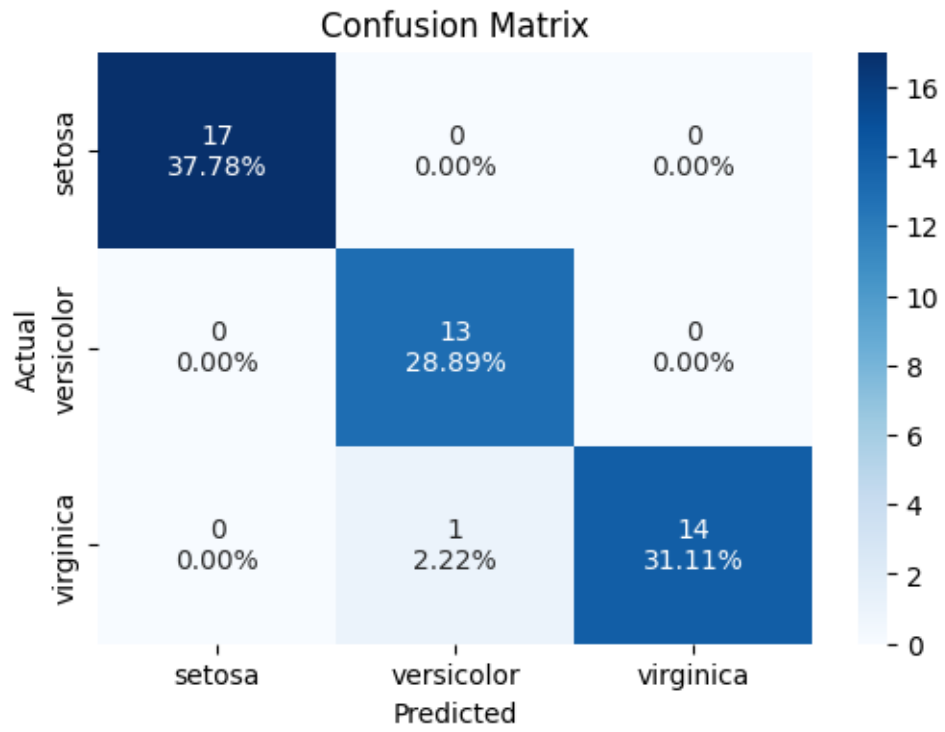
0.0.3 SVM CLASSIFIER

[6]:
```
def SVMClassifier(split, kernalValue = 'rbf', degreeValue = 3, gammaValue =␣
 ↪'scale', maxIter = -1):
  from sklearn.model_selection import train_test_split
  from sklearn.svm import SVC
  from sklearn.metrics import accuracy_score
  from sklearn.preprocessing import StandardScaler
  scaler = StandardScaler()
  scaler.fit(X)
  X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = split,␣
 ↪random_state=44)
  classifier = SVC(kernel = kernalValue, degree = degreeValue, gamma =␣
 ↪gammaValue, max_iter = maxIter)
  classifier.fit(X_train, y_train)
  y_pred = classifier.predict(X_test)
  accuracy = accuracy_score(y_test, y_pred)

  if str(split) in dict_svm:
    dict_svm[str(split)] = max(accuracy, dict_svm[str(split)])
    if str(split) == '0.3' and accuracy > dict_svm[str(split)]:
      RocAucSvm['max'] = {'y_test': y_test, 'y_pred': y_pred}
  else:
    dict_svm[str(split)] = accuracy
    if str(split) == '0.3':
      RocAucSvm['max'] = {'y_test': y_test, 'y_pred': y_pred}

  reports(y_test, y_pred)
```

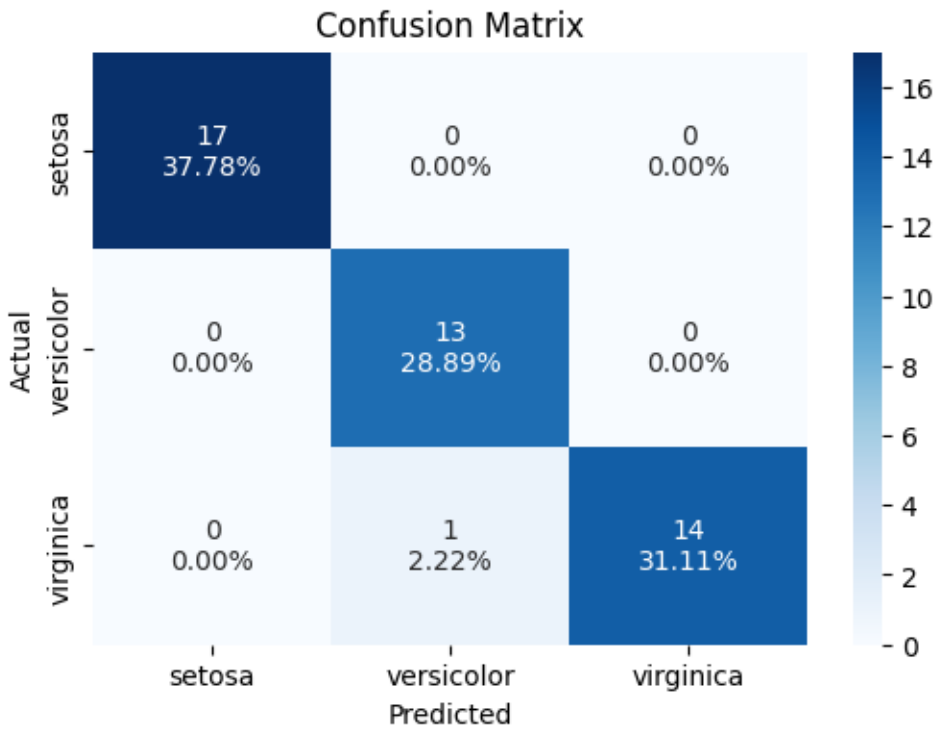
[7]:
```
#Train - Test split 70-30
SVMClassifier(0.3)
```

```
SVMClassifier(0.3, 'linear')
SVMClassifier(0.3, 'poly')
SVMClassifier(0.3, 'sigmoid', 3, 0.022)
```

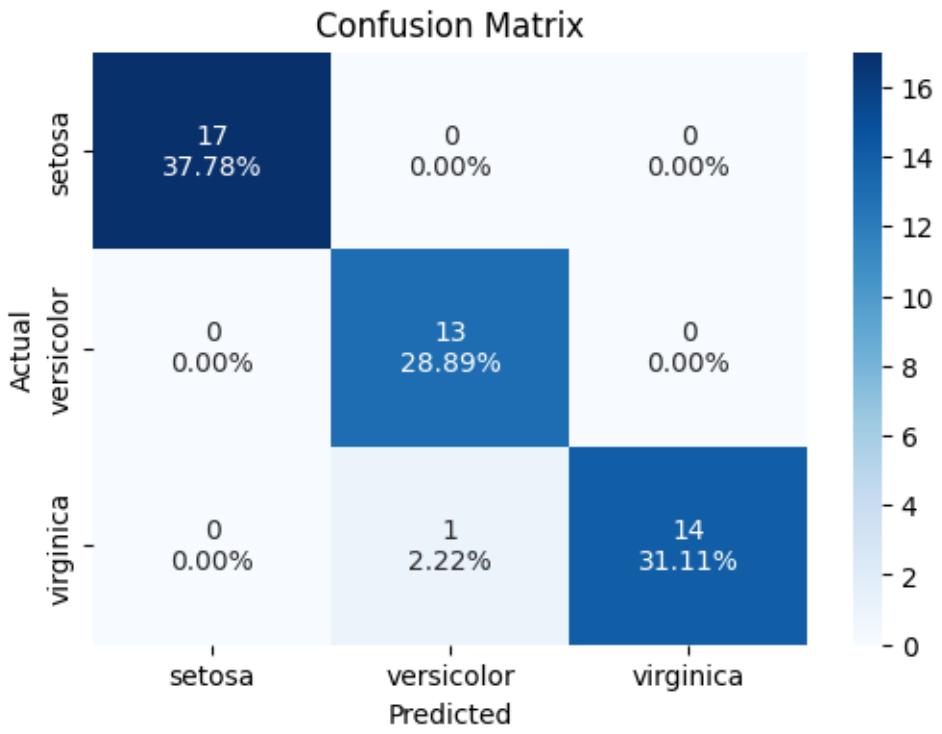
Confusion Matrix :



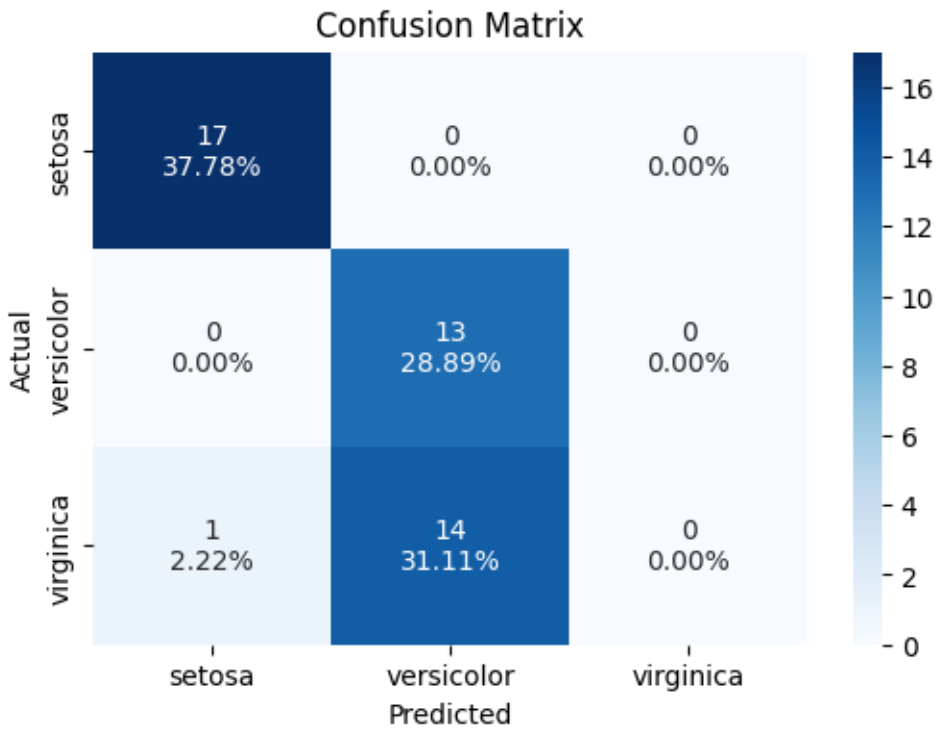

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        17
           1       0.93      1.00      0.96        13
           2       1.00      0.93      0.97        15

    accuracy                           0.98        45
   macro avg       0.98      0.98      0.98        45
weighted avg       0.98      0.98      0.98        45

Confusion Matrix :
```




```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        17
           1       0.93      1.00      0.96        13
           2       1.00      0.93      0.97        15

    accuracy                           0.98        45
   macro avg       0.98      0.98      0.98        45
weighted avg       0.98      0.98      0.98        45

Confusion Matrix :
```



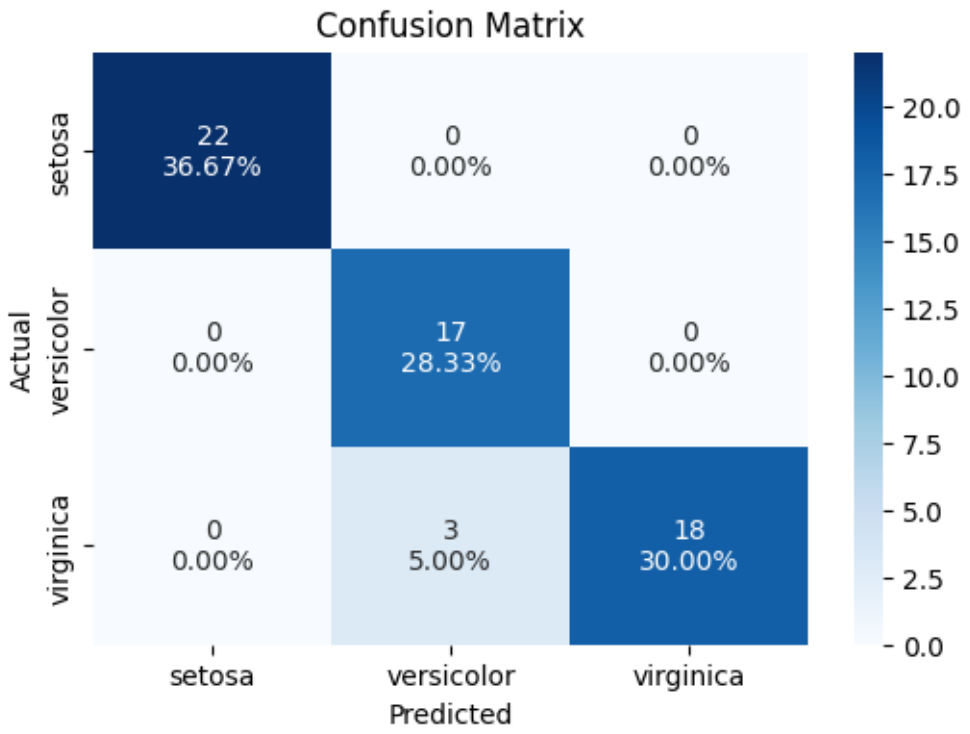

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        17
           1       0.93      1.00      0.96        13
           2       1.00      0.93      0.97        15

    accuracy                           0.98        45
   macro avg       0.98      0.98      0.98        45
weighted avg       0.98      0.98      0.98        45

Confusion Matrix :
```



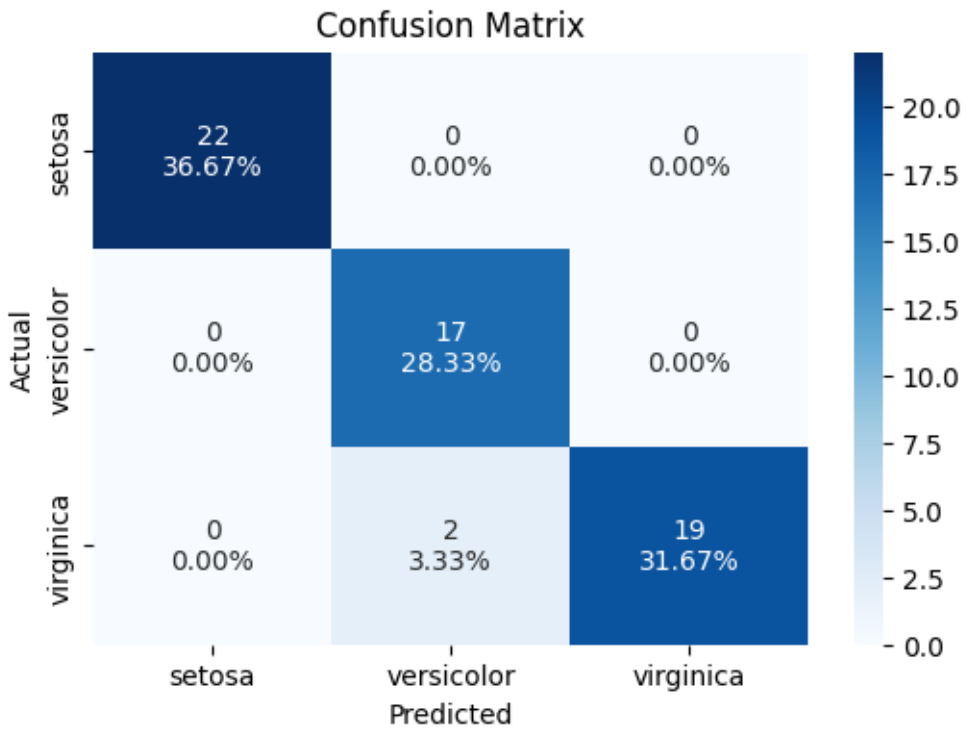

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.94      1.00      0.97        17
           1       0.48      1.00      0.65        13
           2       0.00      0.00      0.00        15

    accuracy                           0.67        45
   macro avg       0.48      0.67      0.54        45
weighted avg       0.50      0.67      0.55        45
```

[8]:
```
#Train - Test split 60-40
SVMClassifier(0.4, 'rbf', 3, 'auto')
SVMClassifier(0.4, 'linear')
SVMClassifier(0.4, 'poly')
SVMClassifier(0.4, 'sigmoid', 3, 0.023 ) #wrost performance
```

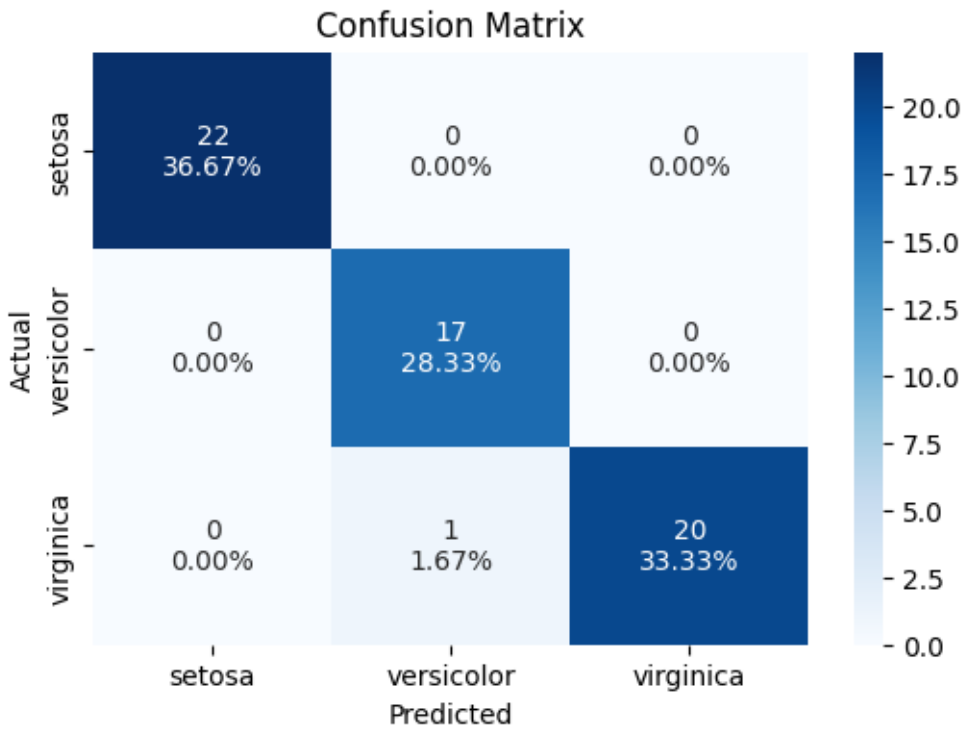
```
Confusion Matrix :
```



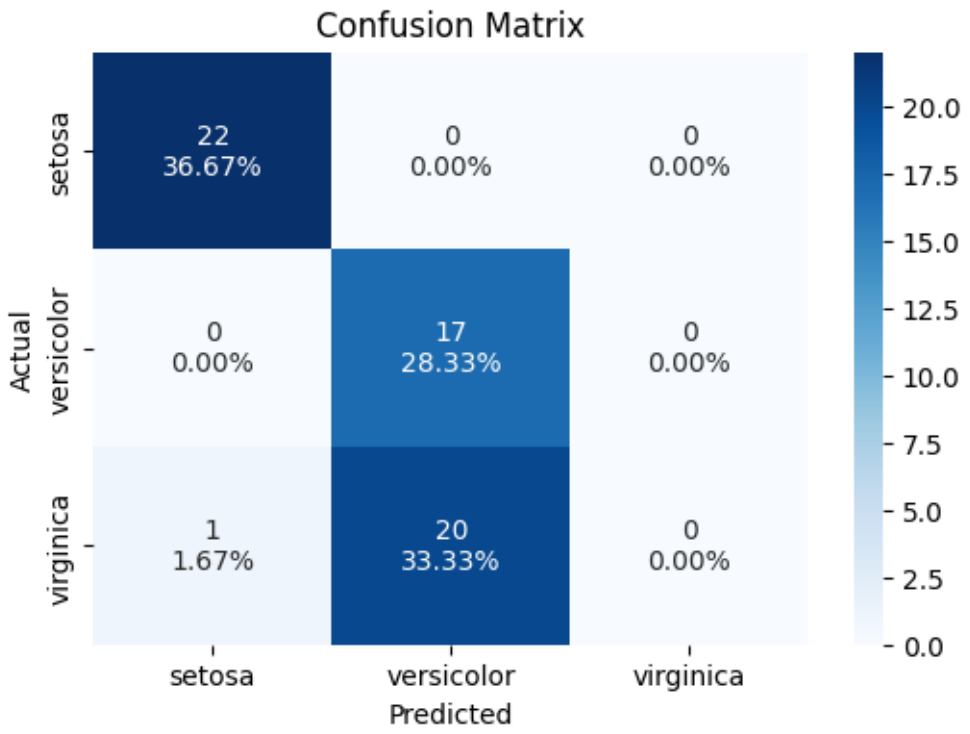

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        22
           1       0.85      1.00      0.92        17
           2       1.00      0.86      0.92        21

    accuracy                           0.95        60
   macro avg       0.95      0.95      0.95        60
weighted avg       0.96      0.95      0.95        60

Confusion Matrix :
```




```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        22
           1       0.89      1.00      0.94        17
           2       1.00      0.90      0.95        21

    accuracy                           0.97        60
   macro avg       0.96      0.97      0.96        60
weighted avg       0.97      0.97      0.97        60

Confusion Matrix :
```



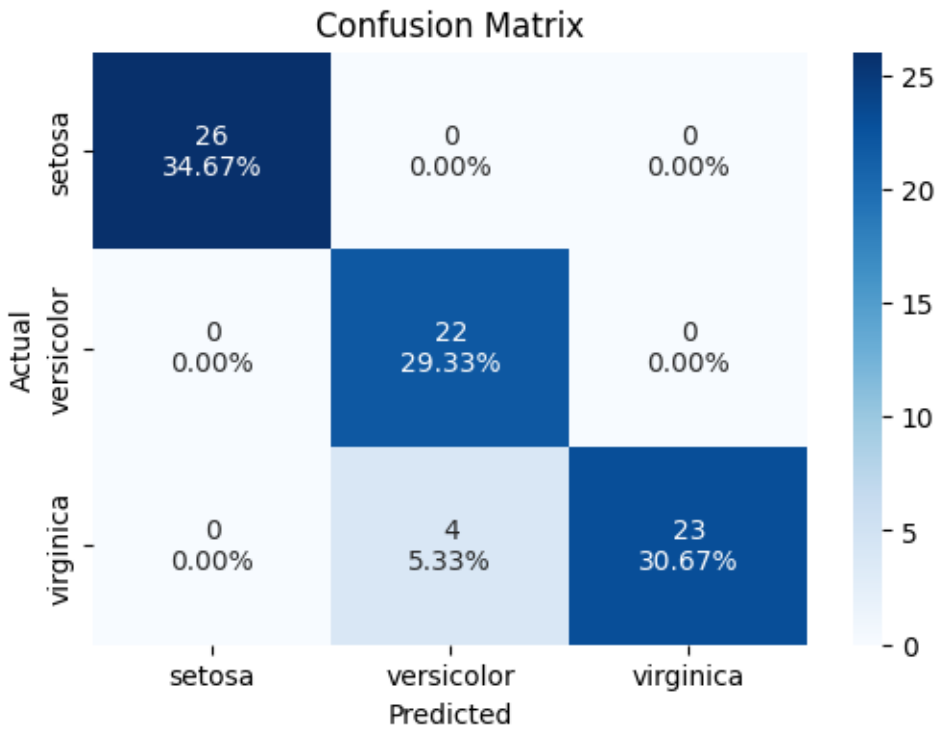

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        22
           1       0.94      1.00      0.97        17
           2       1.00      0.95      0.98        21

    accuracy                           0.98        60
   macro avg       0.98      0.98      0.98        60
weighted avg       0.98      0.98      0.98        60

Confusion Matrix :
```



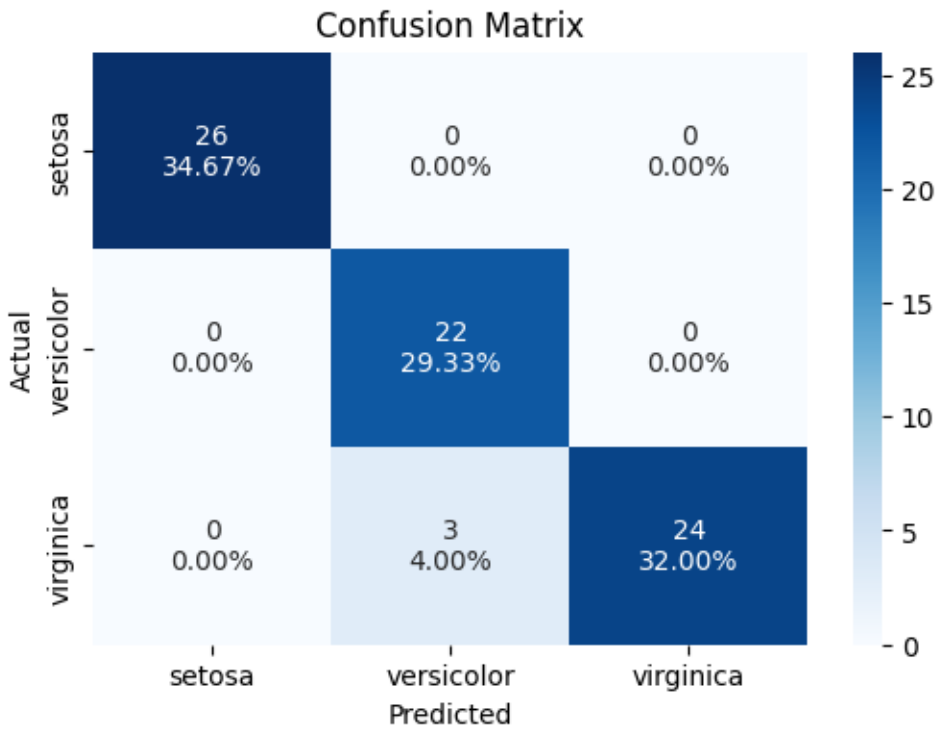

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.96      1.00      0.98        22
           1       0.46      1.00      0.63        17
           2       0.00      0.00      0.00        21

    accuracy                           0.65        60
   macro avg       0.47      0.67      0.54        60
weighted avg       0.48      0.65      0.54        60
```

[9]:
```
#Train - Test split 50-50
SVMClassifier(0.5, 'rbf', 3, 'auto')
SVMClassifier(0.5, 'linear')
SVMClassifier(0.5, 'poly')
SVMClassifier(0.5, 'sigmoid', 3, 0.022) #wrost performance
```

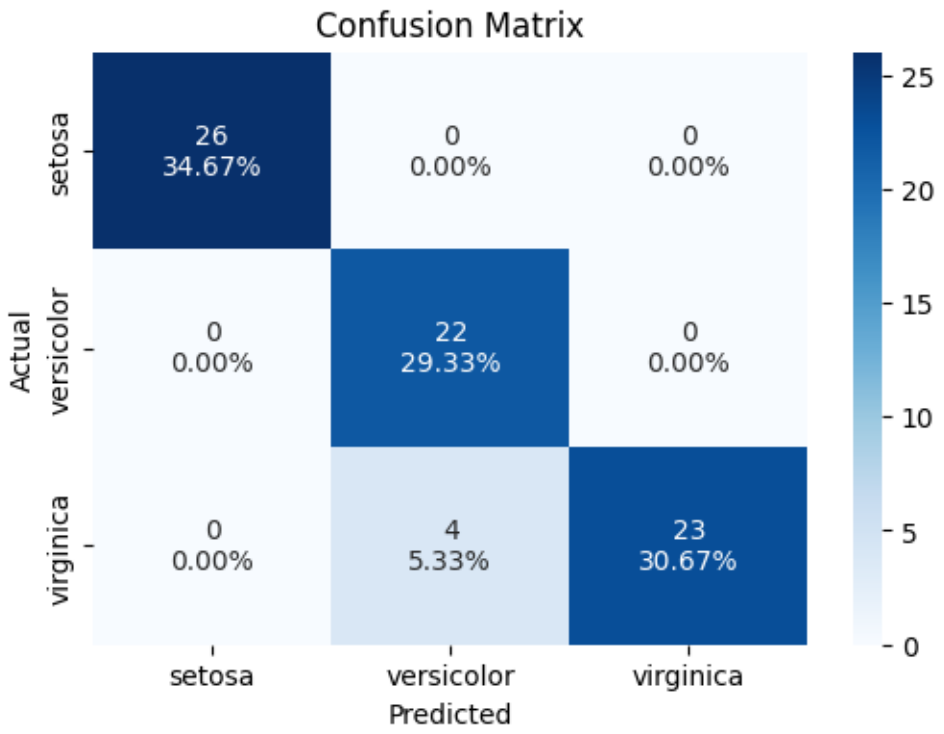
```
Confusion Matrix :
```



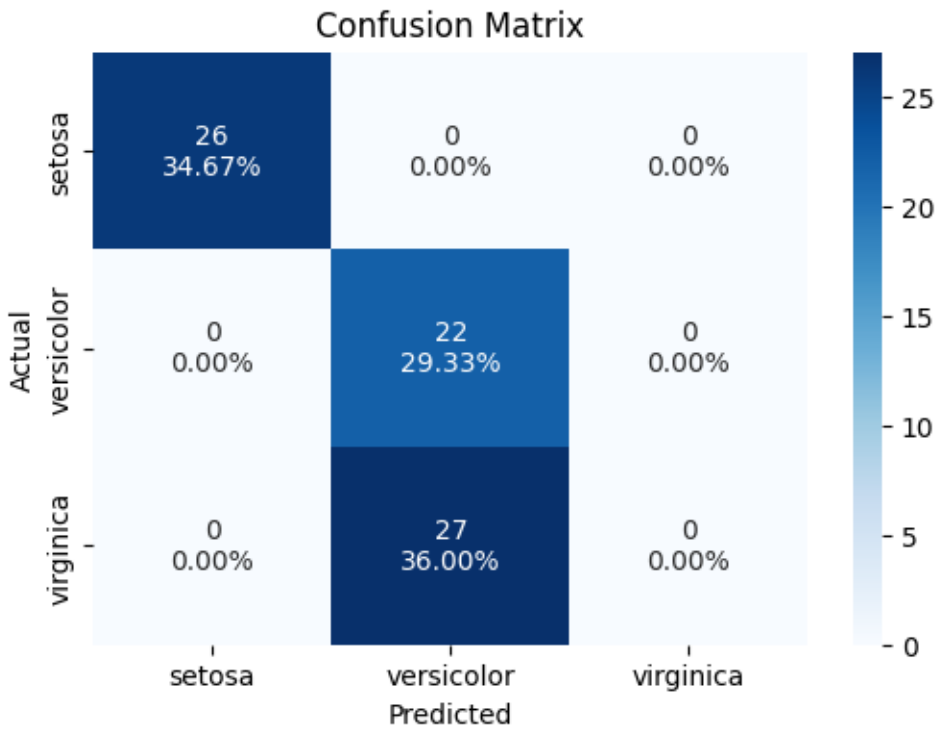

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        26
           1       0.85      1.00      0.92        22
           2       1.00      0.85      0.92        27

    accuracy                           0.95        75
   macro avg       0.95      0.95      0.95        75
weighted avg       0.95      0.95      0.95        75

Confusion Matrix :
```




```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        26
           1       0.88      1.00      0.94        22
           2       1.00      0.89      0.94        27

    accuracy                           0.96        75
   macro avg       0.96      0.96      0.96        75
weighted avg       0.96      0.96      0.96        75

Confusion Matrix :
```



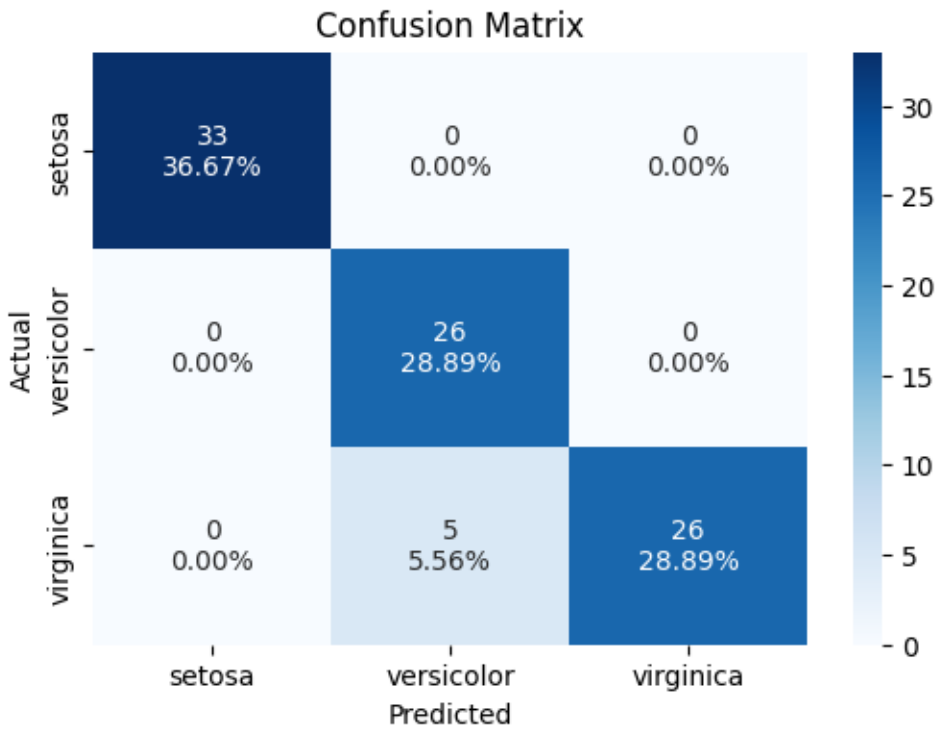

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        26
           1       0.85      1.00      0.92        22
           2       1.00      0.85      0.92        27

    accuracy                           0.95        75
   macro avg       0.95      0.95      0.95        75
weighted avg       0.95      0.95      0.95        75

Confusion Matrix :
```



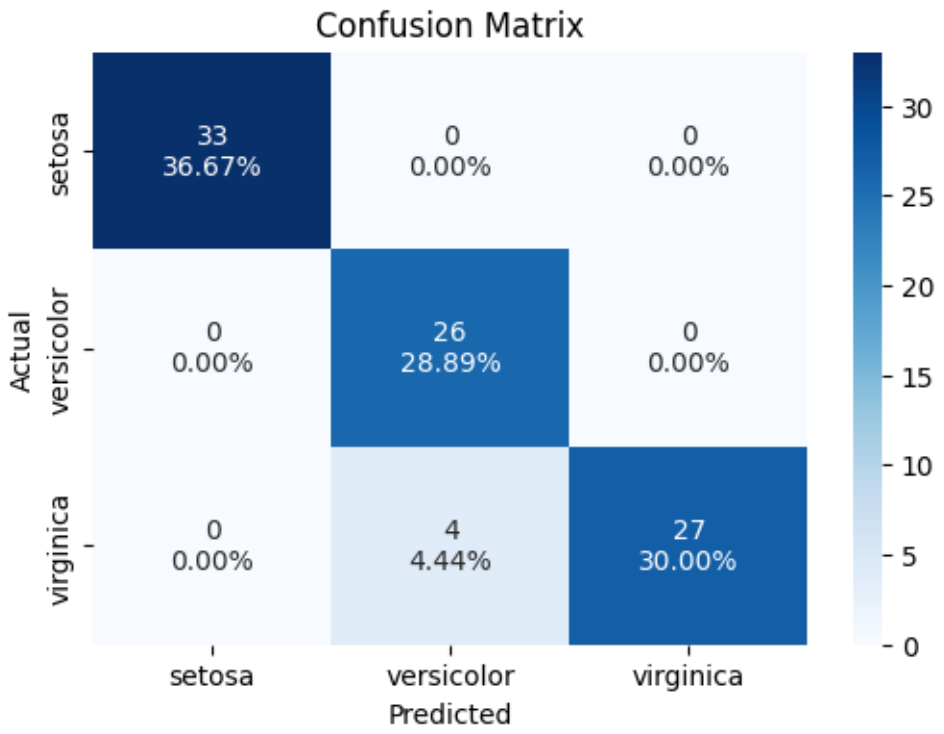

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        26
           1       0.45      1.00      0.62        22
           2       0.00      0.00      0.00        27

    accuracy                           0.64        75
   macro avg       0.48      0.67      0.54        75
weighted avg       0.48      0.64      0.53        75
```

[10]:
```
#Train - Test split 40-60
SVMClassifier(0.6, 'rbf', 3, 'auto')
SVMClassifier(0.6, 'linear')
SVMClassifier(0.6, 'poly')
SVMClassifier(0.6, 'sigmoid', 3, 0.015 ) #wrost performance
```

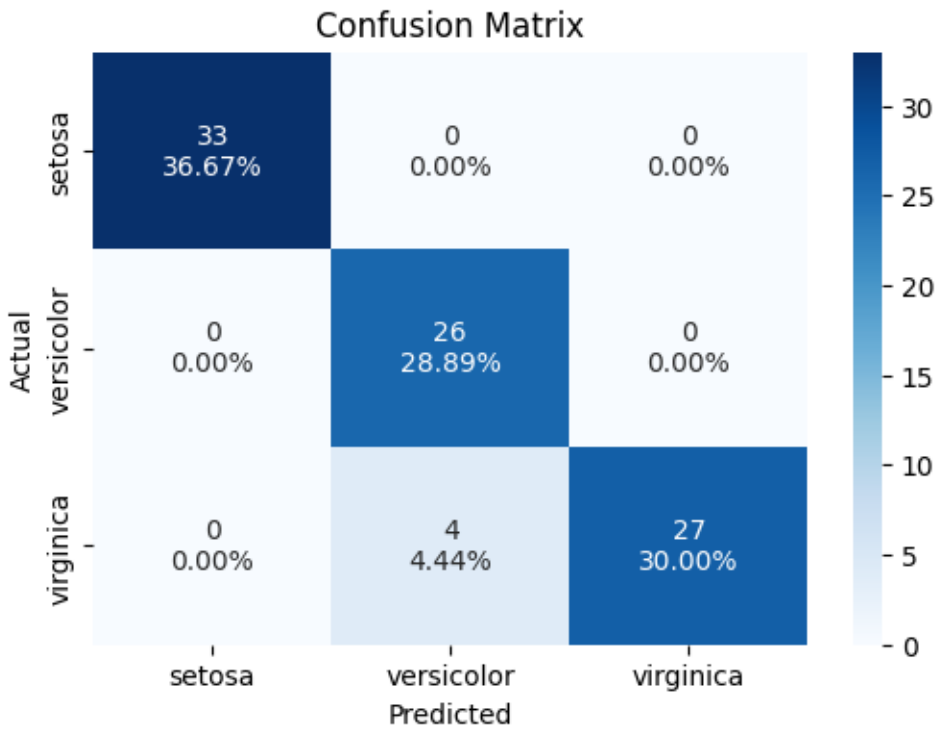
```
Confusion Matrix :
```



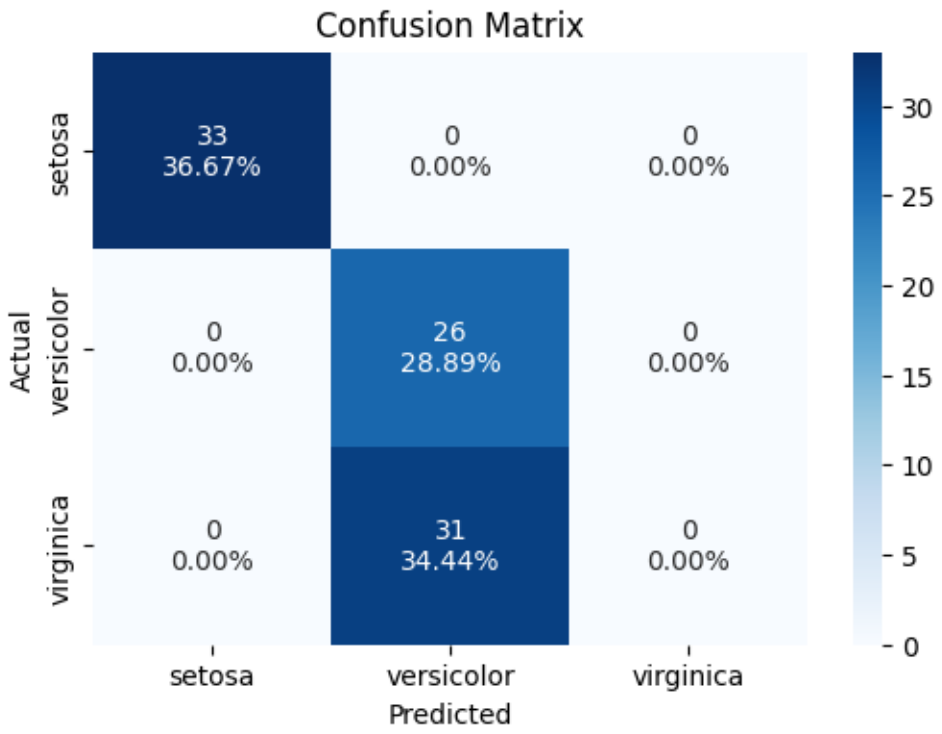

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        33
           1       0.84      1.00      0.91        26
           2       1.00      0.84      0.91        31

    accuracy                           0.94        90
   macro avg       0.95      0.95      0.94        90
weighted avg       0.95      0.94      0.94        90

Confusion Matrix :
```




```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        33
           1       0.87      1.00      0.93        26
           2       1.00      0.87      0.93        31

    accuracy                           0.96        90
   macro avg       0.96      0.96      0.95        90
weighted avg       0.96      0.96      0.96        90

Confusion Matrix :
```



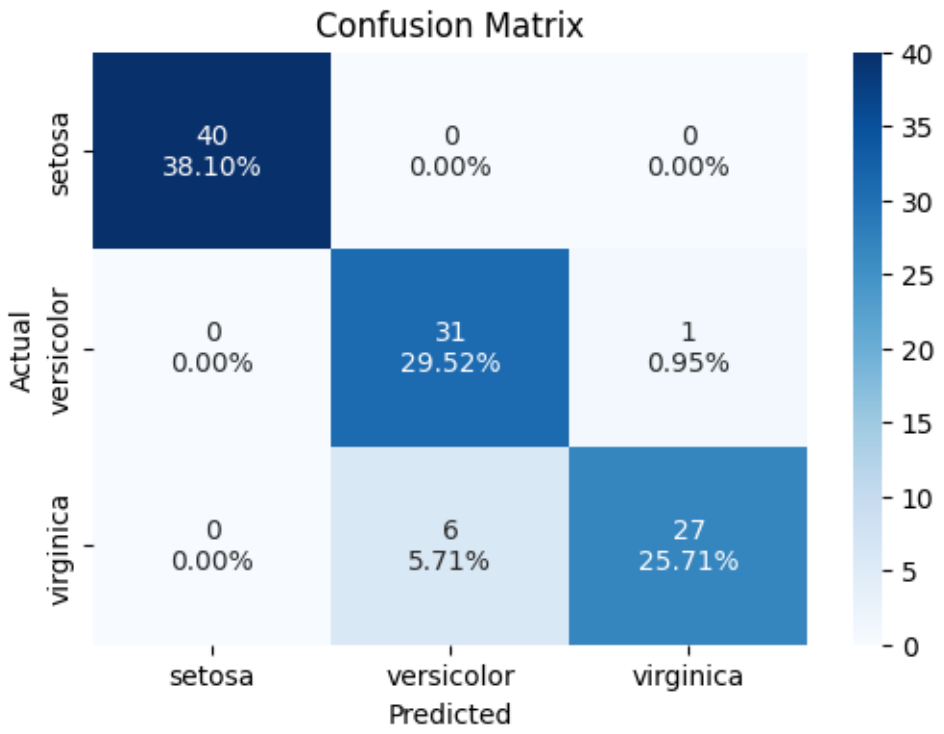

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        33
           1       0.87      1.00      0.93        26
           2       1.00      0.87      0.93        31

    accuracy                           0.96        90
   macro avg       0.96      0.96      0.95        90
weighted avg       0.96      0.96      0.96        90

Confusion Matrix :
```



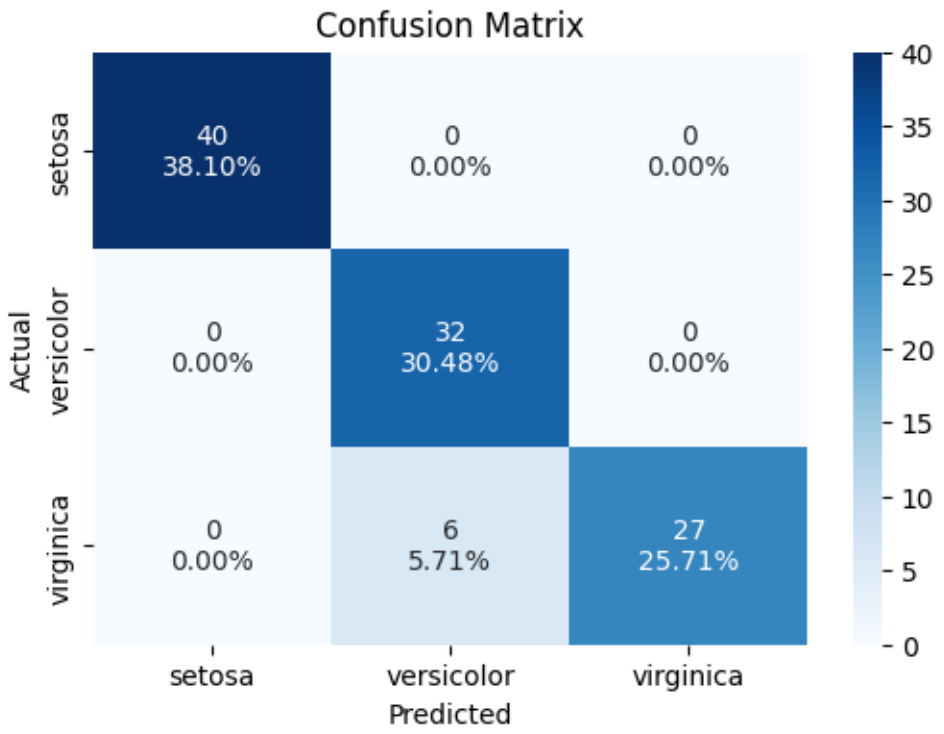

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        33
           1       0.46      1.00      0.63        26
           2       0.00      0.00      0.00        31

    accuracy                           0.66        90
   macro avg       0.49      0.67      0.54        90
weighted avg       0.50      0.66      0.55        90
```

[11]:
```
#Train - Test split 30-70
SVMClassifier(0.7, 'rbf', 3, 'auto')
SVMClassifier(0.7, 'linear')
SVMClassifier(0.7, 'poly')
SVMClassifier(0.7, 'sigmoid', 3, 2 ) #wrost performance
```

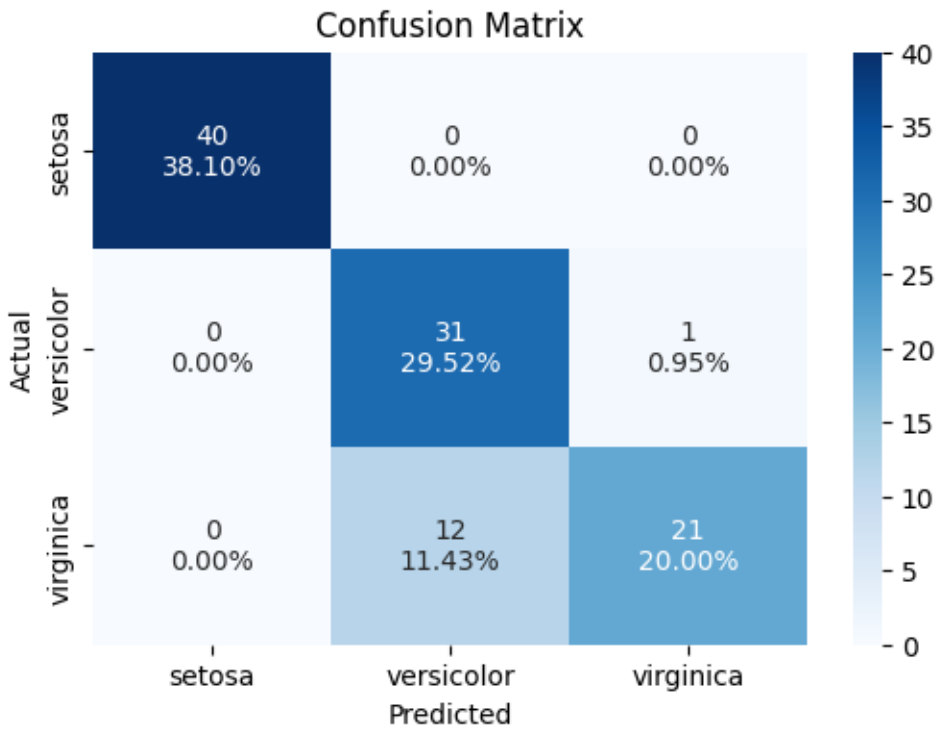
```
Confusion Matrix :
```



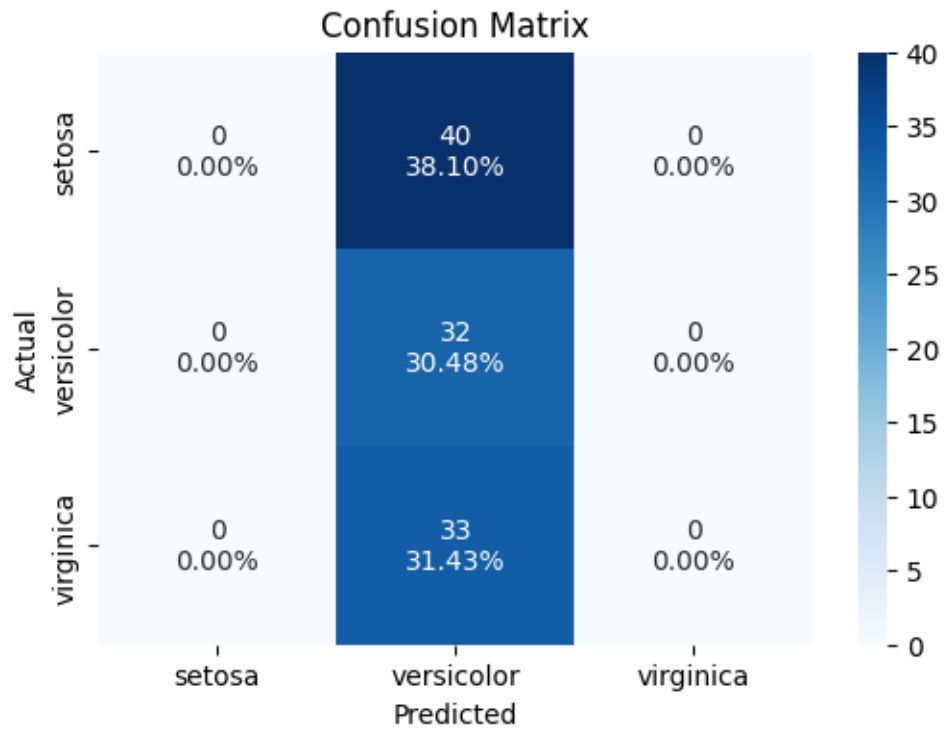

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        40
           1       0.84      0.97      0.90        32
           2       0.96      0.82      0.89        33

    accuracy                           0.93       105
   macro avg       0.93      0.93      0.93       105
weighted avg       0.94      0.93      0.93       105

Confusion Matrix :
```




```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        40
           1       0.84      1.00      0.91        32
           2       1.00      0.82      0.90        33

    accuracy                           0.94       105
   macro avg       0.95      0.94      0.94       105
weighted avg       0.95      0.94      0.94       105

Confusion Matrix :
```



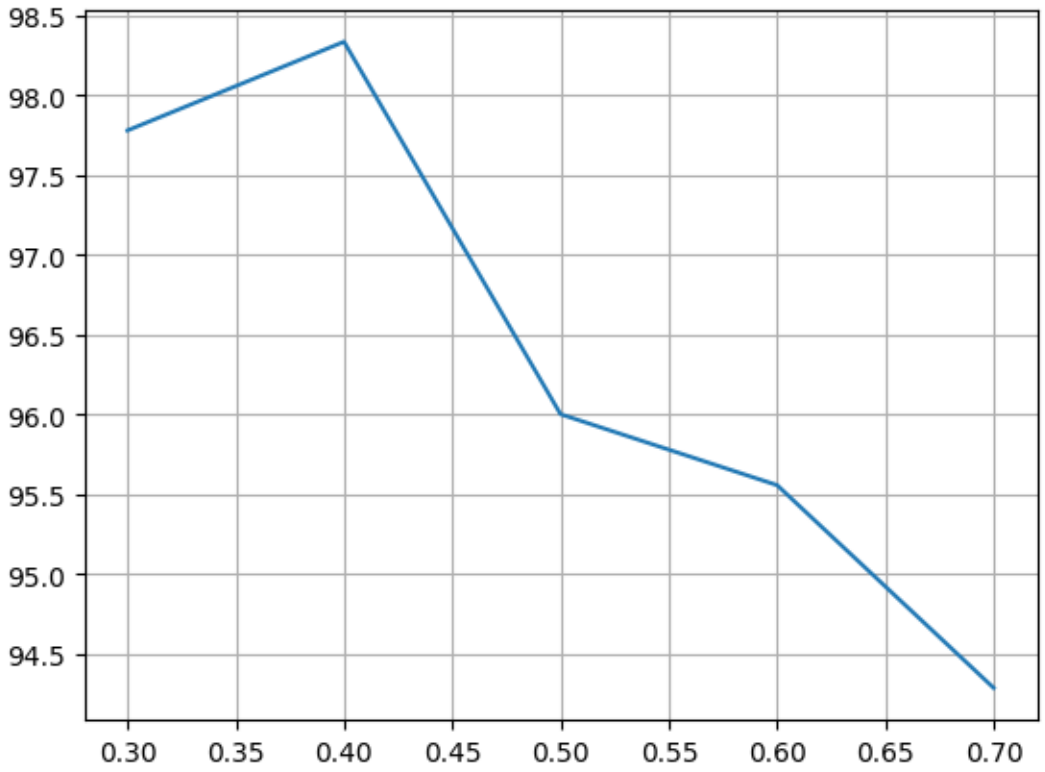

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        40
           1       0.72      0.97      0.83        32
           2       0.95      0.64      0.76        33

    accuracy                           0.88       105
   macro avg       0.89      0.87      0.86       105
weighted avg       0.90      0.88      0.87       105

Confusion Matrix :
```




```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        40
           1       0.30      1.00      0.47        32
           2       0.00      0.00      0.00        33

    accuracy                           0.30       105
   macro avg       0.10      0.33      0.16       105
weighted avg       0.09      0.30      0.14       105
```

0.0.4 split vs accuracy graph

[12]:
```
x_points = [float(key) for key in dict_svm]
y_points = [i*100 for i in dict_svm.values()]
plt.plot(x_points, y_points)
plt.grid(True)
plt.show()
```



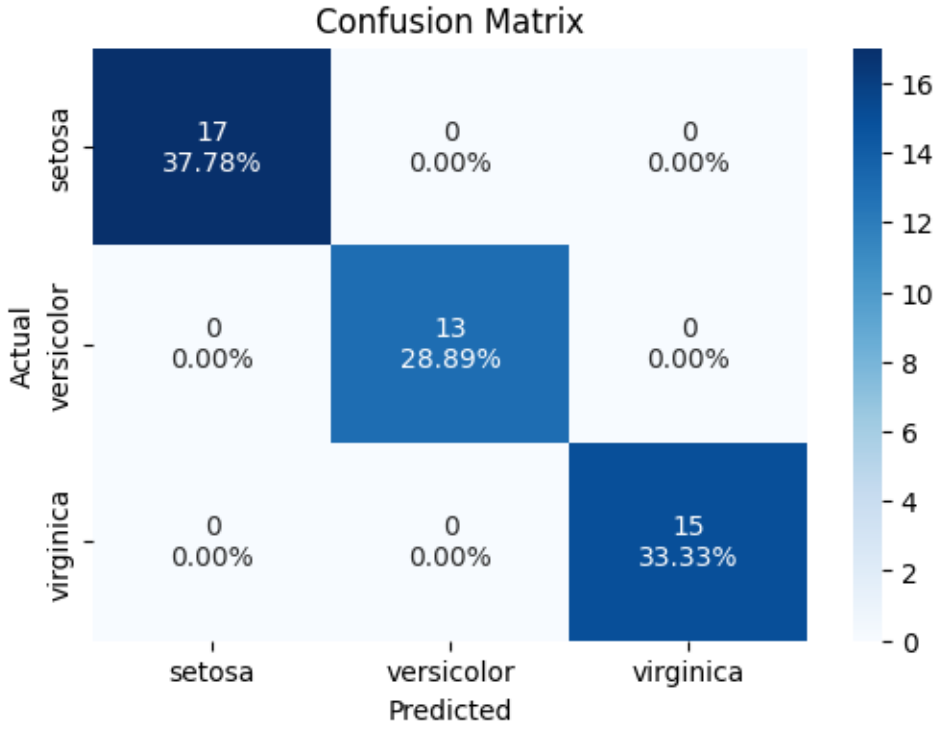
0.0.5 MLP Classifier

[13]:
```
def MLPClassifier(split, hiddenLayerSize = [100, ], activationValue = 'relu',␣
↪solverValue = 'adam'):
 from sklearn.model_selection import train_test_split
 from sklearn.neural_network import MLPClassifier
 from sklearn.metrics import accuracy_score
 X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = split,␣
↪random_state=44)
 classifier = MLPClassifier(hidden_layer_sizes = hiddenLayerSize, activation =␣
↪activationValue, solver = solverValue, random_state = 1)
 classifier.fit(X_train, y_train)
 y_pred = classifier.predict(X_test)
 accuracy = accuracy_score(y_test, y_pred)

 if(str(split) in dict_mlp):
   dict_mlp[str(split)] = max(accuracy, dict_mlp[str(split)])
   if(str(split) == '0.3' and accuracy > dict_svm[str(split)]):
     RocAucMlp['max'] = {'y_test': y_test, 'y_pred': y_pred}
 else:
   dict_mlp[str(split)] = accuracy
   RocAucMlp['max'] = {'y_test': y_test, 'y_pred': y_pred}
```

```
  reports(y_test, y_pred)
```

[14]:
```
#Train - Test split 70-30
MLPClassifier(0.3, [30, ])
```

```
/home/aqeel/.local/lib/python3.10/site-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and
the optimization hasn't converged yet.
  warnings.warn(

Confusion Matrix :
```

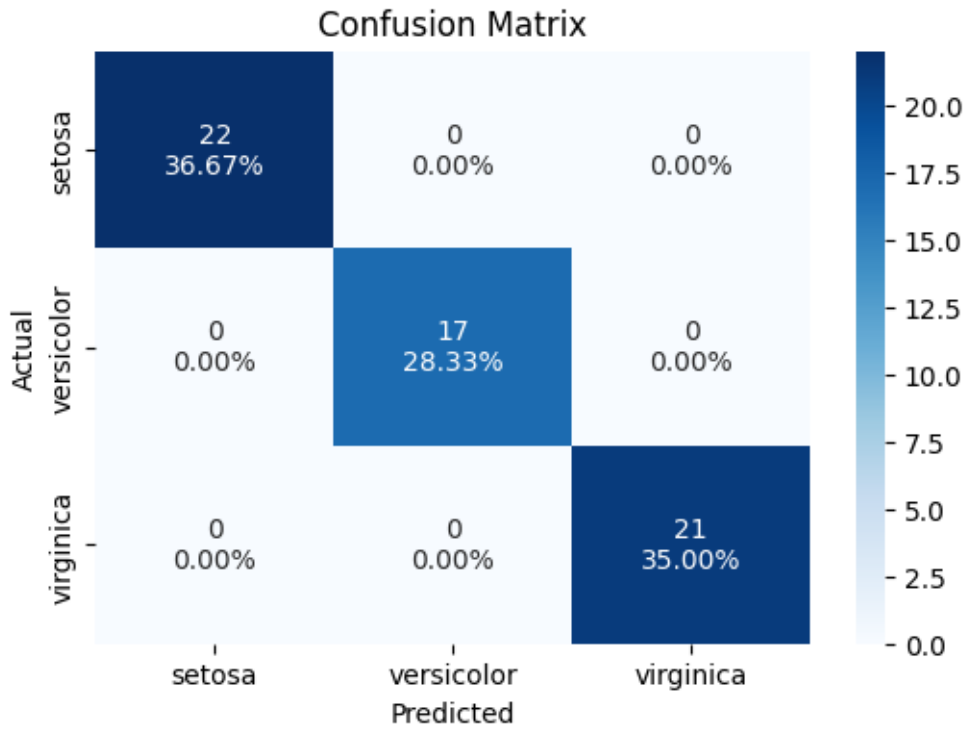



```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        17
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00        15

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
```

```
weighted avg       1.00      1.00      1.00        45
```

[15]:
```
#Train - Test split 60-40
MLPClassifier(0.4, [35, ])
```

```
Confusion Matrix :

/home/aqeel/.local/lib/python3.10/site-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and
the optimization hasn't converged yet.
  warnings.warn(
```

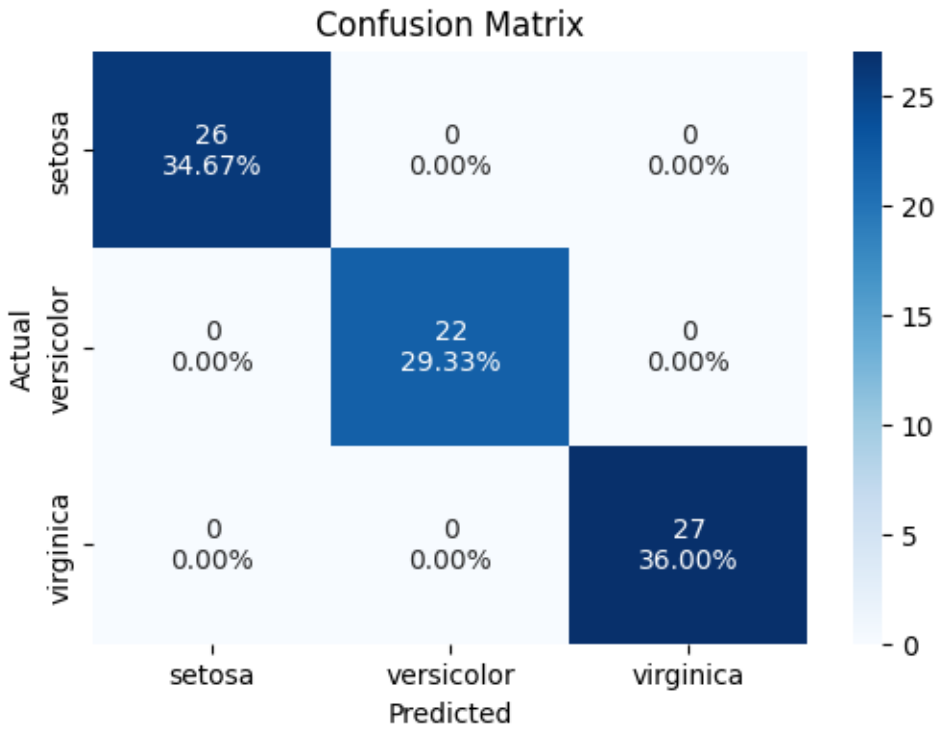



```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        22
           1       1.00      1.00      1.00        17
           2       1.00      1.00      1.00        21

    accuracy                           1.00        60
```

```
   macro avg       1.00      1.00      1.00        60
weighted avg       1.00      1.00      1.00        60
```

[16]:
```
#Train - Test split 50-50
MLPClassifier(0.5, [35, ])
```

```
Confusion Matrix :

/home/aqeel/.local/lib/python3.10/site-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and
the optimization hasn't converged yet.
  warnings.warn(
```

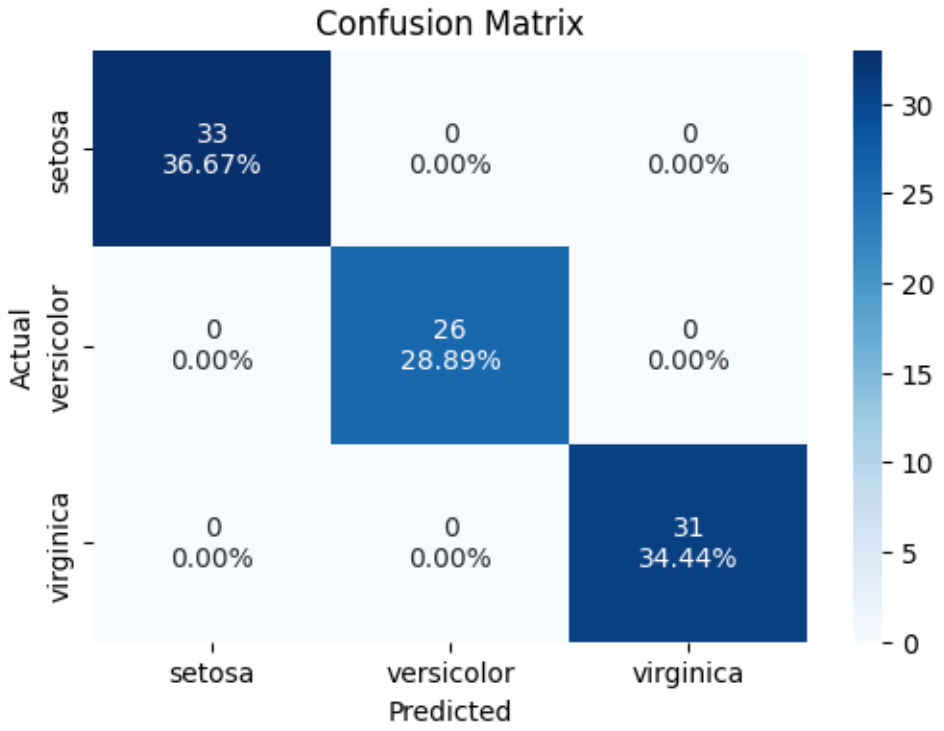



```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        26
           1       1.00      1.00      1.00        22
           2       1.00      1.00      1.00        27
```

```
    accuracy                           1.00        75
   macro avg       1.00      1.00      1.00        75
weighted avg       1.00      1.00      1.00        75
```

[17]:
```
#Train - Test split 40-60
MLPClassifier(0.6, [45, 8])
```

```
Confusion Matrix :

/home/aqeel/.local/lib/python3.10/site-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and
the optimization hasn't converged yet.
  warnings.warn(
```

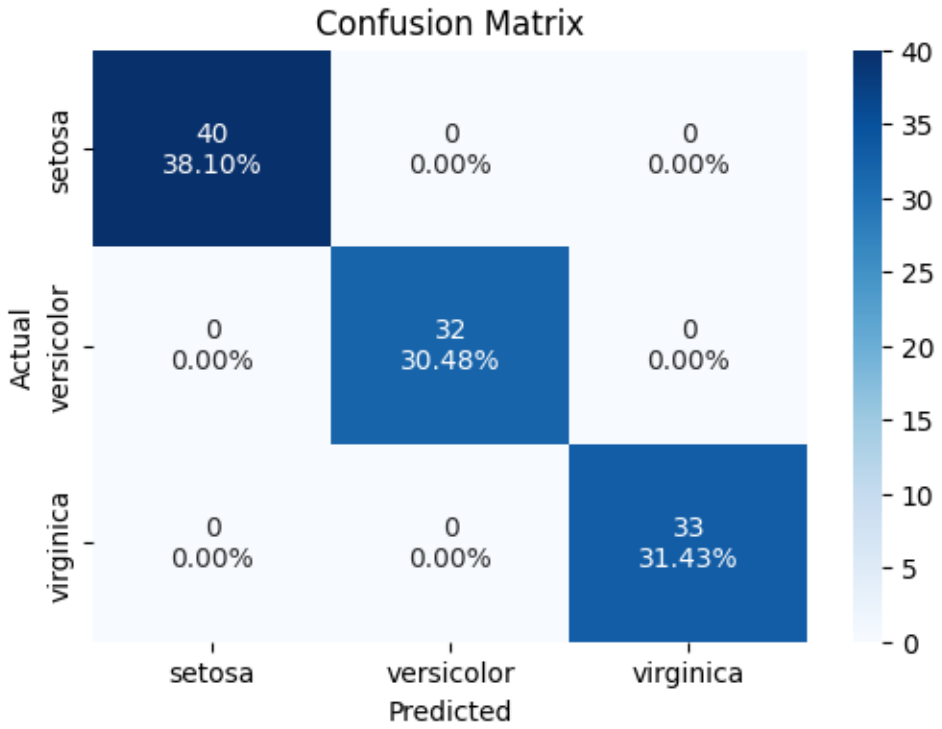



```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        33
           1       1.00      1.00      1.00        26
           2       1.00      1.00      1.00        31
```

```
    accuracy                           1.00        90
   macro avg       1.00      1.00      1.00        90
weighted avg       1.00      1.00      1.00        90
```

[18]:
```
#Train - Test split 30-70
MLPClassifier(0.7, [50, 10])
```

```
/home/aqeel/.local/lib/python3.10/site-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and
the optimization hasn't converged yet.
  warnings.warn(

Confusion Matrix :
```

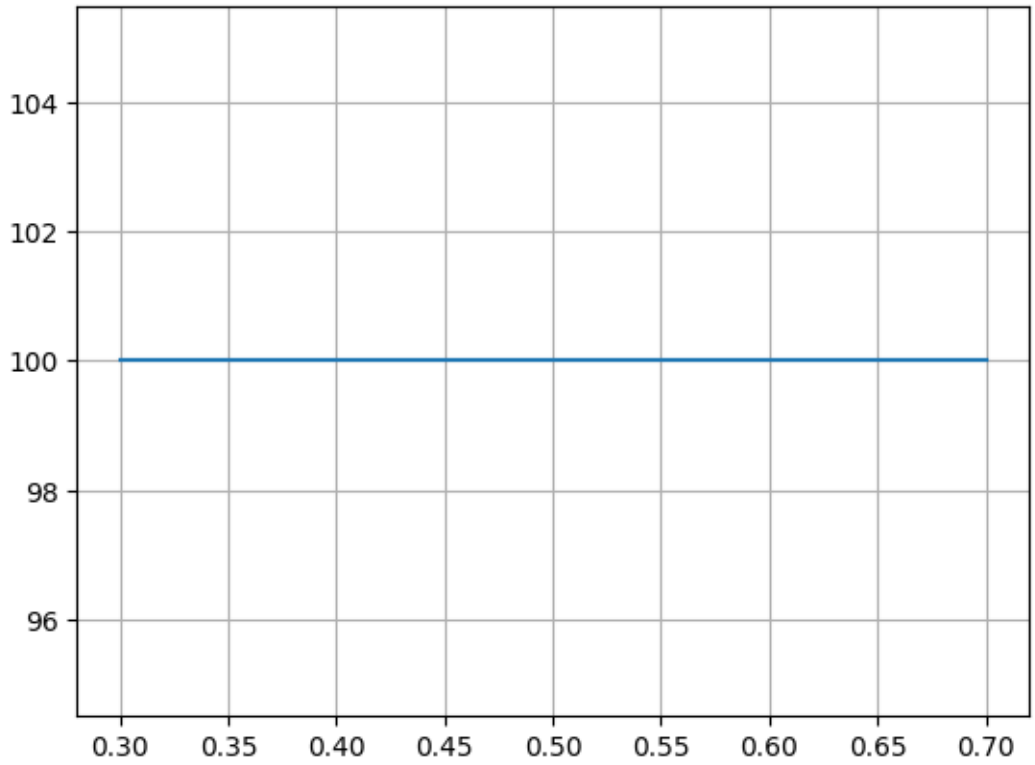



```
*******************************************
*******************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        40
           1       1.00      1.00      1.00        32
```

```
           2       1.00      1.00      1.00        33

    accuracy                           1.00       105
   macro avg       1.00      1.00      1.00       105
weighted avg       1.00      1.00      1.00       105
```

0.0.6 split vs accuracy graph

[19]:
```
x_points = [float(key) for key in dict_mlp]
y_points = [i*100 for i in dict_mlp.values()]
plt.plot(x_points, y_points)
plt.grid(True)
plt.show()
```



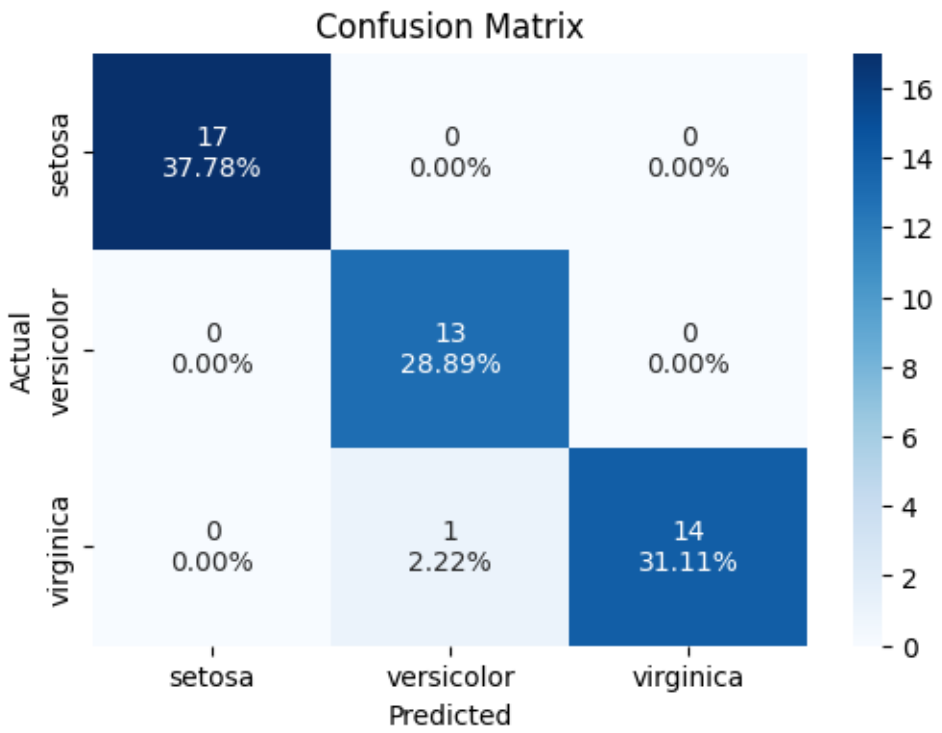
0.0.7 Random Forest Classifier

[20]:
```
def randomForest(split, estimator = 100, criterionValue = 'gini', ):
  from sklearn.model_selection import train_test_split
  from sklearn.ensemble import RandomForestClassifier
  from sklearn.metrics import accuracy_score
  from sklearn.preprocessing import StandardScaler
```

```
  scaler = StandardScaler()
  scaler.fit(X)
  X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = split,␣
↪random_state=44)
  classifier = RandomForestClassifier(n_estimators = estimator, criterion =␣
↪criterionValue)
  classifier.fit(X_train, y_train)
  y_pred = classifier.predict(X_test)
  accuracy = accuracy_score(y_test, y_pred)

  if(str(split) in dict_rfr):
    dict_rfr[str(split)] = max(accuracy, dict_rfr[str(split)])
    if(str(split) == '0.3' and accuracy > dict_svm[str(split)]):
      RocAucRfr['max'] = {'y_test': y_test, 'y_pred': y_pred}
  else:
    dict_rfr[str(split)] = accuracy
    if(str(split) == '0.3'):
      RocAucRfr['max'] = {'y_test': y_test, 'y_pred': y_pred}

  reports(y_test, y_pred)
```

[21]:
```
randomForest(0.3)
```

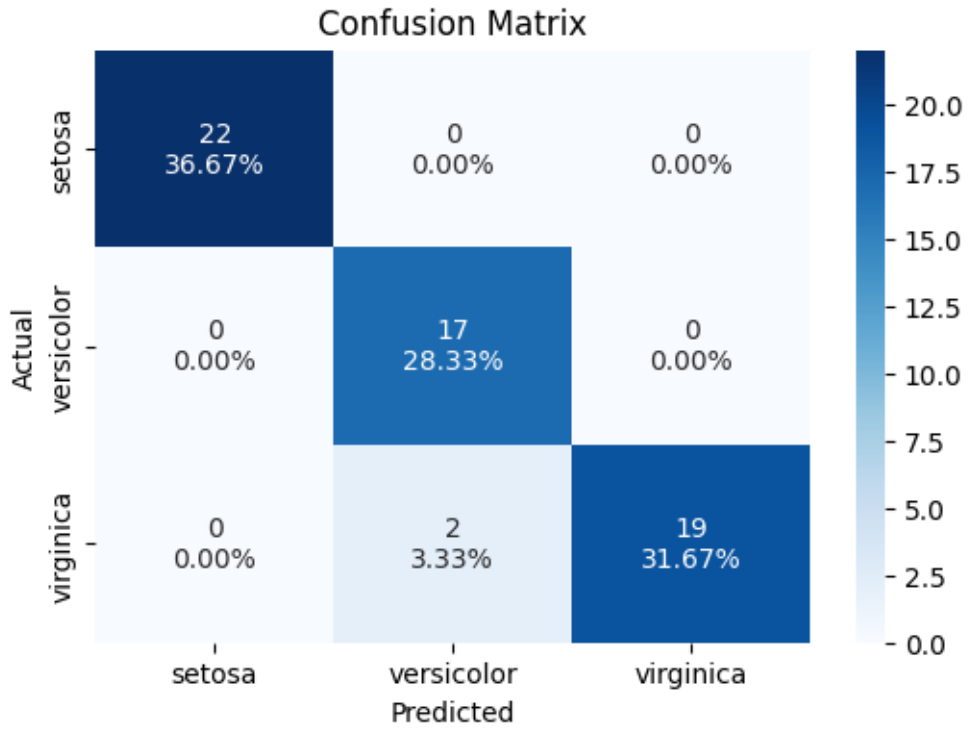
```
Confusion Matrix :
```



```
*********************************************
*********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        17
           1       0.93      1.00      0.96        13
           2       1.00      0.93      0.97        15

    accuracy                           0.98        45
   macro avg       0.98      0.98      0.98        45
weighted avg       0.98      0.98      0.98        45
```

[22]:
```
randomForest(0.4)
```

```
Confusion Matrix :
```

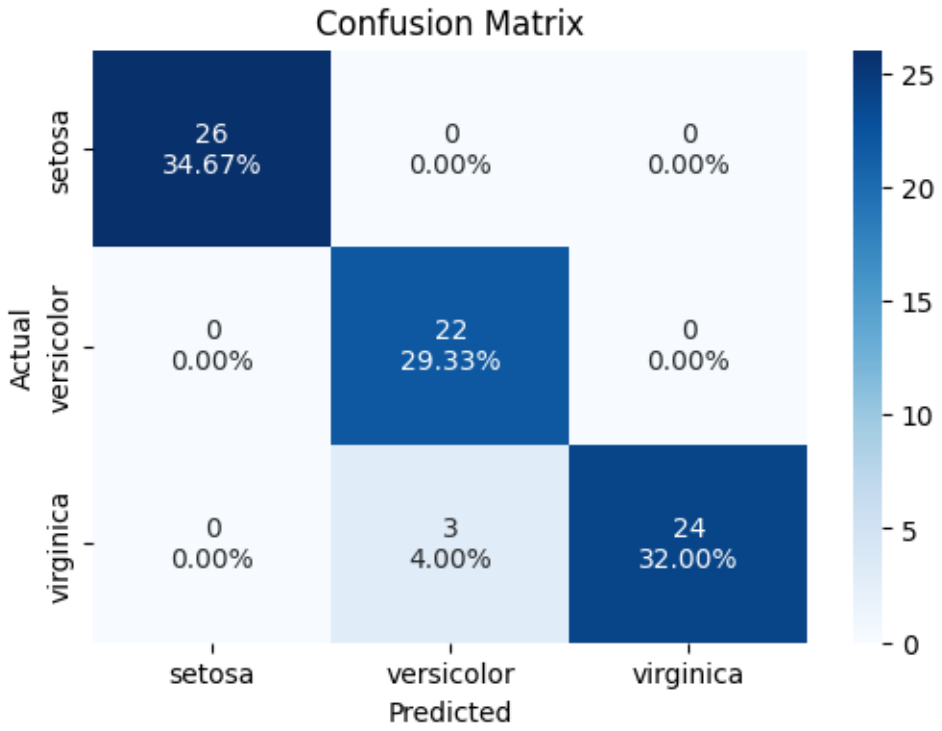



```
*********************************************
*********************************************
Classification Evaluation :
              precision    recall  f1-score   support
```

```
              0       1.00      1.00      1.00        22
              1       0.89      1.00      0.94        17
              2       1.00      0.90      0.95        21

       accuracy                           0.97        60
      macro avg       0.96      0.97      0.96        60
   weighted avg       0.97      0.97      0.97        60
```

[23]:
```
randomForest(0.5)
```

```
Confusion Matrix :
```

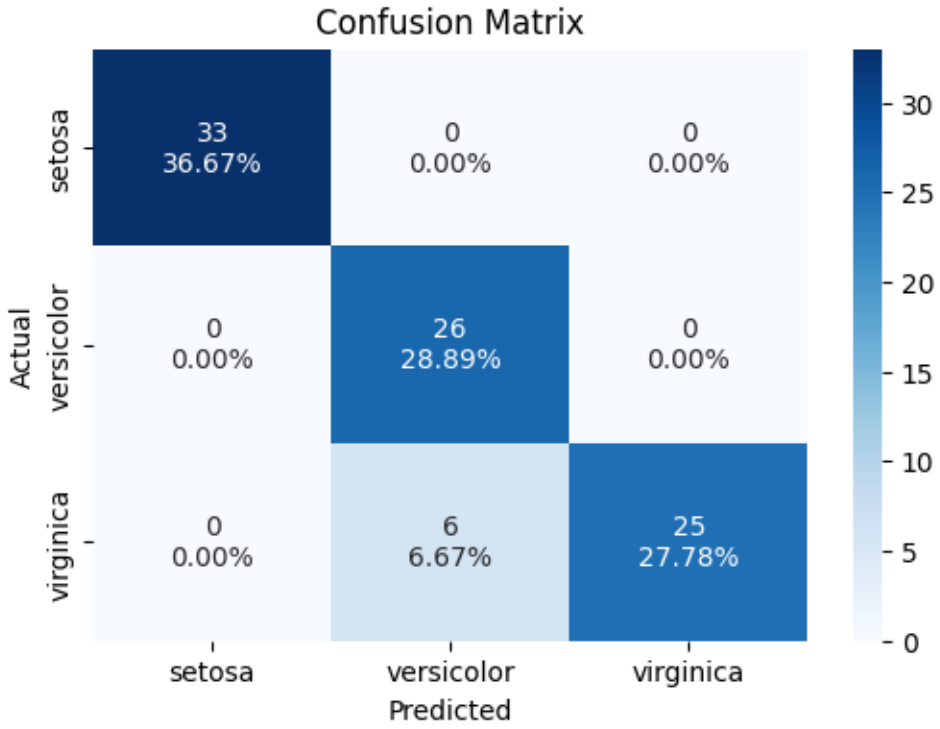



```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        26
           1       0.88      1.00      0.94        22
           2       1.00      0.89      0.94        27

    accuracy                           0.96        75
```

```
   macro avg       0.96      0.96      0.96        75
weighted avg       0.96      0.96      0.96        75
```

[24]:
```
randomForest(0.6, 100, 'entropy')
```

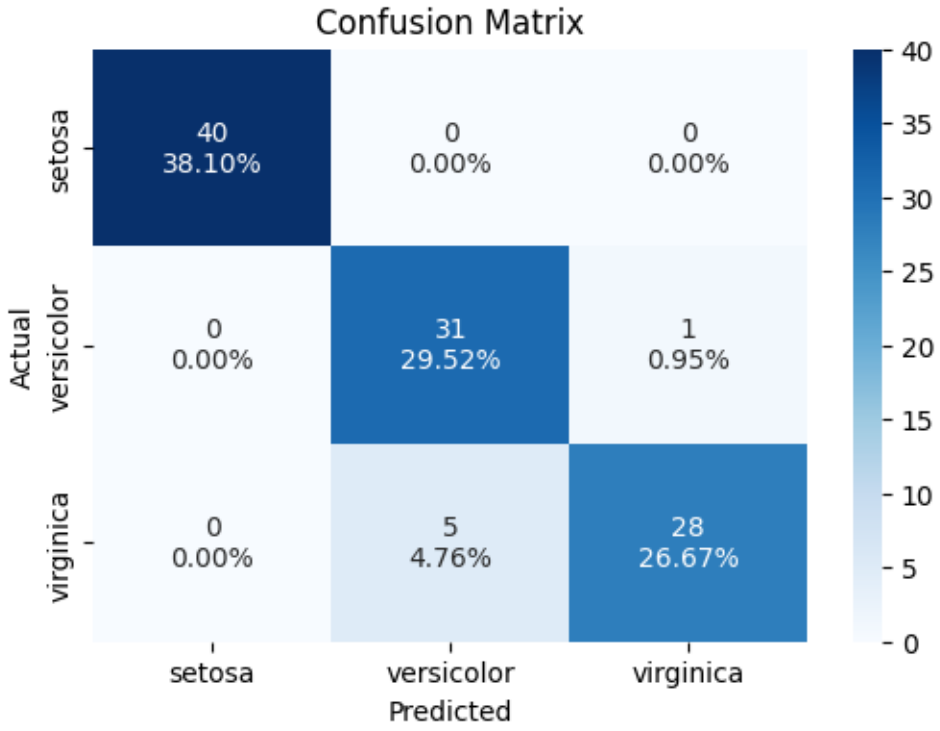
```
Confusion Matrix :
```




```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        33
           1       0.81      1.00      0.90        26
           2       1.00      0.81      0.89        31

    accuracy                           0.93        90
   macro avg       0.94      0.94      0.93        90
weighted avg       0.95      0.93      0.93        90
```

[25]:
```
randomForest(0.7, 120)
```

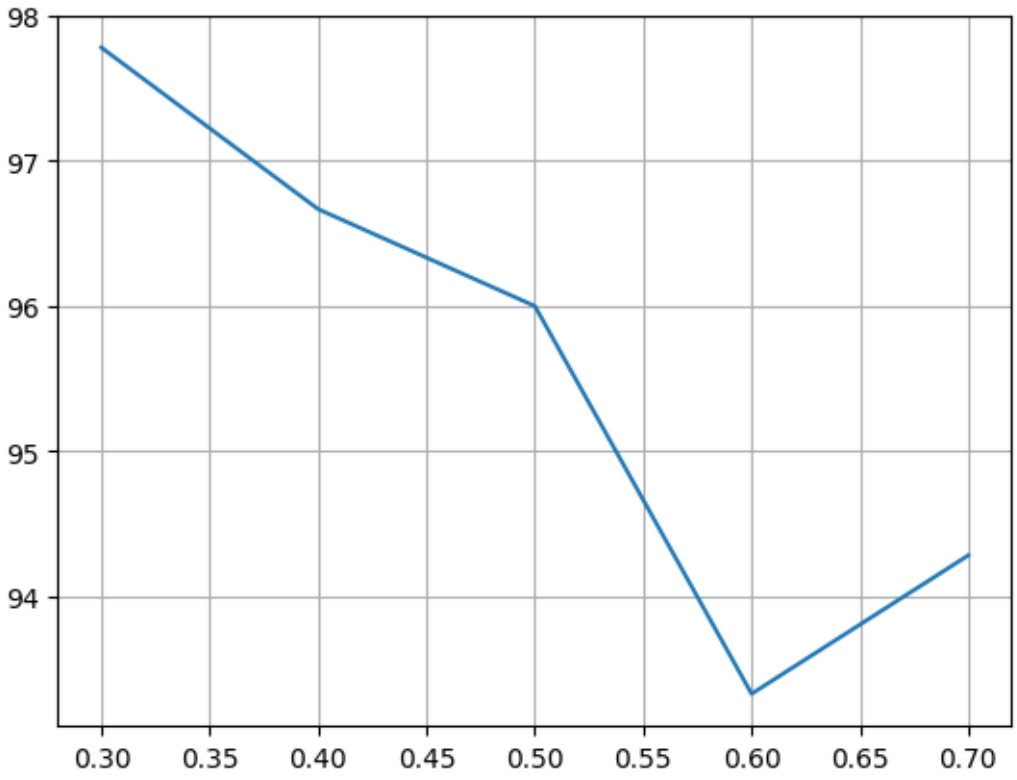
```
Confusion Matrix :
```




```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        40
           1       0.86      0.97      0.91        32
           2       0.97      0.85      0.90        33

    accuracy                           0.94       105
   macro avg       0.94      0.94      0.94       105
weighted avg       0.95      0.94      0.94       105
```

0.0.8 split vs accuracy graph

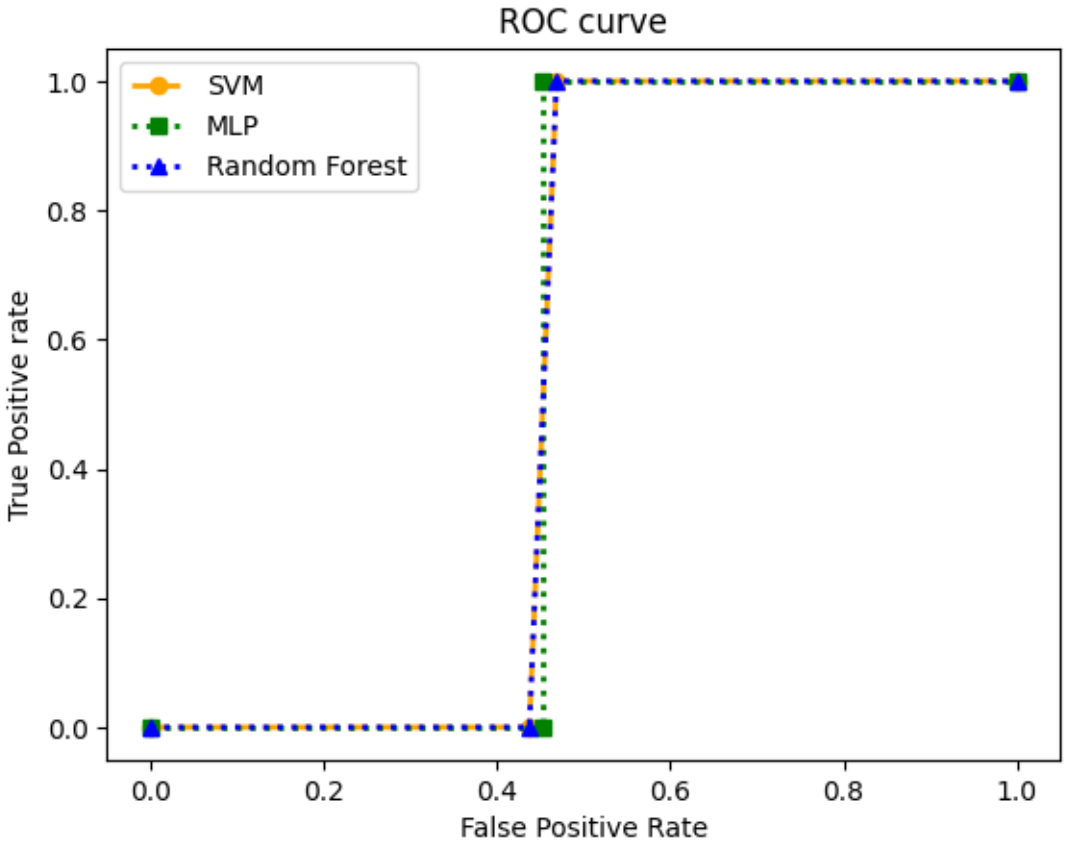
[26]:
```
x_points = [float(key) for key in dict_rfr]
y_points = [i*100 for i in dict_rfr.values()]
plt.plot(x_points, y_points)
plt.grid(True)
plt.show()
```



0.0.9 ROC curve and ROC_AUC score for all the classifier having maximum accuracy when train test split 70-30.

[27]:
```
from sklearn import metrics
def auc_roc():
    fpr1, tpr1, _1 = metrics.roc_curve(RocAucSvm['max']['y_test'],␣
 ↪RocAucSvm['max']['y_pred'], pos_label=1)
    fpr2, tpr2, _2 = metrics.roc_curve(RocAucMlp['max']['y_test'],␣
 ↪RocAucMlp['max']['y_pred'], pos_label=1)
    fpr3, tpr3, _3 = metrics.roc_curve(RocAucRfr['max']['y_test'],␣
 ↪RocAucRfr['max']['y_pred'], pos_label=1)
    plt.plot(fpr1, tpr1, linestyle='--', linewidth=2, color='orange',␣
 ↪marker='o', markersize=6, label='SVM')
    plt.plot(fpr2, tpr2, linestyle='dotted', linewidth=2, color='green',␣
 ↪marker='s', markersize=6, label='MLP')
    plt.plot(fpr3, tpr3, linestyle=':', linewidth=2, color='blue', marker='^',␣
 ↪markersize=6, label='Random Forest')
    plt.title('ROC curve')
    # x label
    plt.xlabel('False Positive Rate')
    # y label
```

```
    plt.ylabel('True Positive rate')

    plt.legend(loc='best')
    plt.savefig('ROC',dpi=300)
    plt.show()
auc_roc()
```



ionosphere-assignment-2

August 13, 2023

0.0.1 IONOSPHERE Dataset classification using SVM, MLP and Random Forest classifier

[ ]:

[31]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

[ ]:
```
url = 'https://raw.githubusercontent.com/Aqeel-0/phone.html/master/
 ↪ionosphere_data.csv'
df = pd.read_csv(url)
df.head()
#column_names = list(df.columns.values)
```

0.0.2 Pre Preprocessing

[ ]:
```
from sklearn.preprocessing import LabelEncoder
X = df.iloc[:, 0:-1]
y = df.iloc[: , -1]
le = LabelEncoder()
encoded = le.fit_transform(df['column_ai'])
df.drop("column_ai", axis=1, inplace=True)
df["column_ai"] = encoded
y = df["column_ai"]
dict_svm = {}
dict_mlp = {}
dict_rfr = {}
RocAucSvm = {}
RocAucMlp = {}
RocAucRfr = {}
y.info(), X.info()
y.value_counts()
```

0.0.3 Used for ploting confusion matrix

[34]:
```
def plot(y_test, y_pred):
  from sklearn.metrics import confusion_matrix
  import seaborn as sns

  print("Confusion Matrix : ")
  cf_matrix = confusion_matrix(y_test, y_pred)
  group_names = ['True Pos','False Pos','False Neg','True neg']
  group_counts = ["{0:0.0f}".format(value) for value in
                  cf_matrix.flatten()]
  group_percentages = ["{0:.2%}".format(value) for value in
                       cf_matrix.flatten()/np.sum(cf_matrix)]
  labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in
            zip(group_names,group_counts,group_percentages)]
  labels = np.asarray(labels).reshape(2,2)
  plt.figure(figsize=(6, 4))
  sns.heatmap(cf_matrix, annot=labels, fmt='', cmap='Blues',␣
 ↪xticklabels=['Good', 'Bad'], yticklabels=['Good', 'Bad'])
  plt.xlabel('Predicted')
  plt.ylabel('Actual')
  plt.title('Confusion Matrix')
  plt.show()
  print("************************************************")
```

[35]:
```
def reports(y_test, y_pred):
  from sklearn.metrics import classification_report, confusion_matrix
  plot(y_test, y_pred)
  print("************************************************")
  print("Classification Evaluation : ")
  print(classification_report(y_test, y_pred, zero_division = 0))
```

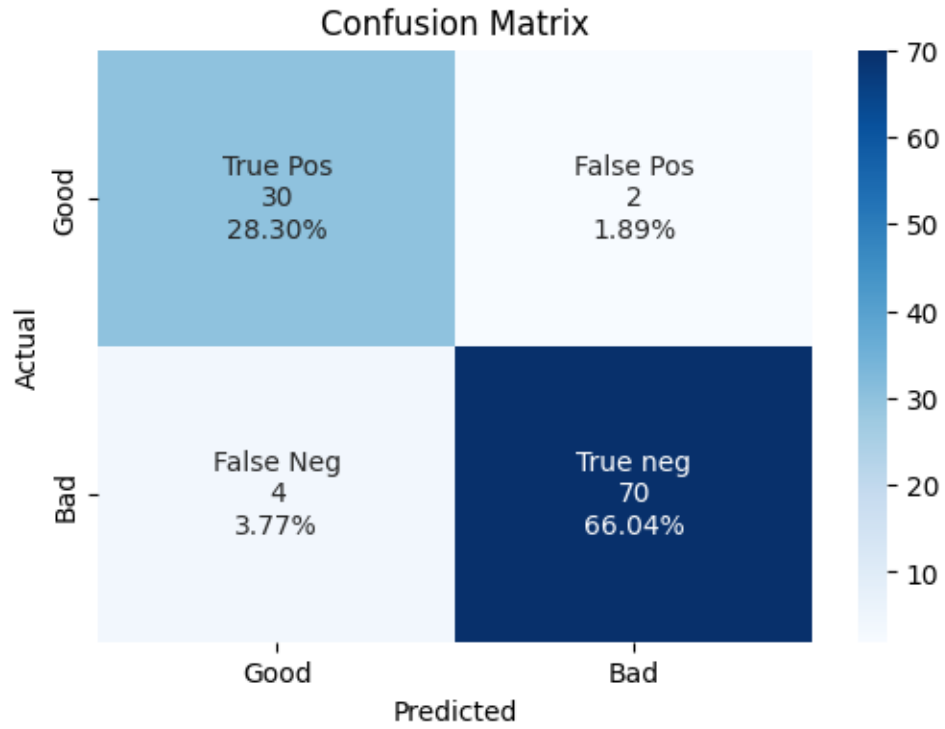
0.0.4 SVM CLASSIFIER

[36]:
```
def SVMClassifier(split, kernalValue = 'rbf', degreeValue = 3, gammaValue =␣
 ↪'scale', maxIter = -1):
  from sklearn.model_selection import train_test_split
  from sklearn.svm import SVC
  from sklearn.metrics import accuracy_score
  from sklearn.preprocessing import StandardScaler
  scaler = StandardScaler()
  X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = split,␣
 ↪random_state=44)
  scaler.fit_transform(X_train)
  scaler.transform(X_test)
  classifier = SVC(kernel = kernalValue, degree = degreeValue, gamma =␣
 ↪gammaValue, max_iter = maxIter)
```

```
  classifier.fit(X_train, y_train)
  y_pred = classifier.predict(X_test)
  accuracy = accuracy_score(y_test, y_pred)

  if str(split) in dict_svm:
    dict_svm[str(split)] = max(accuracy, dict_svm[str(split)])
    if str(split) == '0.3' and accuracy > dict_svm[str(split)]:
      RocAucSvm['max'] = {'y_test': y_test, 'y_pred': y_pred}
  else:
    dict_svm[str(split)] = accuracy
    if str(split) == '0.3':
      RocAucSvm['max'] = {'y_test': y_test, 'y_pred': y_pred}
  reports(y_test, y_pred)
```

[37]:
```
#Train - Test split 70-30
SVMClassifier(0.3, 'rbf', 3, 0.42)
SVMClassifier(0.3, 'linear', 3, 0.1)
SVMClassifier(0.3, 'poly', 5, )
SVMClassifier(0.3, 'sigmoid',3, 0.01)
```

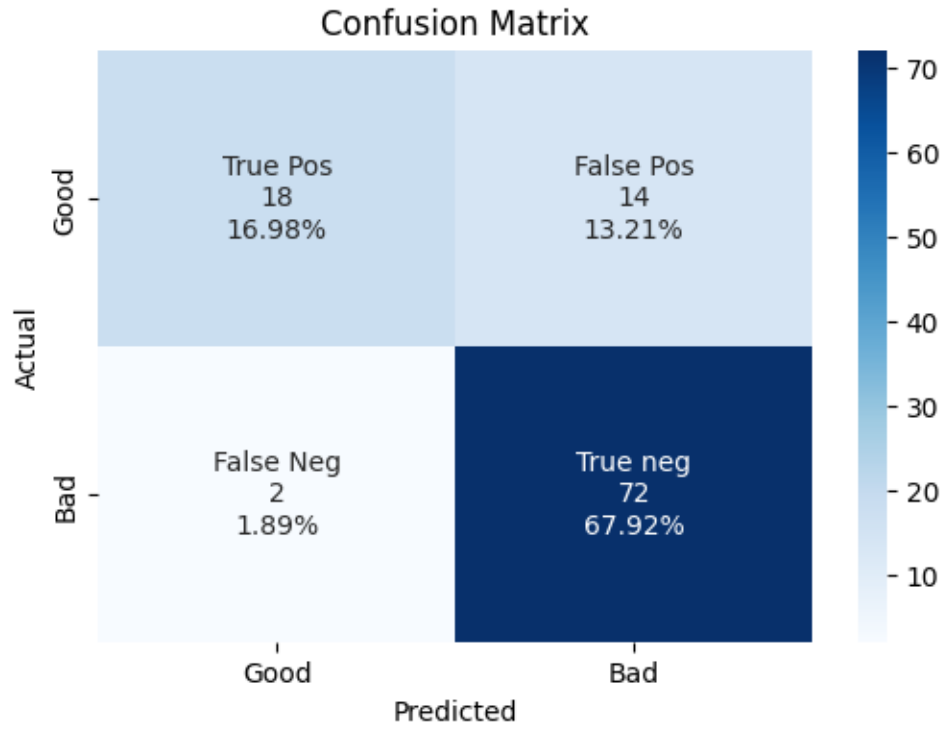
Confusion Matrix :



****
****

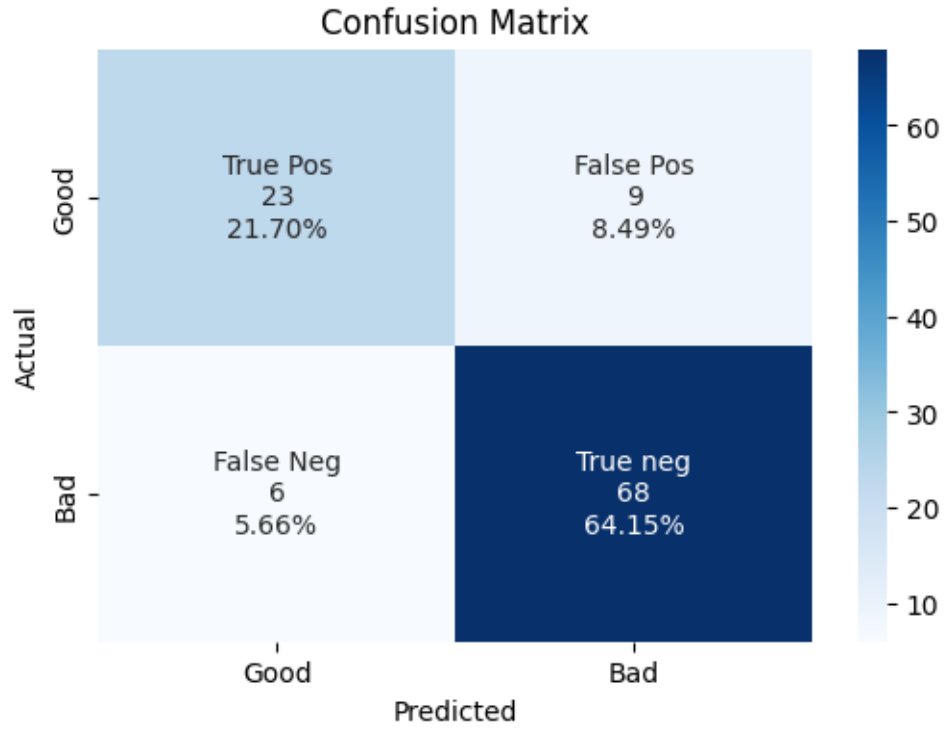
```
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.88      0.94      0.91        32
           1       0.97      0.95      0.96        74

    accuracy                           0.94       106
   macro avg       0.93      0.94      0.93       106
weighted avg       0.95      0.94      0.94       106
```

Confusion Matrix :




```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.90      0.56      0.69        32
           1       0.84      0.97      0.90        74

    accuracy                           0.85       106
   macro avg       0.87      0.77      0.80       106
weighted avg       0.86      0.85      0.84       106
```

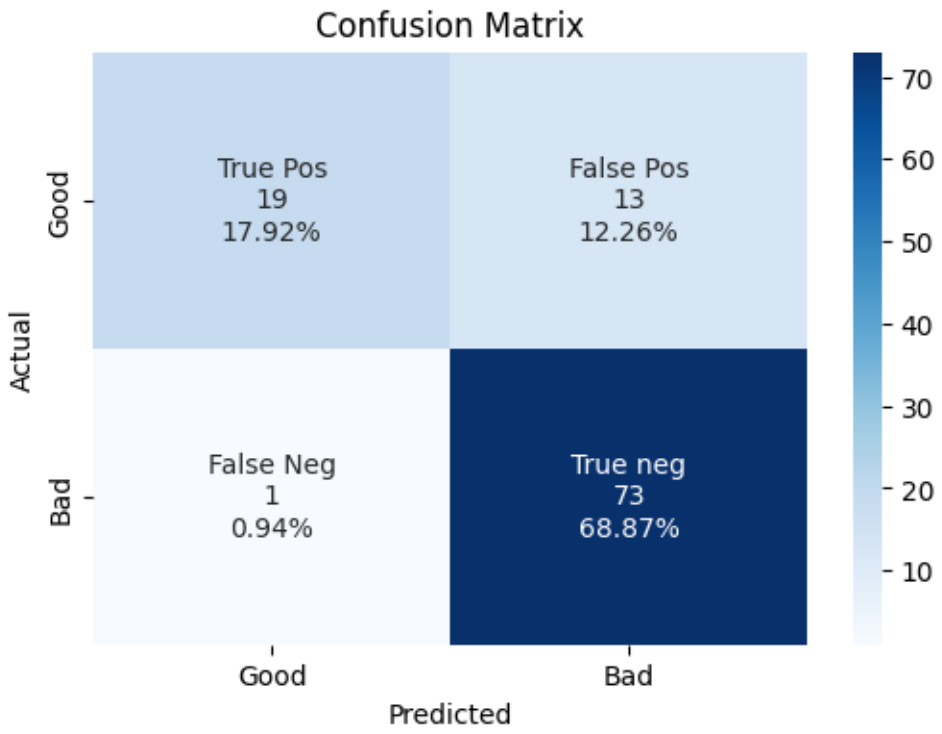
Confusion Matrix :



****
****
Classification Evaluation :

	precision	recall	f1-score	support
0	0.79	0.72	0.75	32
1	0.88	0.92	0.90	74
accuracy			0.86	106
macro avg	0.84	0.82	0.83	106
weighted avg	0.86	0.86	0.86	106

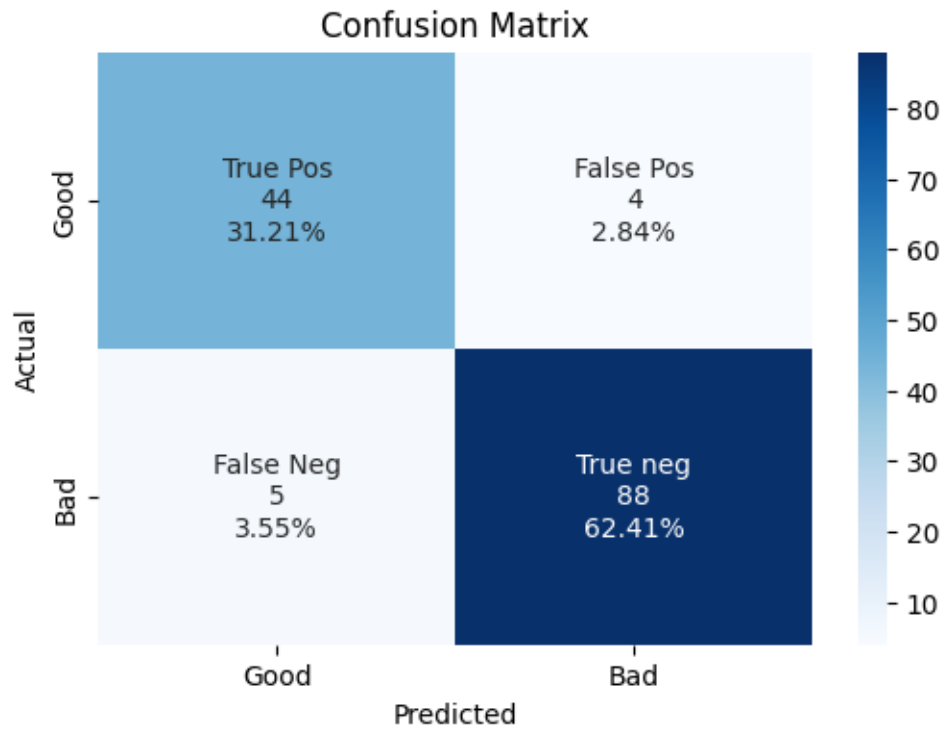
Confusion Matrix :



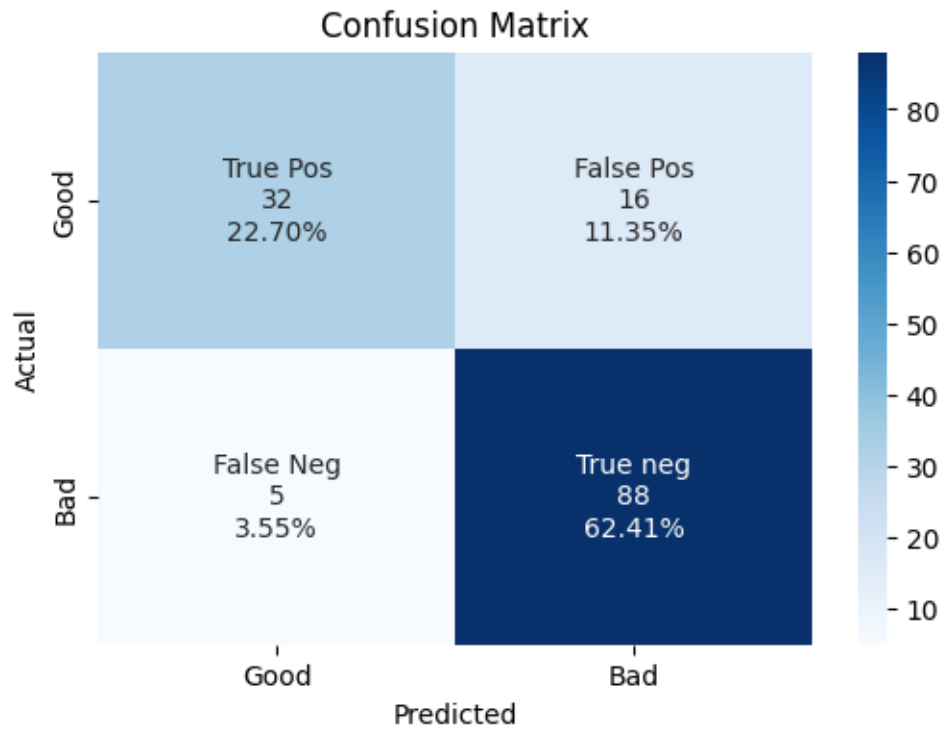

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.95      0.59      0.73        32
           1       0.85      0.99      0.91        74

    accuracy                           0.87       106
   macro avg       0.90      0.79      0.82       106
weighted avg       0.88      0.87      0.86       106
```

[38]:
```
#Train - Test split 60-40
SVMClassifier(0.4, 'rbf', 3, 0.31)
SVMClassifier(0.4, 'linear', 3, 0.01)
SVMClassifier(0.4, 'poly', 5, )
SVMClassifier(0.4, 'sigmoid', 3, 0.01)
```

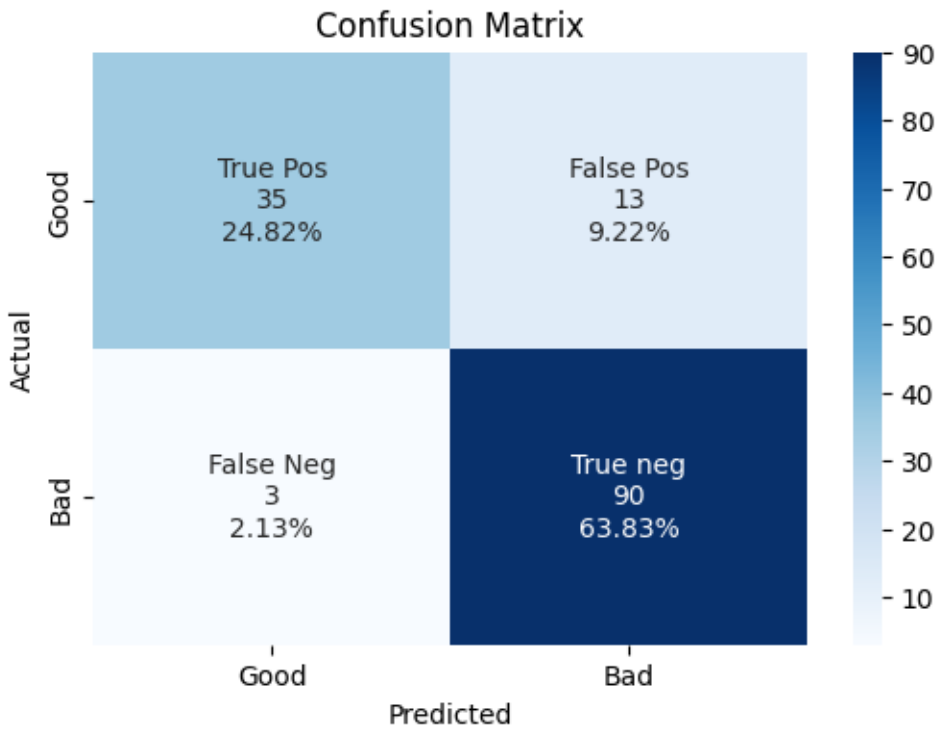
```
Confusion Matrix :
```



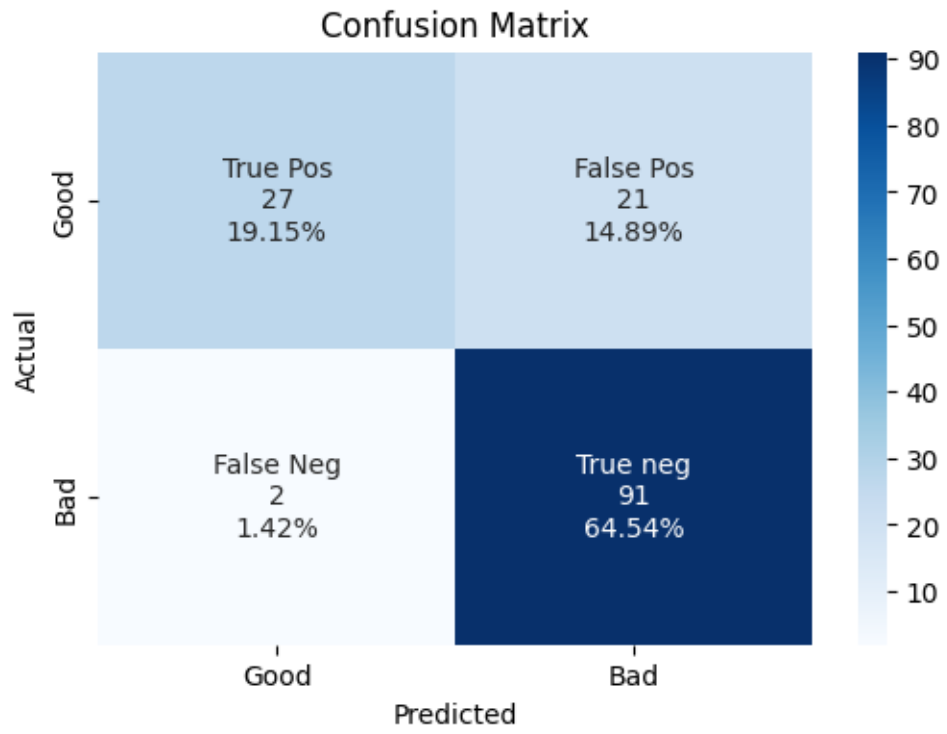

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.90      0.92      0.91        48
           1       0.96      0.95      0.95        93

    accuracy                           0.94       141
   macro avg       0.93      0.93      0.93       141
weighted avg       0.94      0.94      0.94       141

Confusion Matrix :
```




```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.86      0.67      0.75        48
           1       0.85      0.95      0.89        93

    accuracy                           0.85       141
   macro avg       0.86      0.81      0.82       141
weighted avg       0.85      0.85      0.85       141

Confusion Matrix :
```



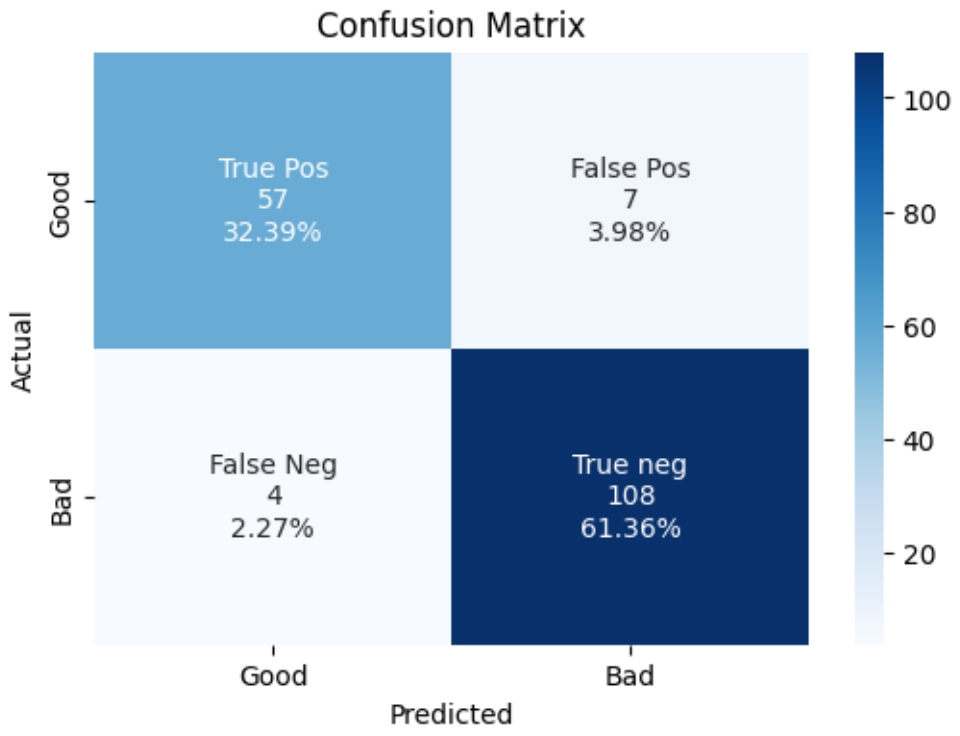

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.92      0.73      0.81        48
           1       0.87      0.97      0.92        93

    accuracy                           0.89       141
   macro avg       0.90      0.85      0.87       141
weighted avg       0.89      0.89      0.88       141

Confusion Matrix :
```



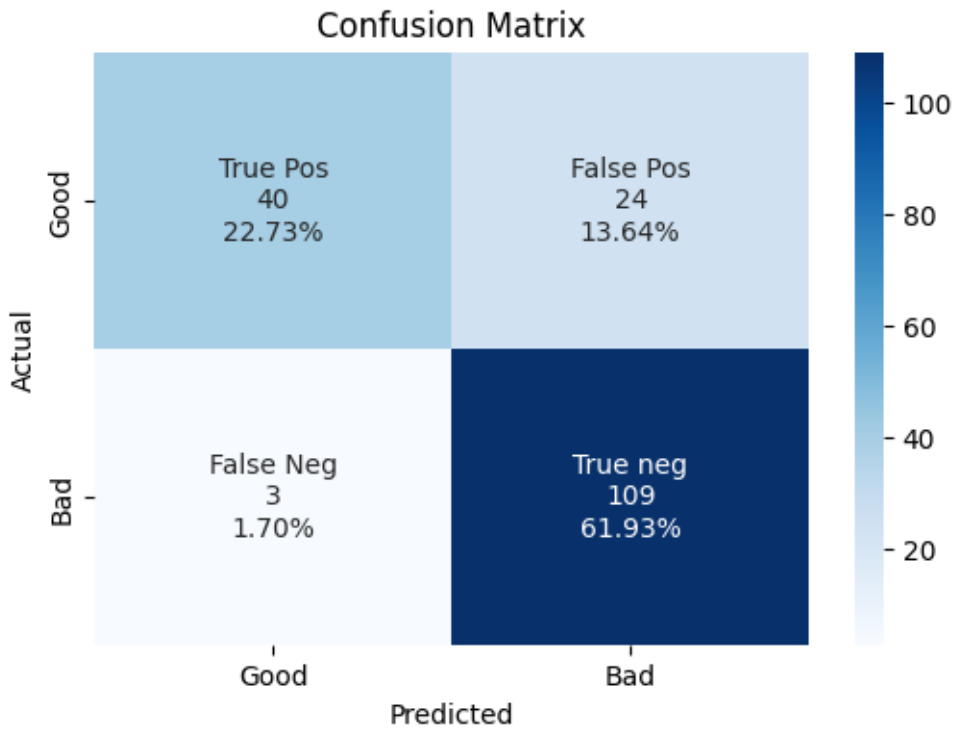

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.93      0.56      0.70        48
           1       0.81      0.98      0.89        93

    accuracy                           0.84       141
   macro avg       0.87      0.77      0.79       141
weighted avg       0.85      0.84      0.82       141
```

[39]:
```
#Train - Test split 50-50
SVMClassifier(0.5, 'rbf', 3, 0.18)
SVMClassifier(0.5, 'linear', 3, )
SVMClassifier(0.5, 'poly', 4, )
SVMClassifier(0.5, 'sigmoid', 3, 0.09 ) #wrost performance
```

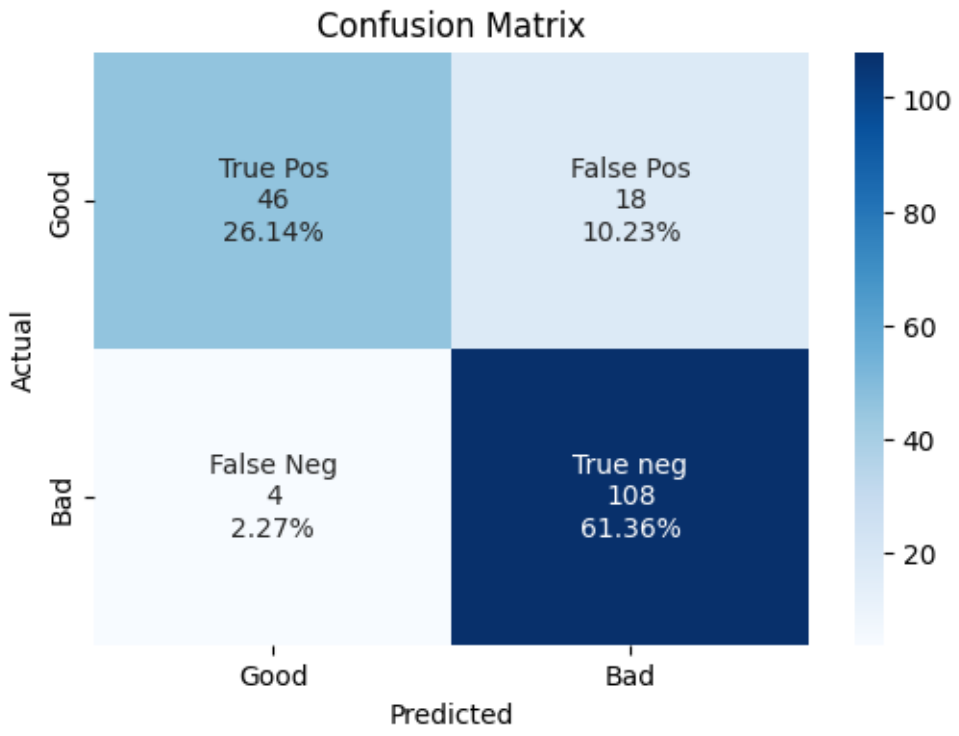
```
Confusion Matrix :
```



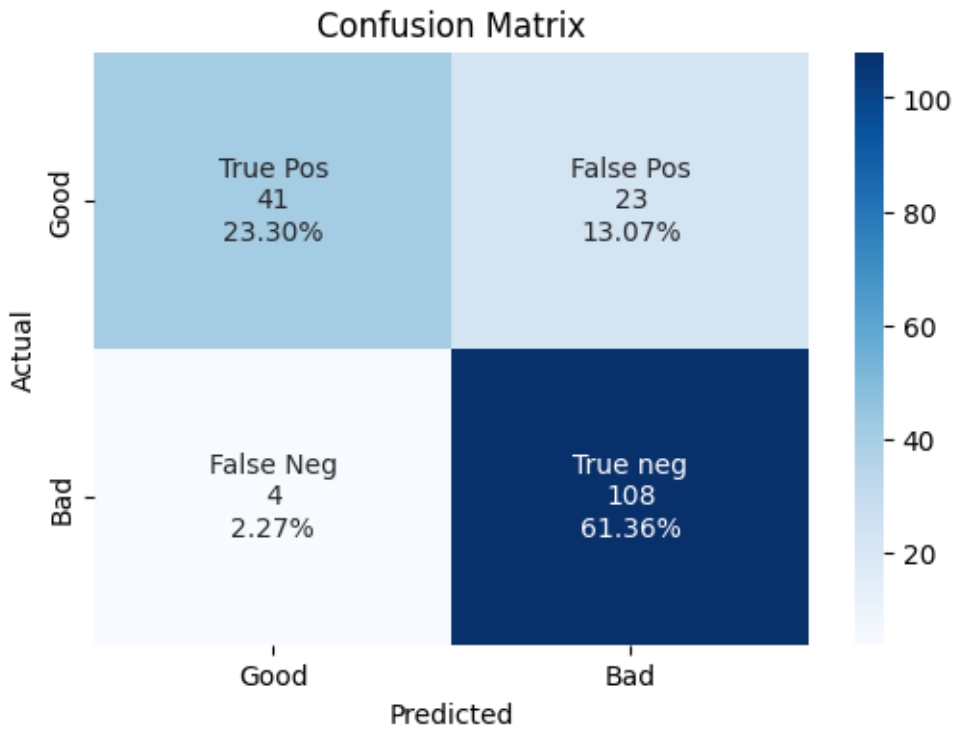

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.93      0.89      0.91        64
           1       0.94      0.96      0.95       112

    accuracy                           0.94       176
   macro avg       0.94      0.93      0.93       176
weighted avg       0.94      0.94      0.94       176

Confusion Matrix :
```




```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.93      0.62      0.75        64
           1       0.82      0.97      0.89       112

    accuracy                           0.85       176
   macro avg       0.87      0.80      0.82       176
weighted avg       0.86      0.85      0.84       176

Confusion Matrix :
```



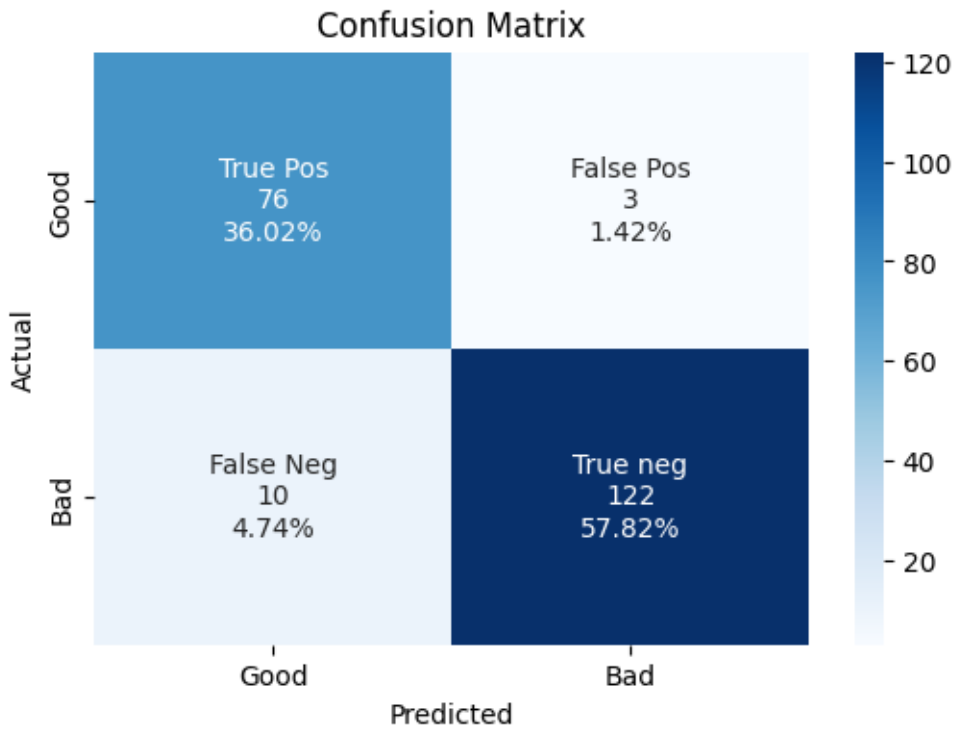

```
*********************************************
*********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.92      0.72      0.81        64
           1       0.86      0.96      0.91       112

    accuracy                           0.88       176
   macro avg       0.89      0.84      0.86       176
weighted avg       0.88      0.88      0.87       176

Confusion Matrix :
```



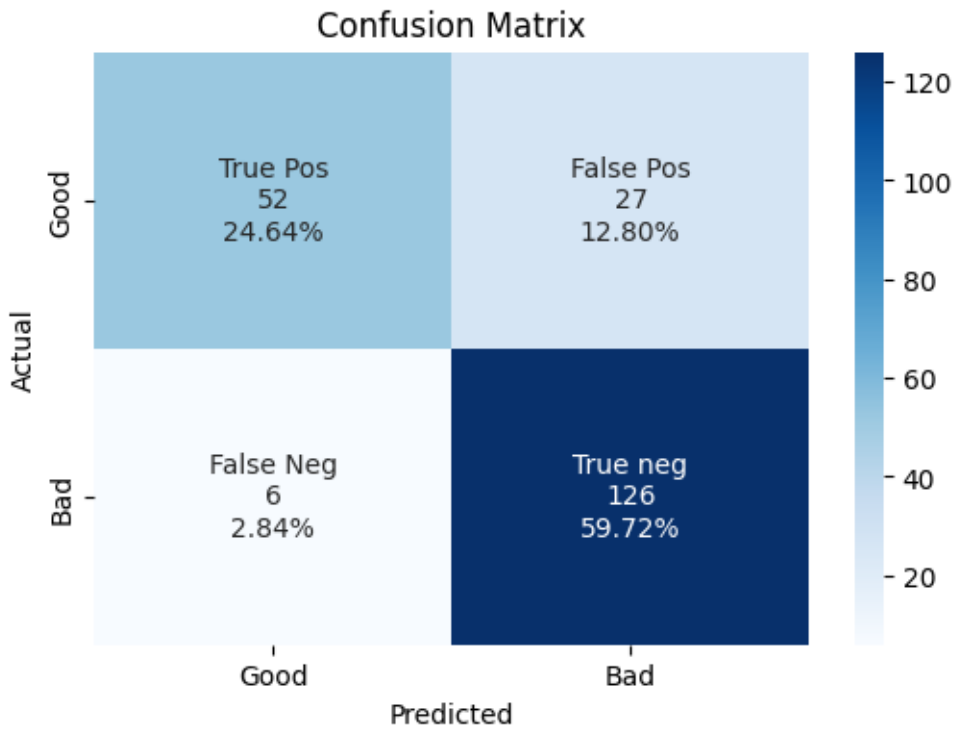

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.91      0.64      0.75        64
           1       0.82      0.96      0.89       112

    accuracy                           0.85       176
   macro avg       0.87      0.80      0.82       176
weighted avg       0.86      0.85      0.84       176
```

[40]:
```
#Train - Test split 40-60
SVMClassifier(0.6, 'rbf', 3, 0.51)
SVMClassifier(0.6, 'linear', 3, )
SVMClassifier(0.6, 'poly', 2, 0.14)
SVMClassifier(0.6, 'sigmoid', 3,) #wrost performance
```

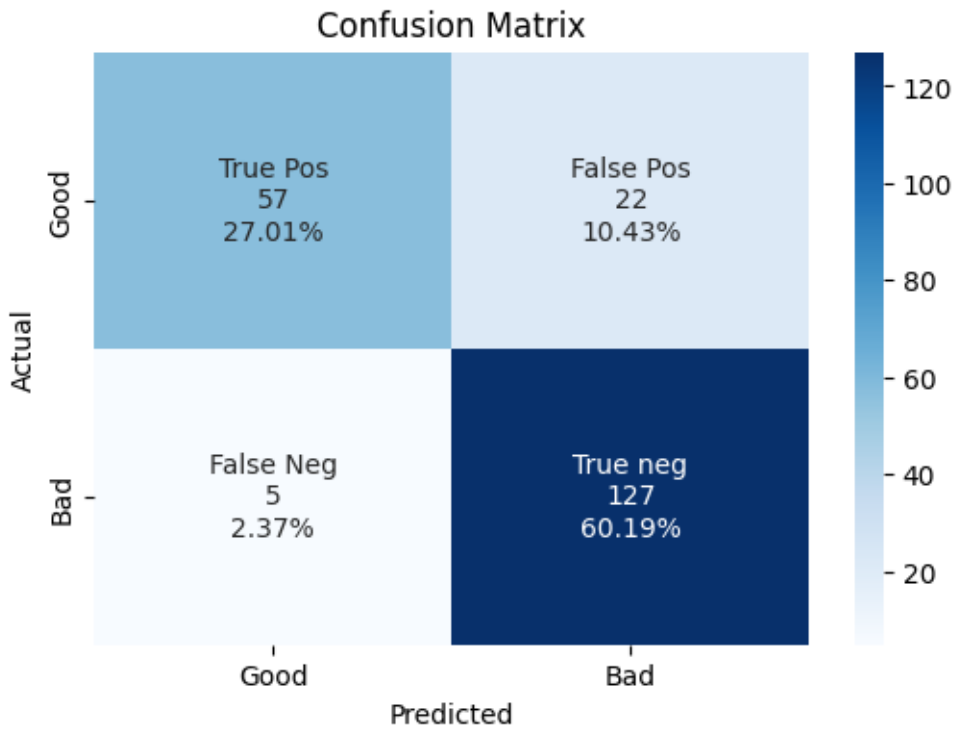
```
Confusion Matrix :
```



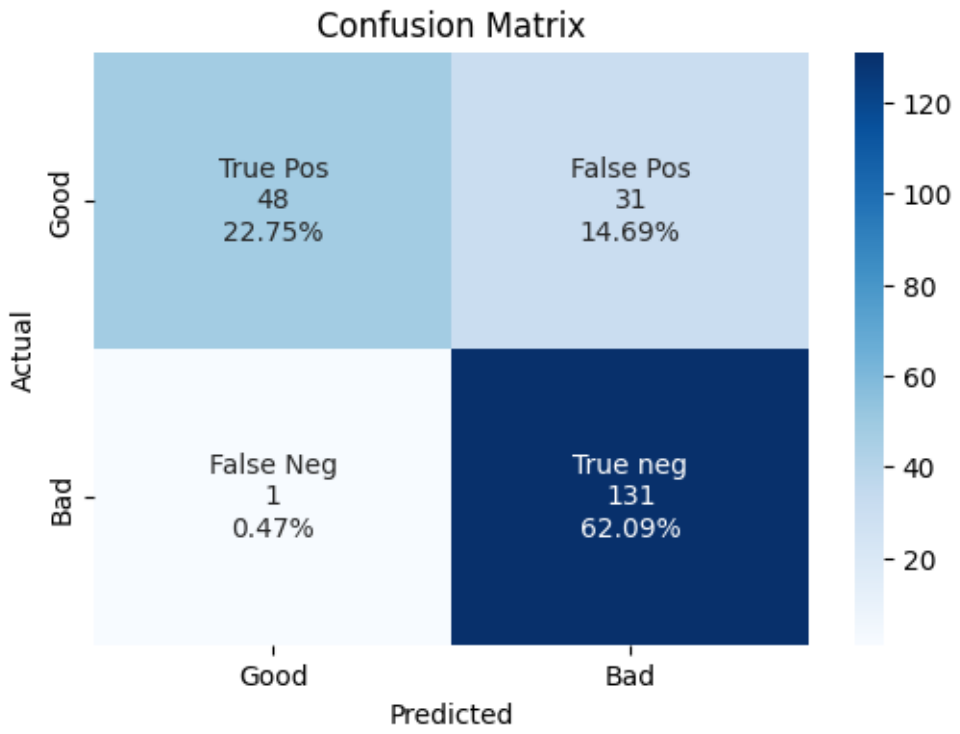

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.88      0.96      0.92        79
           1       0.98      0.92      0.95       132

    accuracy                           0.94       211
   macro avg       0.93      0.94      0.94       211
weighted avg       0.94      0.94      0.94       211

Confusion Matrix :
```




```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.90      0.66      0.76        79
           1       0.82      0.95      0.88       132

    accuracy                           0.84       211
   macro avg       0.86      0.81      0.82       211
weighted avg       0.85      0.84      0.84       211

Confusion Matrix :
```



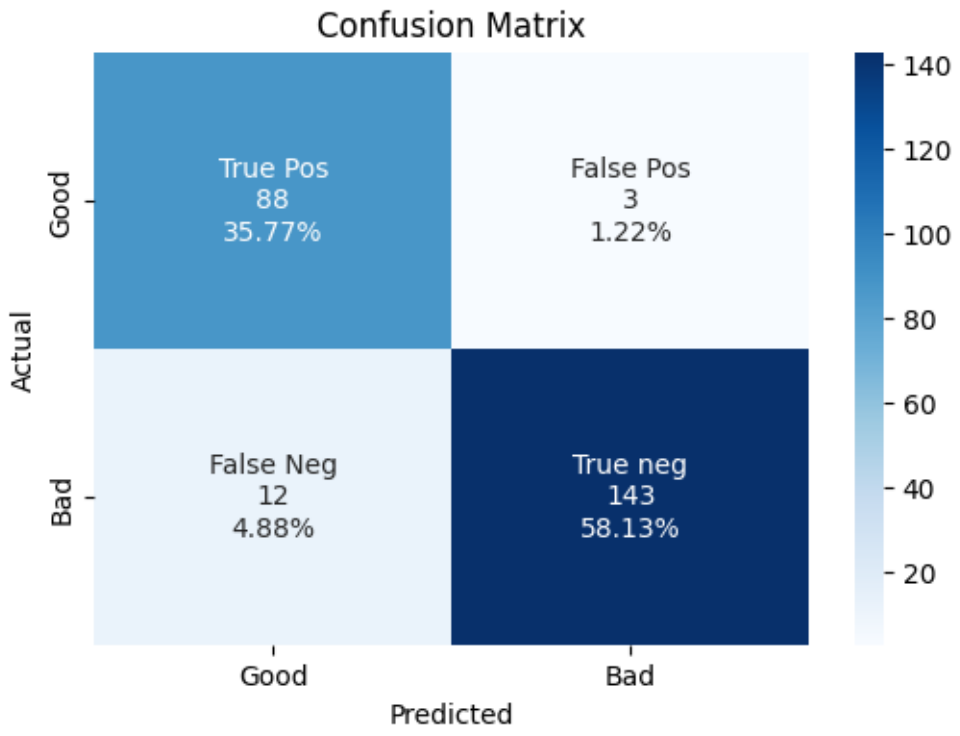

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.92      0.72      0.81        79
           1       0.85      0.96      0.90       132

    accuracy                           0.87       211
   macro avg       0.89      0.84      0.86       211
weighted avg       0.88      0.87      0.87       211

Confusion Matrix :
```



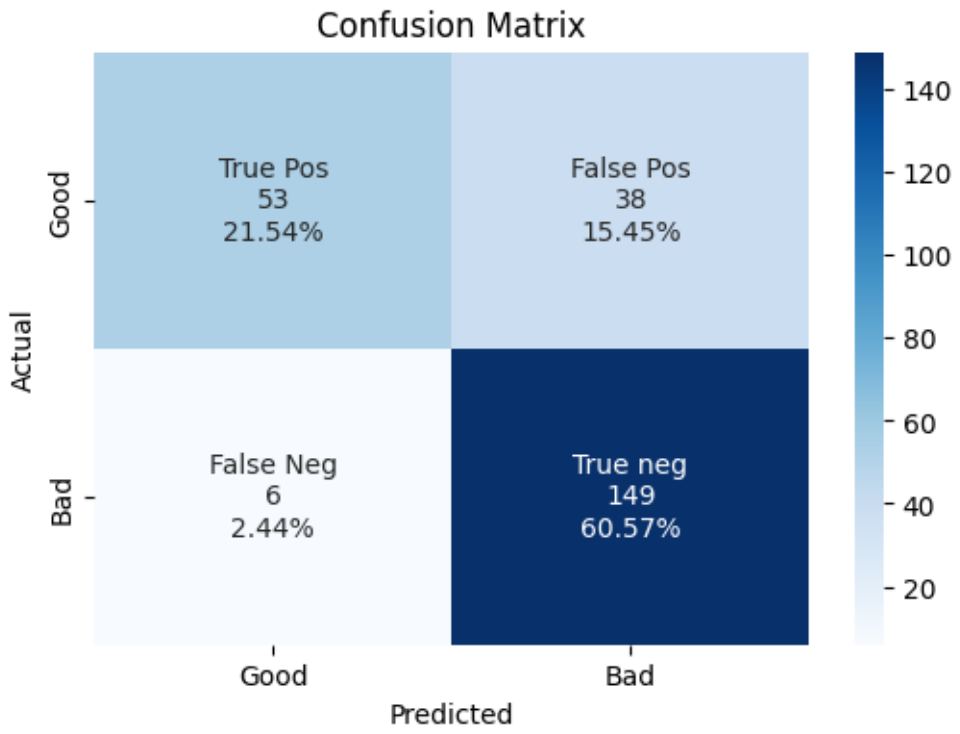

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.98      0.61      0.75        79
           1       0.81      0.99      0.89       132

    accuracy                           0.85       211
   macro avg       0.89      0.80      0.82       211
weighted avg       0.87      0.85      0.84       211
```

[41]:
```
#Train - Test split 30-70
SVMClassifier(0.7, 'rbf', 3, 0.64)
SVMClassifier(0.7, 'linear')
SVMClassifier(0.7, 'poly', 2,)
SVMClassifier(0.7, 'sigmoid' ) #wrost performance
```

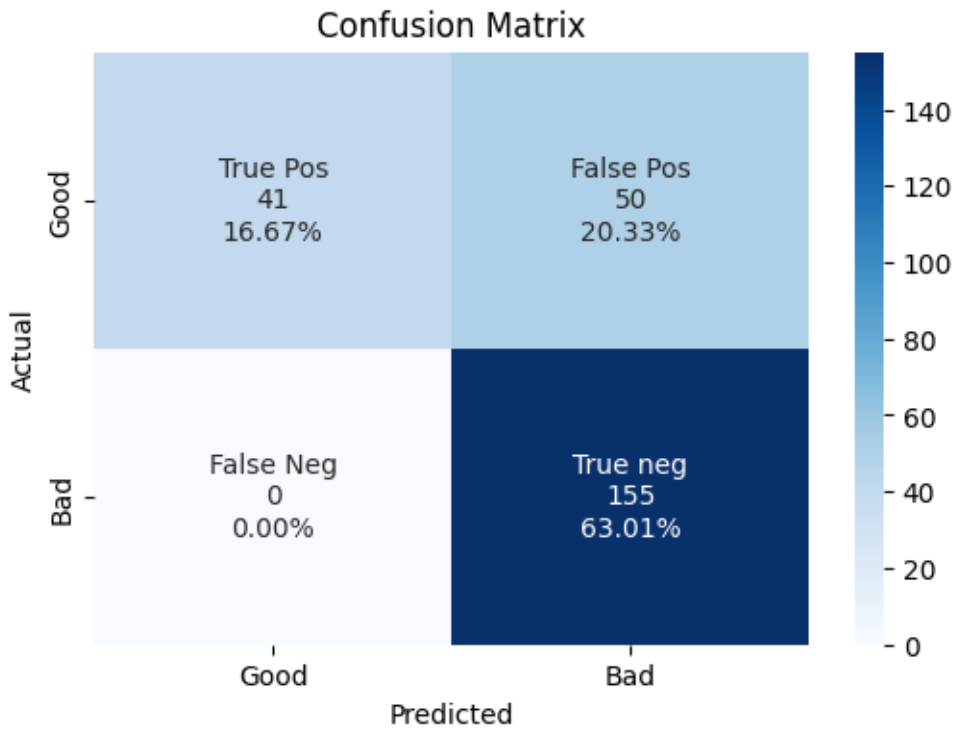
```
Confusion Matrix :
```



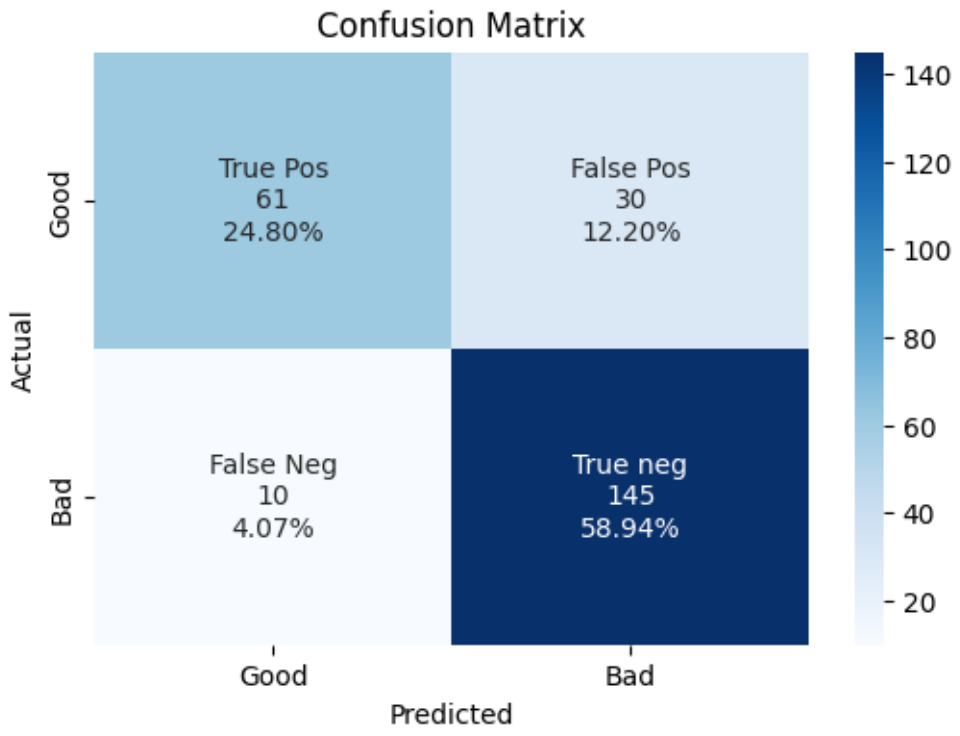

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.88      0.97      0.92        91
           1       0.98      0.92      0.95       155

    accuracy                           0.94       246
   macro avg       0.93      0.94      0.94       246
weighted avg       0.94      0.94      0.94       246

Confusion Matrix :
```




```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.90      0.58      0.71        91
           1       0.80      0.96      0.87       155

    accuracy                           0.82       246
   macro avg       0.85      0.77      0.79       246
weighted avg       0.83      0.82      0.81       246

Confusion Matrix :
```



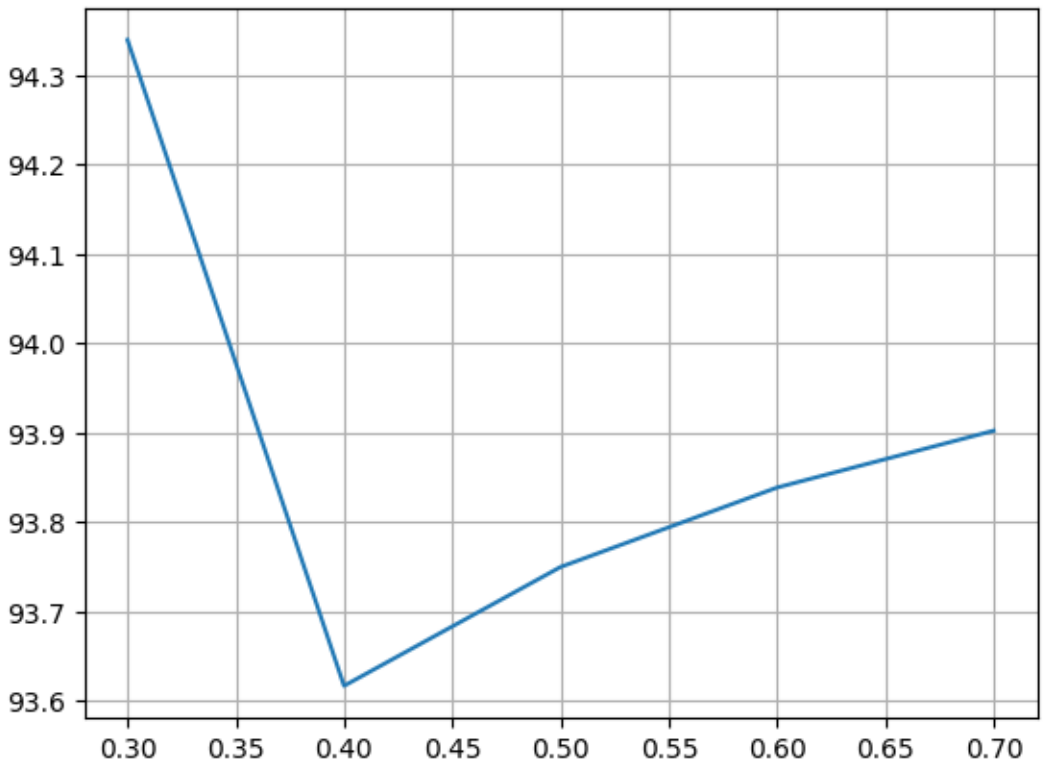

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       1.00      0.45      0.62        91
           1       0.76      1.00      0.86       155

    accuracy                           0.80       246
   macro avg       0.88      0.73      0.74       246
weighted avg       0.85      0.80      0.77       246

Confusion Matrix :
```




```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.86      0.67      0.75        91
           1       0.83      0.94      0.88       155

    accuracy                           0.84       246
   macro avg       0.84      0.80      0.82       246
weighted avg       0.84      0.84      0.83       246
```

0.0.5 split vs accuracy graph

[42]:
```
x_points = [float(key) for key in dict_svm]
y_points = [i*100 for i in dict_svm.values()]
plt.plot(x_points, y_points)
plt.grid(True)
plt.show()
```



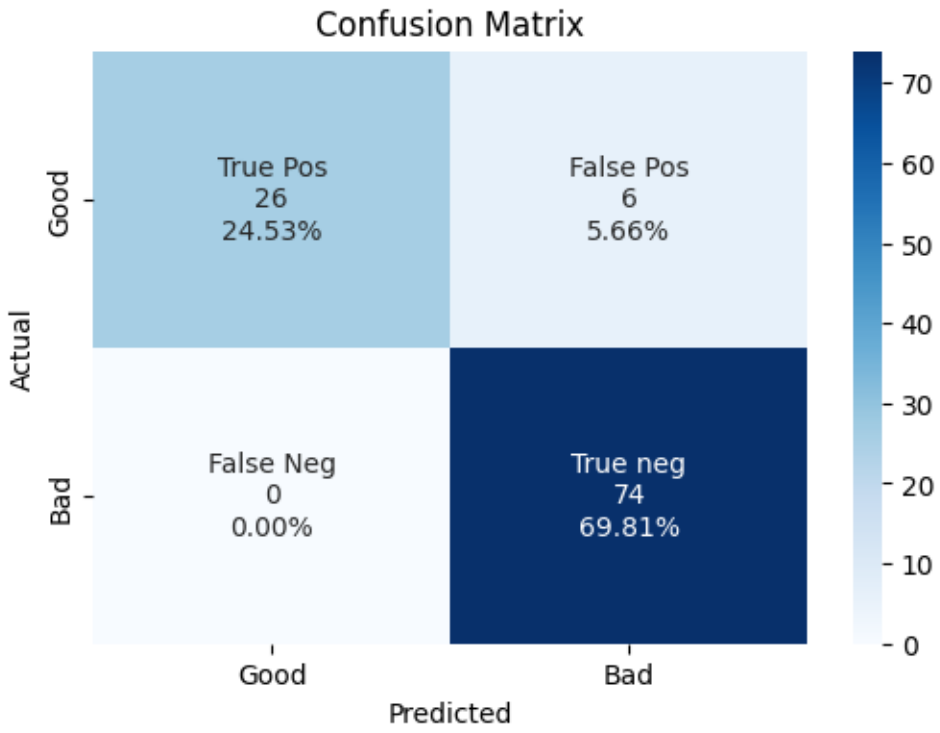
0.0.6 MLP Classifier

[43]:
```
def MLPClassifier(split, hiddenLayerSize = [100, ], activationValue = 'relu',␣
↪solverValue = 'adam'):
 from sklearn.model_selection import train_test_split
 from sklearn.neural_network import MLPClassifier
 from sklearn.metrics import accuracy_score
 from sklearn.preprocessing import StandardScaler
 scaler = StandardScaler()
 X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = split,␣
↪random_state=44)
 scaler.fit_transform(X_train)
 scaler.transform(X_test)
 classifier = MLPClassifier(hidden_layer_sizes = hiddenLayerSize, activation =␣
↪activationValue, solver = solverValue, random_state = 1)
 classifier.fit(X_train, y_train)
 y_pred = classifier.predict(X_test)
 accuracy = accuracy_score(y_test, y_pred)
 if(str(split) in dict_mlp):
   dict_mlp[str(split)] = max(accuracy, dict_mlp[str(split)])
   if(str(split) == '0.3' and accuracy > dict_svm[str(split)]):
     RocAucMlp['max'] = {'y_test': y_test, 'y_pred': y_pred}
```

```
  else:
    dict_mlp[str(split)] = accuracy
    RocAucMlp['max'] = {'y_test': y_test, 'y_pred': y_pred}

  reports(y_test, y_pred)
```

[44]:
```
#Train - Test split 70-30
MLPClassifier(0.3, [80, 20])
```

```
/home/aqeel/.local/lib/python3.10/site-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and
the optimization hasn't converged yet.
  warnings.warn(

Confusion Matrix :
```



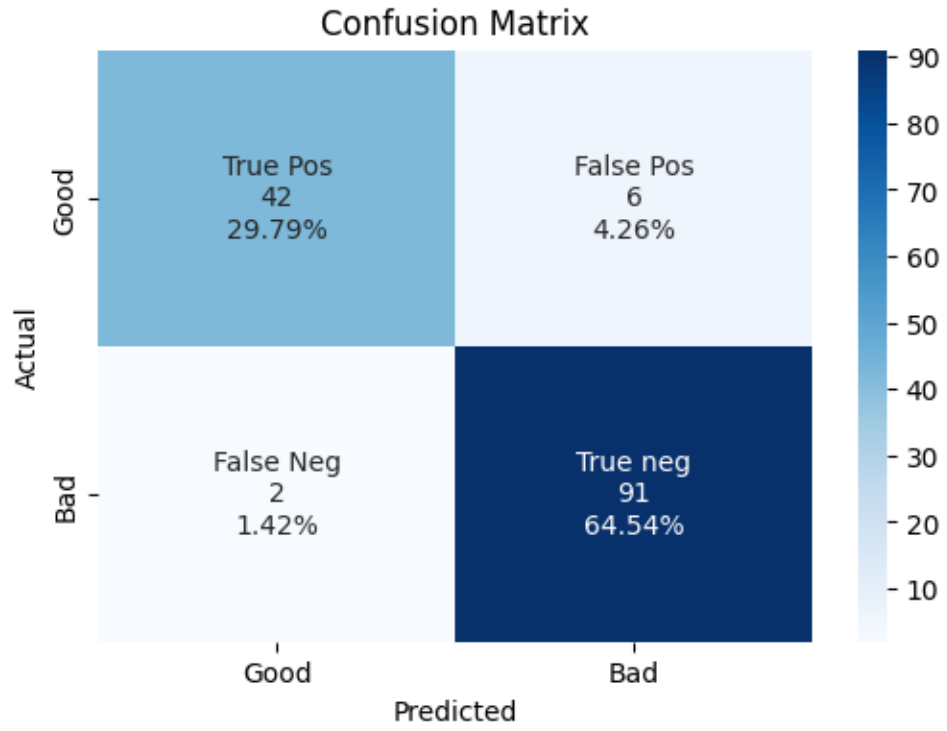

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       1.00      0.81      0.90        32
           1       0.93      1.00      0.96        74
```

```
    accuracy                           0.94       106
   macro avg       0.96      0.91      0.93       106
weighted avg       0.95      0.94      0.94       106
```

[45]:
```
#Train - Test split 60-40
MLPClassifier(0.4, [80, 15])
```

```
/home/aqeel/.local/lib/python3.10/site-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and
the optimization hasn't converged yet.
  warnings.warn(
```

```
Confusion Matrix :
```



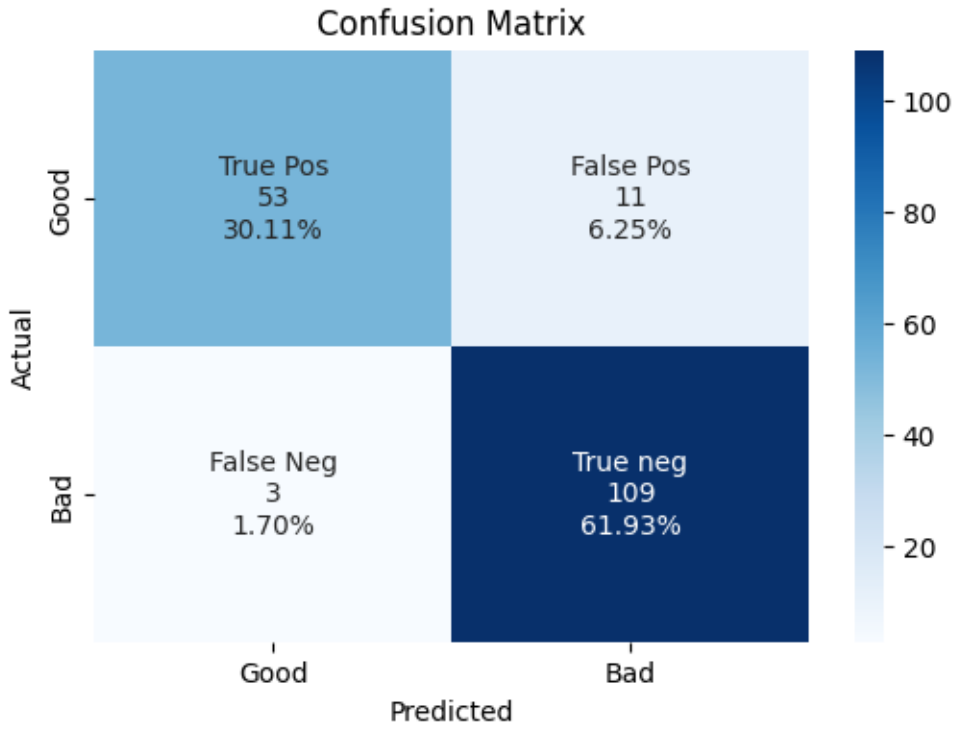

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.95      0.88      0.91        48
           1       0.94      0.98      0.96        93
```

```
    accuracy                           0.94       141
   macro avg       0.95      0.93      0.94       141
weighted avg       0.94      0.94      0.94       141
```

[46]:
```
#Train - Test split 50-50
MLPClassifier(0.5, [80, 15])
```

```
/home/aqeel/.local/lib/python3.10/site-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and
the optimization hasn't converged yet.
  warnings.warn(
```

```
Confusion Matrix :
```



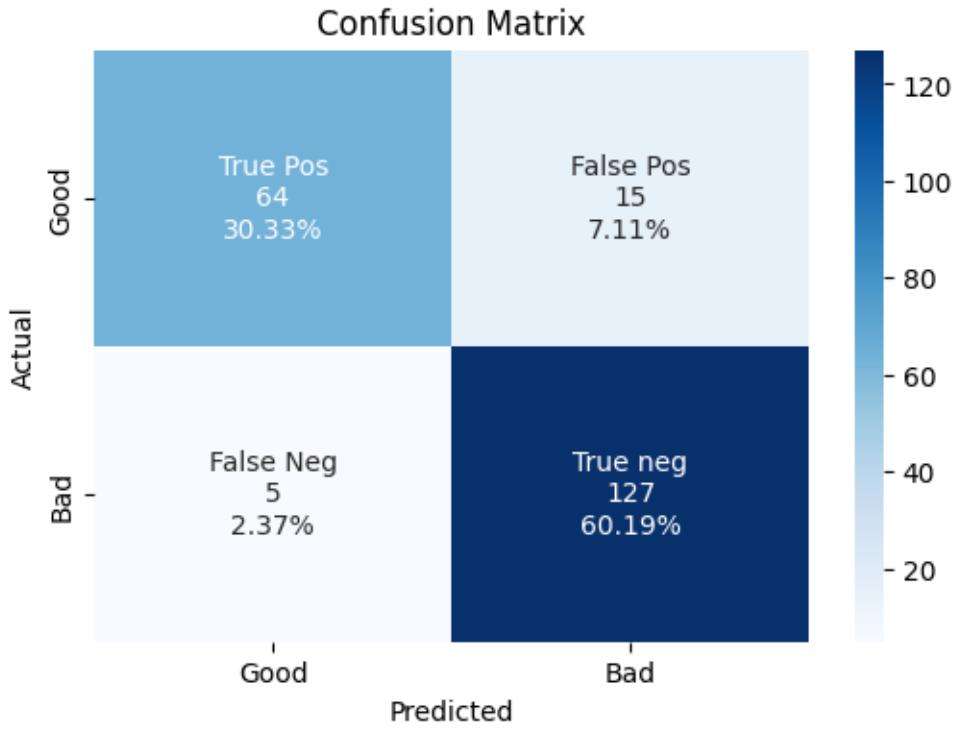

```
*******************************************
*******************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.95      0.83      0.88        64
           1       0.91      0.97      0.94       112
```

```
    accuracy                           0.92       176
   macro avg       0.93      0.90      0.91       176
weighted avg       0.92      0.92      0.92       176
```

[47]:
```
#Train - Test split 40-60
MLPClassifier(0.6, [80, 46])
```

```
/home/aqeel/.local/lib/python3.10/site-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and
the optimization hasn't converged yet.
  warnings.warn(
```

```
Confusion Matrix :
```

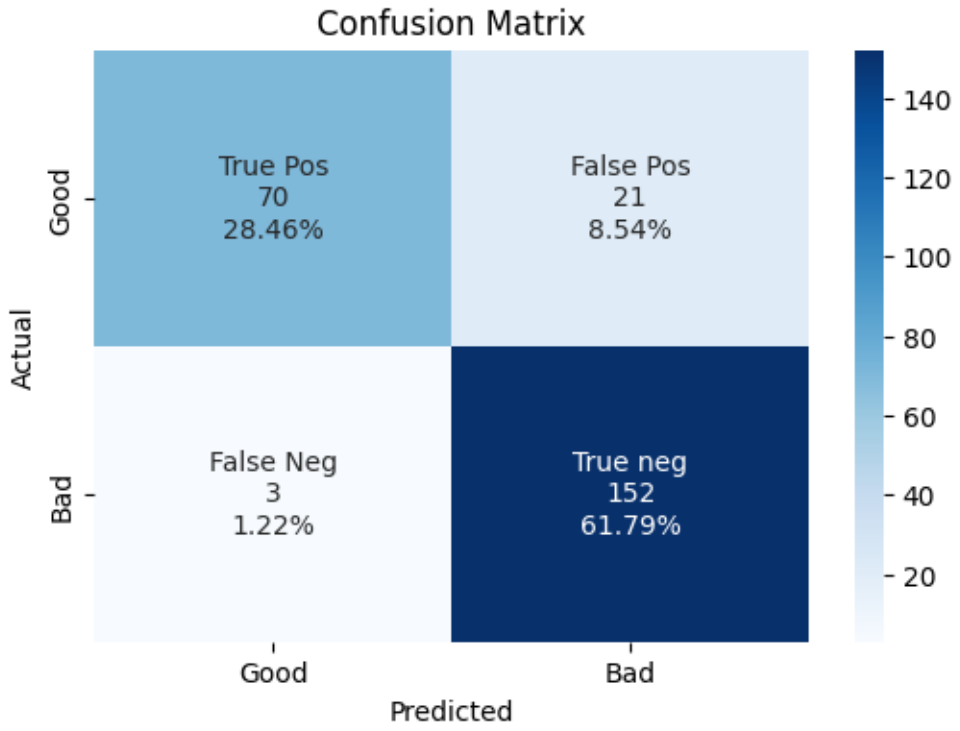



```
*******************************************
*******************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.93      0.81      0.86        79
           1       0.89      0.96      0.93       132
```

```
    accuracy                           0.91       211
   macro avg       0.91      0.89      0.90       211
weighted avg       0.91      0.91      0.90       211
```

[48]:
```
#Train - Test split 30-70
MLPClassifier(0.7, [50, 22])
```

```
Confusion Matrix :

/home/aqeel/.local/lib/python3.10/site-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and
the optimization hasn't converged yet.
  warnings.warn(
```

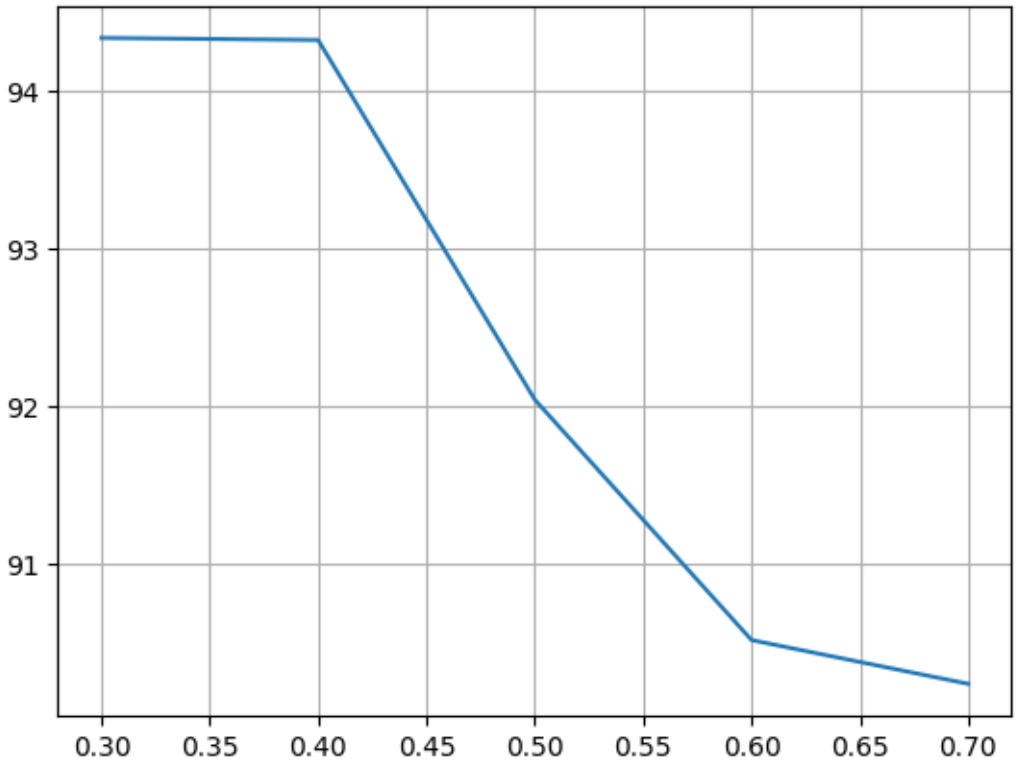



```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.96      0.77      0.85        91
           1       0.88      0.98      0.93       155
```

```
    accuracy                           0.90       246
   macro avg       0.92      0.87      0.89       246
weighted avg       0.91      0.90      0.90       246
```

0.0.7 split vs accuracy graph

[49]:
```
x_points = [float(key) for key in dict_mlp]
y_points = [i*100 for i in dict_mlp.values()]
plt.plot(x_points, y_points)
plt.grid(True)
plt.show()
```

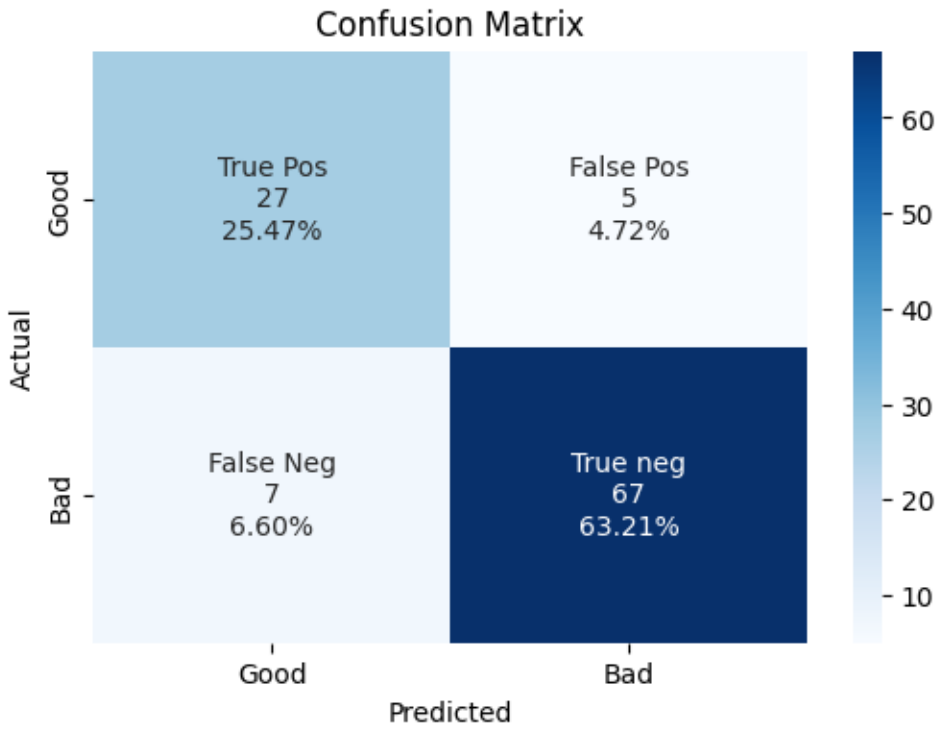


[50]:
```
def randomForest(split, estimator = 100, criterionValue = 'gini', ):
  from sklearn.model_selection import train_test_split
  from sklearn.ensemble import RandomForestClassifier
  from sklearn.metrics import accuracy_score
  from sklearn.preprocessing import StandardScaler
  scaler = StandardScaler()
  X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = split,␣
 ↪random_state=44)
```

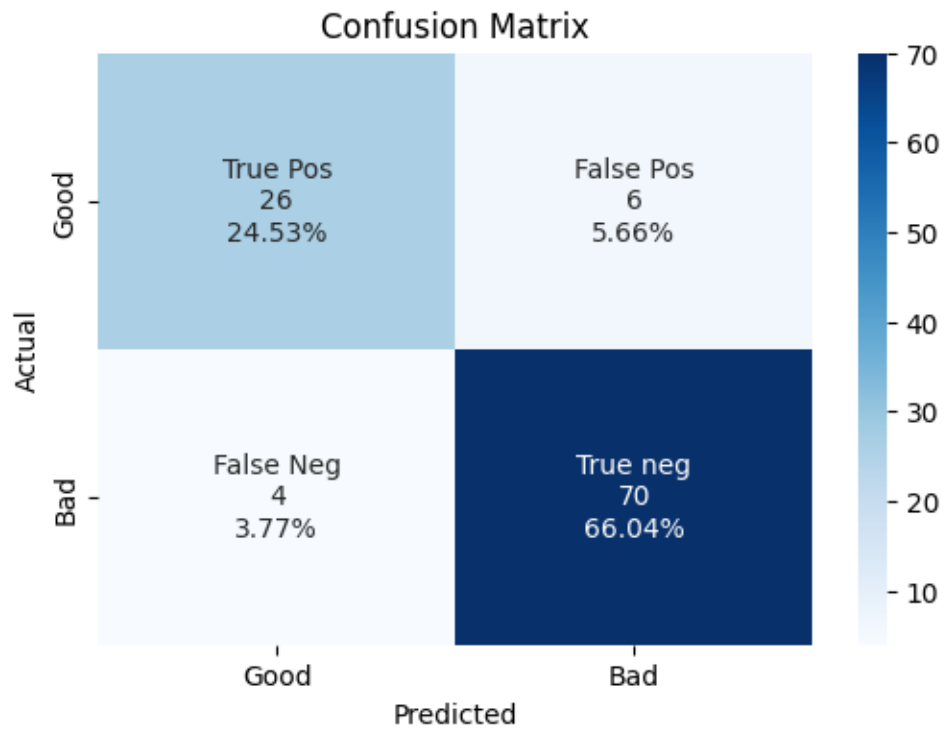
```
  scaler.fit_transform(X_train)
  scaler.transform(X_test)
  classifier = RandomForestClassifier(n_estimators = estimator, criterion =␣
↪criterionValue)
  classifier.fit(X_train, y_train)
  y_pred = classifier.predict(X_test)
  accuracy = accuracy_score(y_test, y_pred)

  if(str(split) in dict_rfr):
    dict_rfr[str(split)] = max(accuracy, dict_rfr[str(split)])
    if(str(split) == '0.3' and accuracy > dict_svm[str(split)]):
      RocAucRfr['max'] = {'y_test': y_test, 'y_pred': y_pred}
  else:
    dict_rfr[str(split)] = accuracy
    if(str(split) == '0.3'):
      RocAucRfr['max'] = {'y_test': y_test, 'y_pred': y_pred}

  reports(y_test, y_pred)
```

[51]:
```
randomForest(0.3, 170)
randomForest(0.3, 205, 'entropy')
```

```
Confusion Matrix :
```



```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.79      0.84      0.82        32
           1       0.93      0.91      0.92        74

    accuracy                           0.89       106
   macro avg       0.86      0.87      0.87       106
weighted avg       0.89      0.89      0.89       106

Confusion Matrix :
```

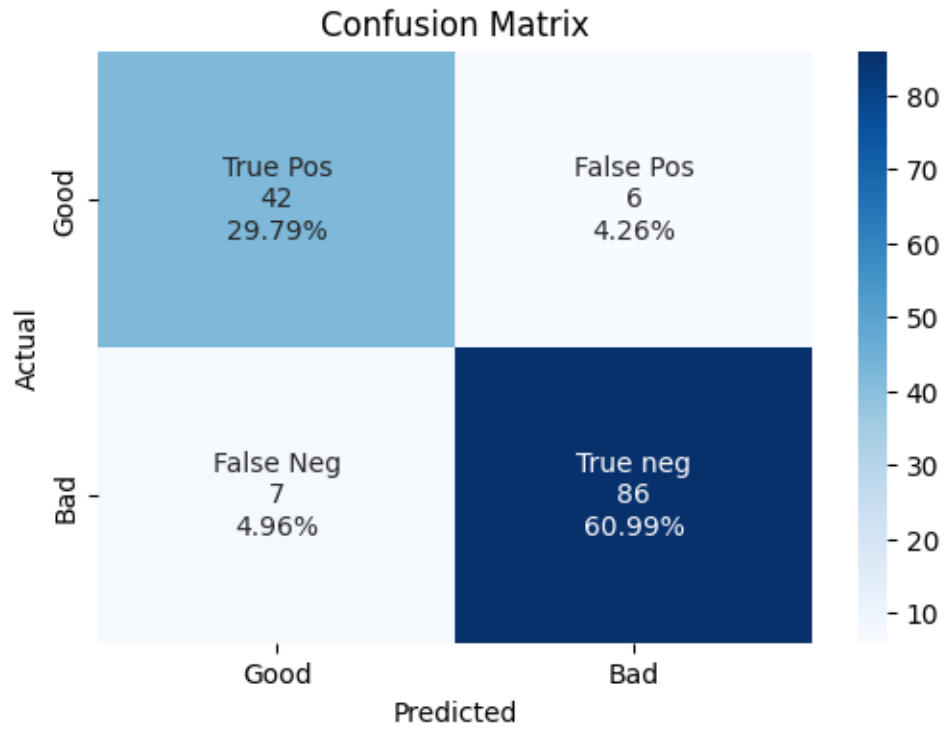



```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.87      0.81      0.84        32
           1       0.92      0.95      0.93        74

    accuracy                           0.91       106
   macro avg       0.89      0.88      0.89       106
```

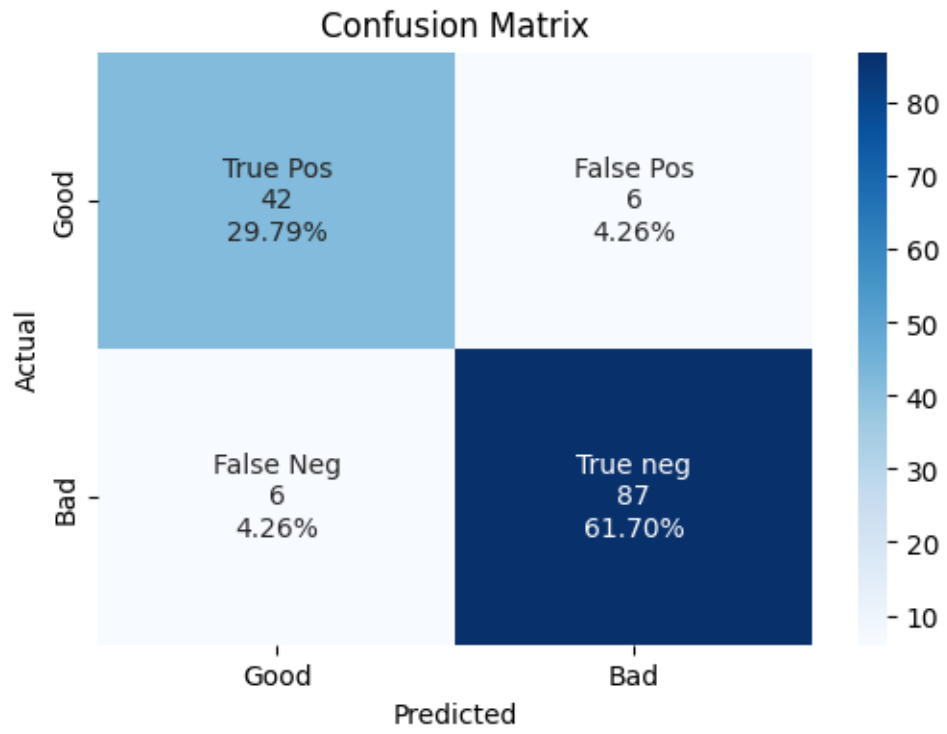
```
weighted avg       0.90      0.91      0.90       106
```

[52]:
```
randomForest(0.4, 70)
randomForest(0.4, 80, 'entropy')
```

```
Confusion Matrix :
```



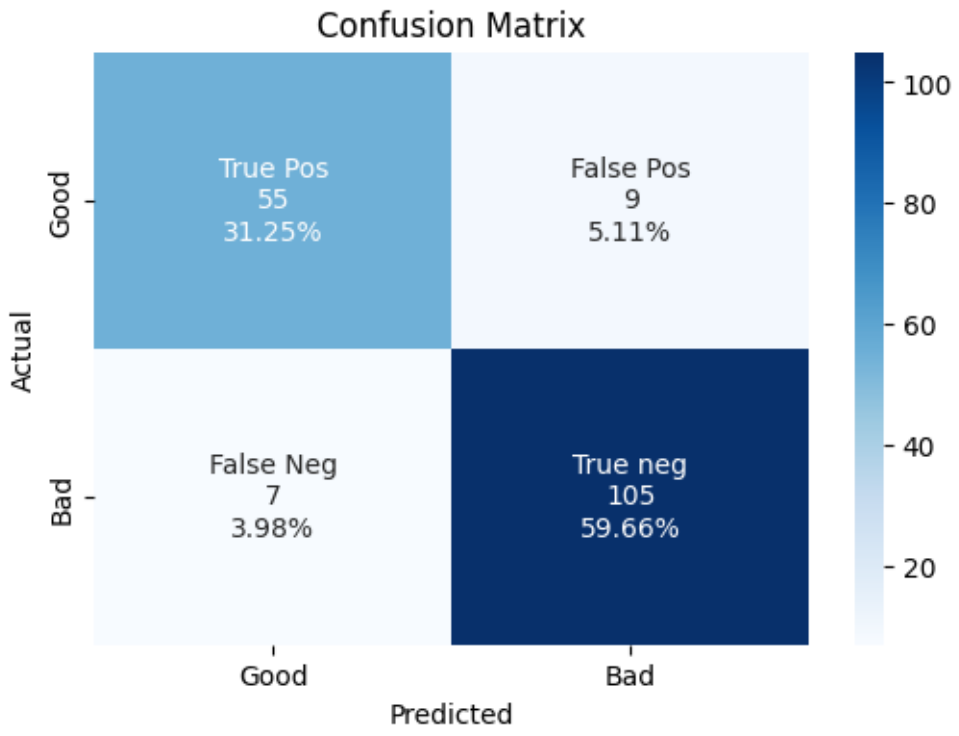

```
*********************************************
*********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.86      0.88      0.87        48
           1       0.93      0.92      0.93        93

    accuracy                           0.91       141
   macro avg       0.90      0.90      0.90       141
weighted avg       0.91      0.91      0.91       141

Confusion Matrix :
```



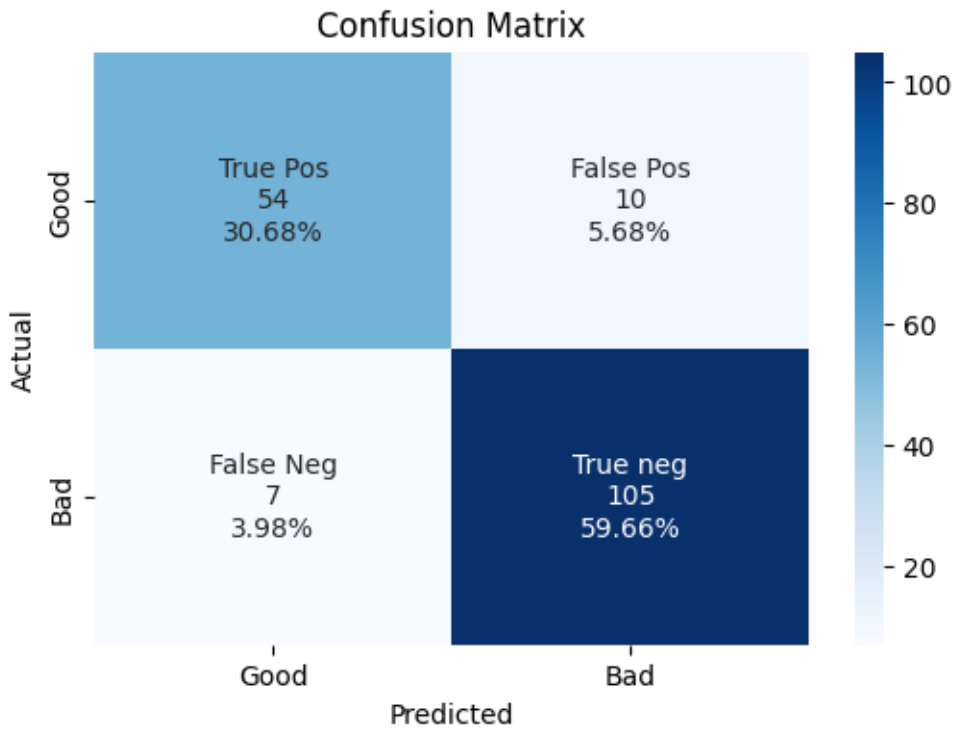

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.88      0.88      0.88        48
           1       0.94      0.94      0.94        93

    accuracy                           0.91       141
   macro avg       0.91      0.91      0.91       141
weighted avg       0.91      0.91      0.91       141
```

[53]:
```
randomForest(0.5, 66)
randomForest(0.5, 140, 'entropy')
```

```
Confusion Matrix :
```



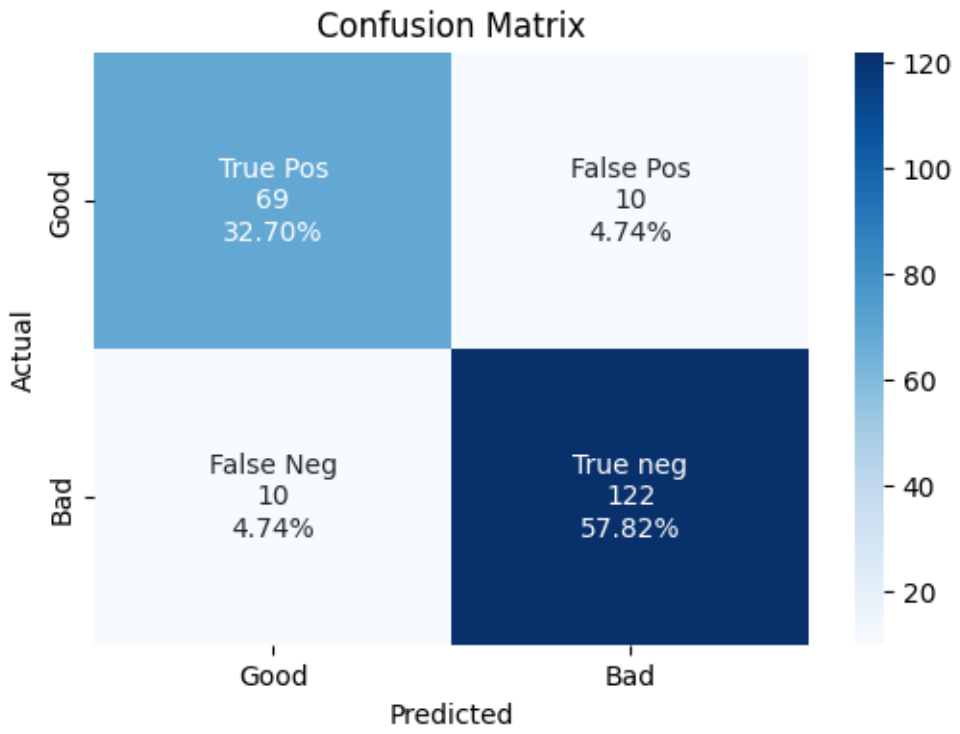

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.89      0.86      0.87        64
           1       0.92      0.94      0.93       112

    accuracy                           0.91       176
   macro avg       0.90      0.90      0.90       176
weighted avg       0.91      0.91      0.91       176

Confusion Matrix :
```



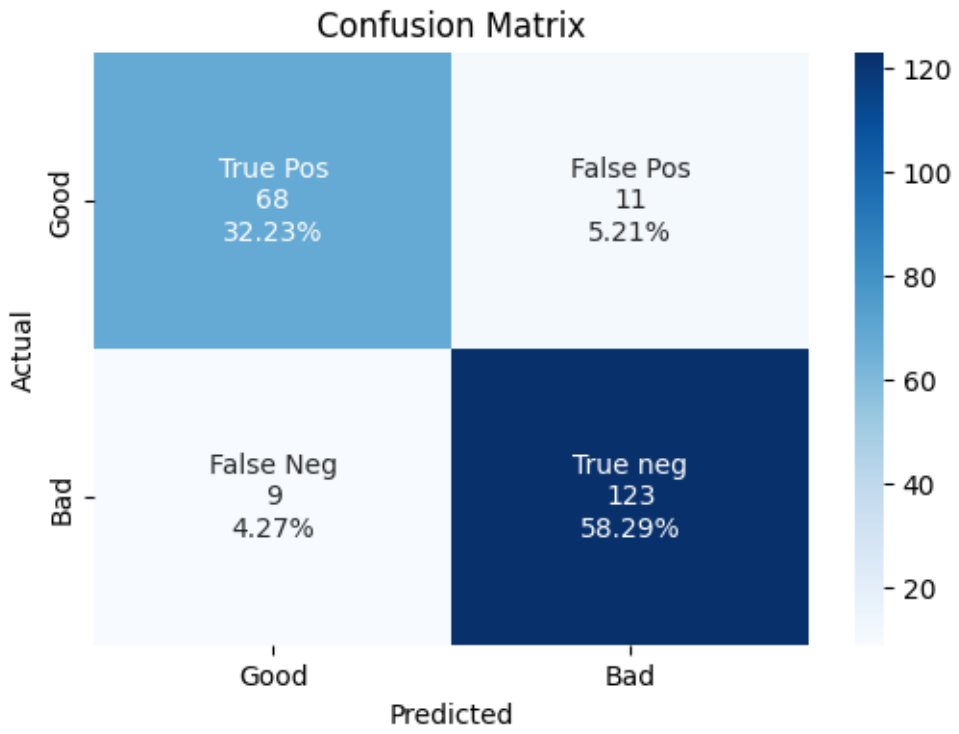

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.89      0.84      0.86        64
           1       0.91      0.94      0.93       112

    accuracy                           0.90       176
   macro avg       0.90      0.89      0.89       176
weighted avg       0.90      0.90      0.90       176
```

[54]:
```
randomForest(0.6, )
randomForest(0.6, 100, 'entropy')
```

```
Confusion Matrix :
```



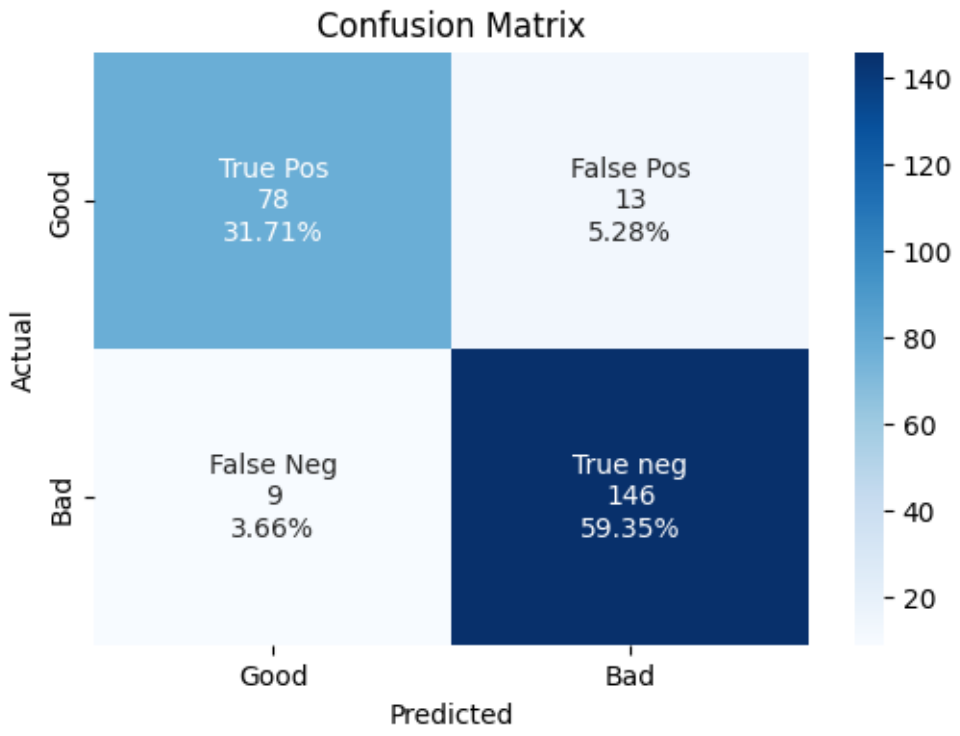

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.87      0.87      0.87        79
           1       0.92      0.92      0.92       132

    accuracy                           0.91       211
   macro avg       0.90      0.90      0.90       211
weighted avg       0.91      0.91      0.91       211

Confusion Matrix :
```



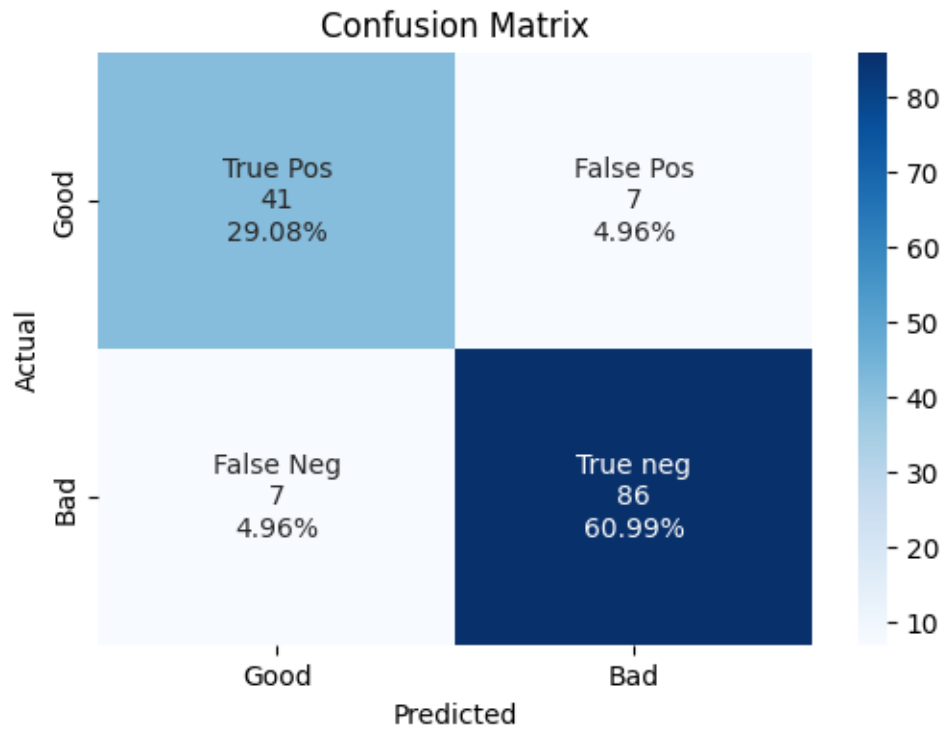

```
*********************************************
*********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.88      0.86      0.87        79
           1       0.92      0.93      0.92       132

    accuracy                           0.91       211
   macro avg       0.90      0.90      0.90       211
weighted avg       0.90      0.91      0.90       211
```

[55]:
```
randomForest(0.7, 120)
randomForest(0.4, 80, 'entropy')
```

```
Confusion Matrix :
```



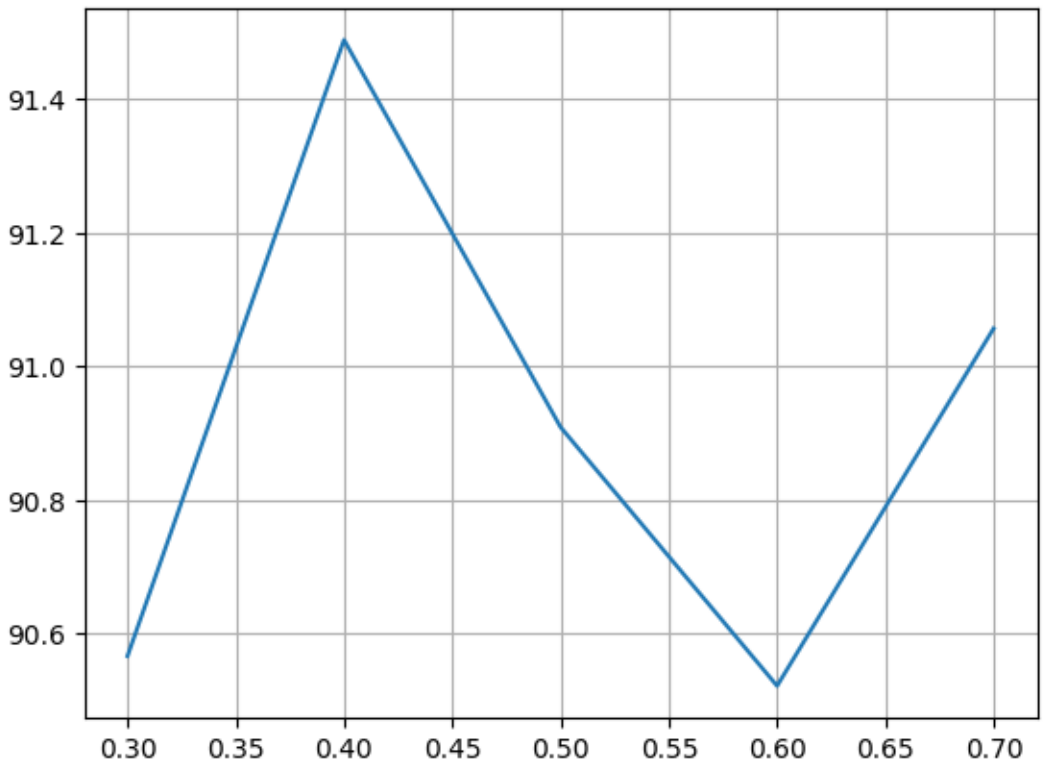

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.90      0.86      0.88        91
           1       0.92      0.94      0.93       155

    accuracy                           0.91       246
   macro avg       0.91      0.90      0.90       246
weighted avg       0.91      0.91      0.91       246

Confusion Matrix :
```




```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.85      0.85      0.85        48
           1       0.92      0.92      0.92        93

    accuracy                           0.90       141
   macro avg       0.89      0.89      0.89       141
weighted avg       0.90      0.90      0.90       141
```

0.0.8 split vs accuracy graph

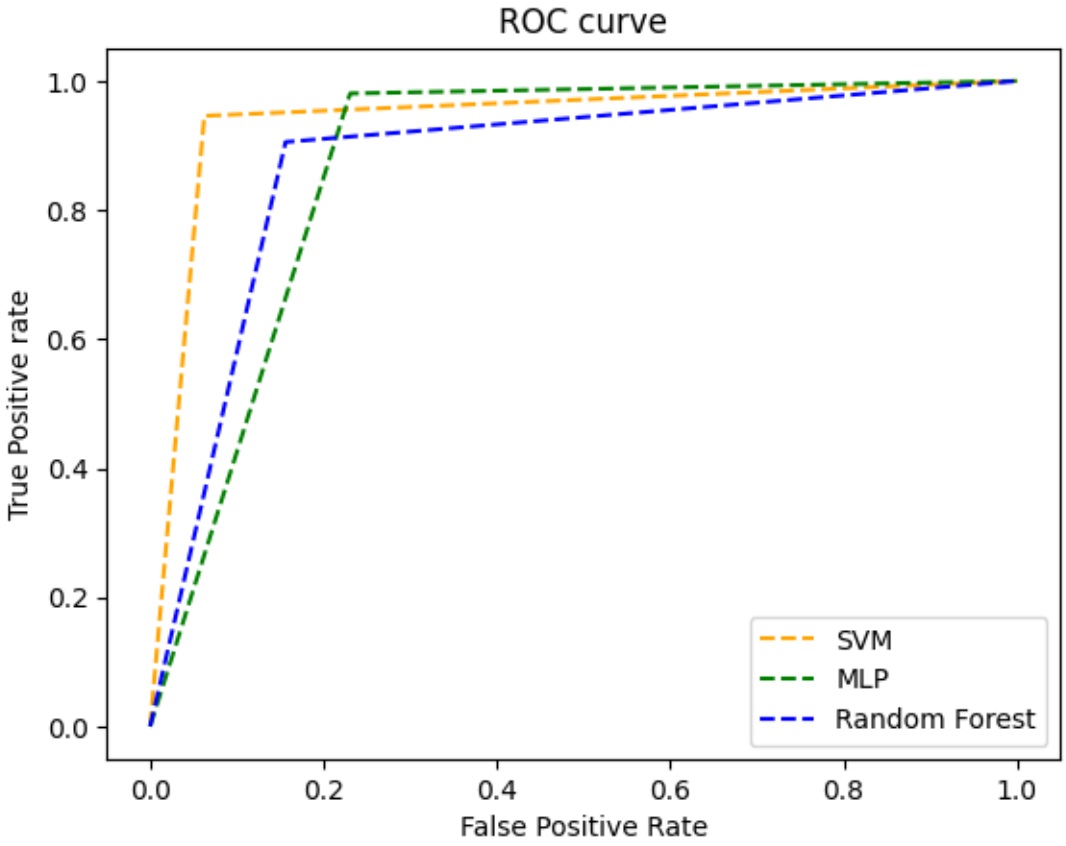
[56]:
```
x_points = [float(key) for key in dict_rfr]
y_points = [i*100 for i in dict_rfr.values()]
plt.plot(x_points, y_points)
plt.grid(True)
plt.show()
```



0.0.9 ROC curve and ROC_AUC score for all the classifier having maximum accuracy when train test split 70-30.

[57]:
```
from sklearn import metrics
def auc_roc():
    fpr1, tpr1, _1 = metrics.roc_curve(RocAucSvm['max']['y_test'],␣
 ↪RocAucSvm['max']['y_pred'], pos_label=1)
    fpr2, tpr2, _2 = metrics.roc_curve(RocAucMlp['max']['y_test'],␣
 ↪RocAucMlp['max']['y_pred'], pos_label=1)
    fpr3, tpr3, _3 = metrics.roc_curve(RocAucRfr['max']['y_test'],␣
 ↪RocAucRfr['max']['y_pred'], pos_label=1)
    plt.plot(fpr1, tpr1, linestyle='--',color='orange', label='SVM')
    plt.plot(fpr2, tpr2, linestyle='--',color='green', label='MLP')
    plt.plot(fpr3, tpr3, linestyle='--', color='blue', label= 'Random Forest')
    plt.title('ROC curve')
    # x label
    plt.xlabel('False Positive Rate')
    # y label
    plt.ylabel('True Positive rate')

    plt.legend(loc='best')
```

```
    plt.savefig('ROC',dpi=300)
    plt.show()
auc_roc()
```



0.0.10 Using PCA on Random Forest Classifiers

[58]:
```
# Standardizing the data (ionosphere dataset is already standardized)
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_std = scaler.fit_transform(X)
print(X_std)


# Performing PCA
from sklearn.decomposition import PCA

number_of_components = 10  # Number of components to retain (your choice)
pca = PCA(n_components=number_of_components)
transformed_data = pca.fit_transform(X)
```

```
print(transformed_data.shape)

## choose train-test split or hyperparameters accordingly
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
import seaborn as sns

X_train, X_test, y_train, y_test = train_test_split(transformed_data, y,␣
 ↪test_size=0.7, random_state=42)
rfc = RandomForestClassifier(n_estimators=100, criterion='gini')  # most␣
 ↪suitable hyperparameters
rfc.fit(X_train, y_train)
y_pred = rfc.predict(X_test)
print('-------------------------------------------------------------------------------')
print(
  'Classification report of Random Forest Classifier after PCA (' +␣
 ↪str(number_of_components) + ' components taken):  ')
print('-------------------------------------------------------------------------------')
print(classification_report(y_test, y_pred))

print("Confusion Matrix for the same: ")
cf_matrix = confusion_matrix(y_test, y_pred)
group_names = ['True Pos', 'False Pos', 'False Neg', 'True neg']
group_counts = ["{0:0.0f}".format(value) for value in
                cf_matrix.flatten()]
group_percentages = ["{0:.2%}".format(value) for value in
                     cf_matrix.flatten() / np.sum(cf_matrix)]
labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in
          zip(group_names, group_counts, group_percentages)]
labels = np.asarray(labels).reshape(2, 2)
plt.figure(figsize=(6, 4))
print(sns.heatmap(cf_matrix, annot=labels, fmt='', cmap='Blues',␣
 ↪xticklabels=['Benign', 'Malignant'],
                  yticklabels=['Benign', 'Malignant']))
```

```
[[ 0.34843328  0.          0.71237237 … -1.05505394 -0.3122206
  -0.99959483]
 [ 0.34843328  0.          0.72164805 … -0.11521328 -0.93260505
  -0.08328554]
 [ 0.34843328  0.          0.72164805 … -0.46409249  0.40444328
  -0.84859079]
 …
 [ 0.34843328  0.          0.61502805 …  0.01601615  1.10669878
  -0.04330004]
 [ 0.34843328  0.          0.53267371 … -0.06586087  1.00526528
```

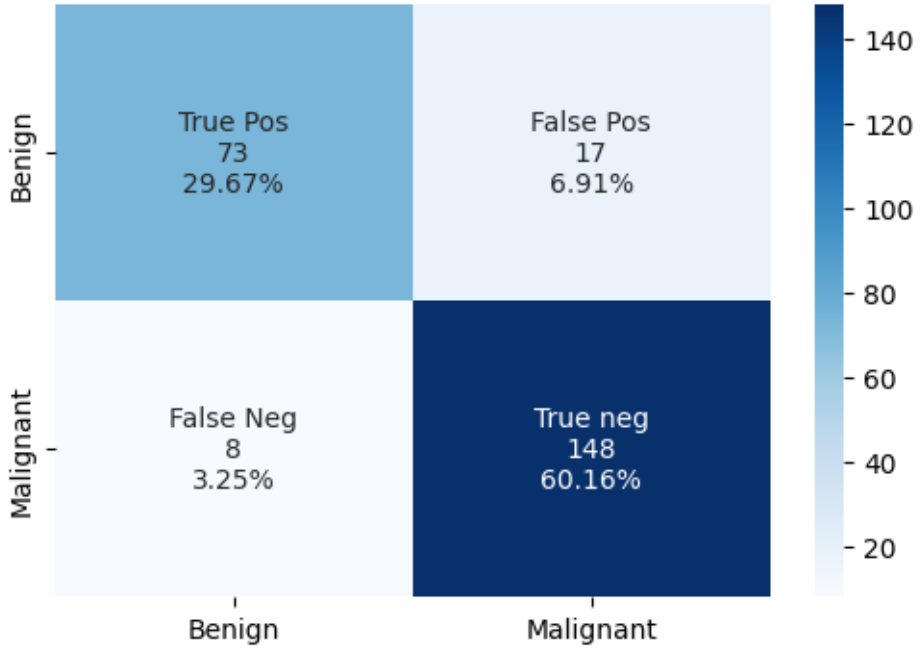
```
  -0.37828012]
 [ 0.34843328  0.          0.41400137 … -0.12281796  0.9738619
  -0.16248675]]
(351, 10)
```

```
Classification report of Random Forest Classifier after PCA (10 components
taken):
```

---

	precision	recall	f1-score	support
0	0.90	0.81	0.85	90
1	0.90	0.95	0.92	156
accuracy			0.90	246
macro avg	0.90	0.88	0.89	246
weighted avg	0.90	0.90	0.90	246

```
Confusion Matrix for the same:
Axes(0.125,0.11;0.62x0.77)
```



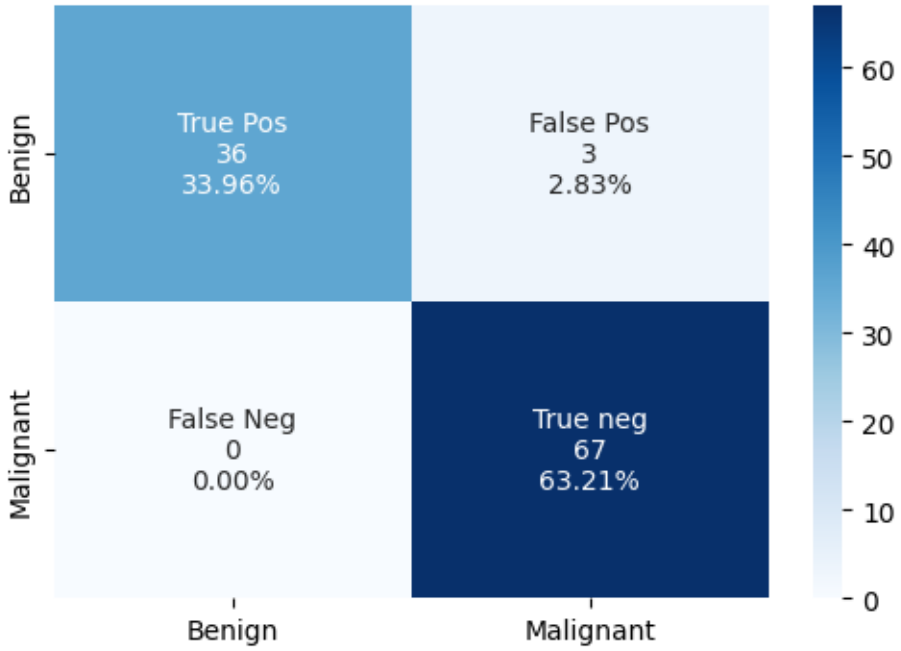
0.0.11 Using PCA on Support Vector Machines

[59]:
```
# Performing PCA
from sklearn.decomposition import PCA
```

```
number_of_components = 12  # Number of components to retain (your choice)
pca = PCA(n_components=number_of_components)
transformed_data = pca.fit_transform(X)
print(transformed_data.shape)

## choose train-test split or hyperparameters accordingly
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
import seaborn as sns

X_train, X_test, y_train, y_test = train_test_split(transformed_data, y,␣
 ↪test_size=0.3, random_state=42)
rfc = SVC(gamma='scale', kernel='rbf', degree=3)  # most suitable␣
 ↪hyperparameters
rfc.fit(X_train, y_train)
y_pred = rfc.predict(X_test)
print('-----------------------------------------------------------------------------------')
print(
  'Classification report of Random Forest Classifier after PCA (' +␣
 ↪str(number_of_components) + ' components taken):  ')
print('-----------------------------------------------------------------------------------')
print(classification_report(y_test, y_pred))

print("Confusion Matrix for the same: ")
cf_matrix = confusion_matrix(y_test, y_pred)
group_names = ['True Pos', 'False Pos', 'False Neg', 'True neg']
group_counts = ["{0:0.0f}".format(value) for value in
                cf_matrix.flatten()]
group_percentages = ["{0:.2%}".format(value) for value in
                     cf_matrix.flatten() / np.sum(cf_matrix)]
labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in
          zip(group_names, group_counts, group_percentages)]
labels = np.asarray(labels).reshape(2, 2)
plt.figure(figsize=(6, 4))
print(sns.heatmap(cf_matrix, annot=labels, fmt='', cmap='Blues',␣
 ↪xticklabels=['Benign', 'Malignant'],
                  yticklabels=['Benign', 'Malignant']))
```

```
(351, 12)
-----------------------------------------------------------------------------------
Classification report of Random Forest Classifier after PCA (12 components
taken):
-----------------------------------------------------------------------------------
              precision    recall  f1-score   support
```

```
              0       1.00      0.92      0.96        39
              1       0.96      1.00      0.98        67

    accuracy                            0.97       106
   macro avg       0.98      0.96      0.97       106
weighted avg       0.97      0.97      0.97       106

Confusion Matrix for the same:
Axes(0.125,0.11;0.62x0.77)
```

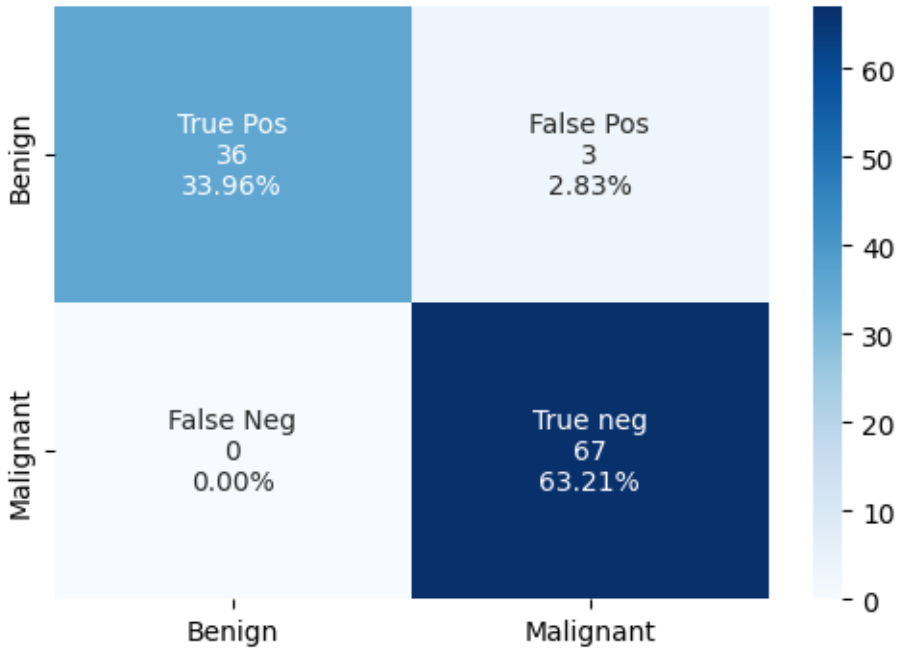


0.0.12 Using PCA on Multi Layer Perceptron

[60]:
```
# Performing PCA
from sklearn.decomposition import PCA

number_of_components = 12  # Number of components to retain (your choice)
pca = PCA(n_components=number_of_components)
transformed_data = pca.fit_transform(X)
print(transformed_data.shape)

## choose train-test split or hyperparameters accordingly
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
```

```
import seaborn as sns

X_train, X_test, y_train, y_test = train_test_split(transformed_data, y,␣
 ↪test_size=0.3, random_state=42)
rfc = SVC(gamma='scale', kernel='rbf', degree=3)  # most suitable␣
 ↪hyperparameters
rfc.fit(X_train, y_train)
y_pred = rfc.predict(X_test)
print('-----------------------------------------------------------------------------------')
print(
  'Classification report of Random Forest Classifier after PCA (' +␣
 ↪str(number_of_components) + ' components taken):  ')
print('-----------------------------------------------------------------------------------')
print(classification_report(y_test, y_pred))

print("Confusion Matrix for the same: ")
cf_matrix = confusion_matrix(y_test, y_pred)
group_names = ['True Pos', 'False Pos', 'False Neg', 'True neg']
group_counts = ["{0:0.0f}".format(value) for value in
                cf_matrix.flatten()]
group_percentages = ["{0:.2%}".format(value) for value in
                     cf_matrix.flatten() / np.sum(cf_matrix)]
labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in
          zip(group_names, group_counts, group_percentages)]
labels = np.asarray(labels).reshape(2, 2)
plt.figure(figsize=(6, 4))
print(sns.heatmap(cf_matrix, annot=labels, fmt='', cmap='Blues',␣
 ↪xticklabels=['Benign', 'Malignant'],
                  yticklabels=['Benign', 'Malignant']))
```

```
(351, 12)
-----------------------------------------------------------------------------------
Classification report of Random Forest Classifier after PCA (12 components
taken):
-----------------------------------------------------------------------------------
              precision    recall  f1-score   support

           0       1.00      0.92      0.96        39
           1       0.96      1.00      0.98        67

    accuracy                           0.97       106
   macro avg       0.98      0.96      0.97       106
weighted avg       0.97      0.97      0.97       106

Confusion Matrix for the same:
Axes(0.125,0.11;0.62x0.77)
```


True Pos
36
33.96%
False Pos
3
2.83%
False Neg
0
0.00%
True neg
67
63.21%
Benign
Malignant
Benign
Malignant
60
50
40
30
20
10
0

[ ]:

ncer-wisconsin-diagnostic-data-set

August 13, 2023

0.0.1 Breast Cancer Wisconsin (Diagnostic) Dataset classification using SVM, MLP and Random Forest classifier

[29]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

[ ]:
```
url = 'https://raw.githubusercontent.com/Aqeel-0/phone.html/master/data.csv'
df = pd.read_csv(url)
df.head()
```

0.0.2 Pre Preprocessing

[ ]:
```
from sklearn.preprocessing import LabelEncoder
X = df.drop(['id', 'diagnosis', 'Unnamed: 32'], axis=1)
X.info()
le = LabelEncoder()
encoded = le.fit_transform(df['diagnosis'])
df.drop("diagnosis", axis=1, inplace=True)
df["diagnosis"] = encoded
y = df["diagnosis"]
y.info()
dict_svm = {}
dict_mlp = {}
dict_rfr = {}
RocAucSvm = {}
RocAucMlp = {}
RocAucRfr = {}
```

0.0.3 Used for ploting confusion matrix

[32]:
```
def plot(y_test, y_pred):
  from sklearn.metrics import confusion_matrix
  import seaborn as sns
  from sklearn.metrics import roc_curve
  from sklearn.metrics import auc
```

```
  print("Confusion Matrix : ")
  cf_matrix = confusion_matrix(y_test, y_pred)
  group_names = ['True Pos','False Pos','False Neg','True neg']
  group_counts = ["{0:0.0f}".format(value) for value in
                  cf_matrix.flatten()]
  group_percentages = ["{0:.2%}".format(value) for value in
                       cf_matrix.flatten()/np.sum(cf_matrix)]
  labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in
            zip(group_names,group_counts,group_percentages)]
  labels = np.asarray(labels).reshape(2,2)
  plt.figure(figsize=(6, 4))
  sns.heatmap(cf_matrix, annot=labels, fmt='', cmap='Blues',␣
 ↪xticklabels=['Benign', 'Malignant'], yticklabels=['Benign', 'Malignant'])
  plt.xlabel('Predicted')
  plt.ylabel('Actual')
  plt.title('Confusion Matrix')
  plt.show()
  print("*********************************************")
```

[33]:
```
def reports(y_test, y_pred):
  from sklearn.metrics import classification_report
  plot(y_test, y_pred)
  print("*********************************************")
  print("Classification Evaluation : ")
  print(classification_report(y_test, y_pred, zero_division = 0))
```

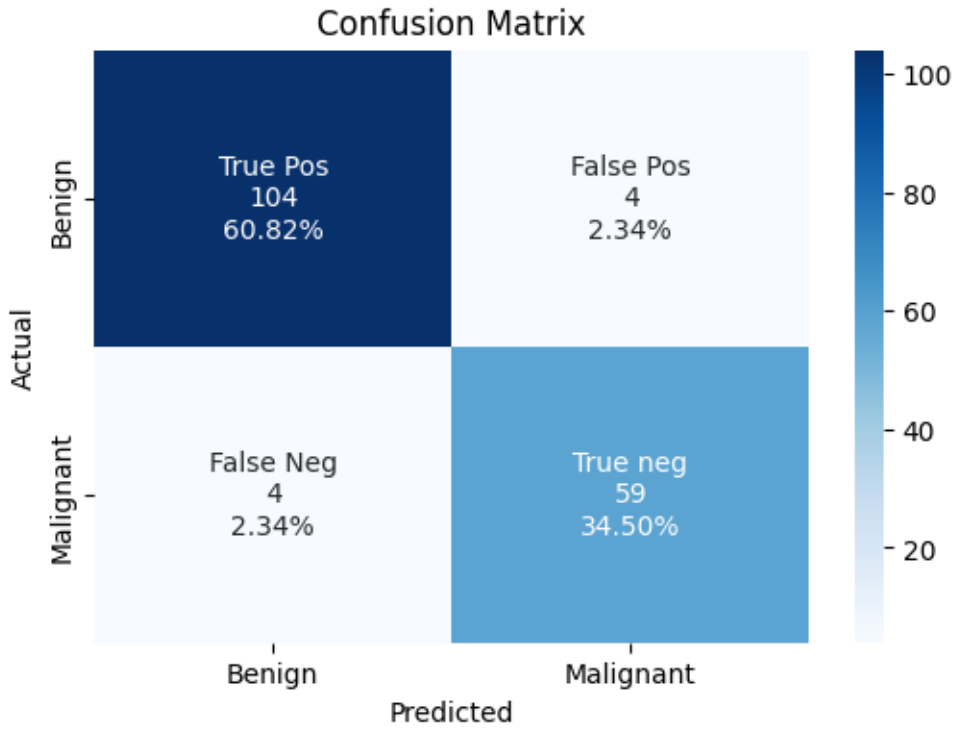
0.0.4 SVMClassifier

[34]:
```
def SVMClassifier(split, kernalValue = 'rbf', degreeValue = 3, gammaValue =␣
 ↪'scale', maxIter = -1):
  from sklearn.model_selection import train_test_split
  from sklearn.svm import SVC
  from sklearn.metrics import accuracy_score
  from sklearn.preprocessing import StandardScaler
  scaler = StandardScaler()
  X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = split,␣
 ↪random_state=44)
  scaler.fit_transform(X_train)
  scaler.transform(X_test)
  classifier = SVC(kernel = kernalValue, degree = degreeValue, gamma =␣
 ↪gammaValue, max_iter = maxIter)
  classifier.fit(X_train, y_train)
  y_pred = classifier.predict(X_test)
  accuracy = accuracy_score(y_test, y_pred)
```

```
  if str(split) in dict_svm:
    dict_svm[str(split)] = max(accuracy, dict_svm[str(split)])
    if str(split) == '0.3' and accuracy > dict_svm[str(split)]:
      RocAucSvm['max'] = {'y_test': y_test, 'y_pred': y_pred}
  else:
    dict_svm[str(split)] = accuracy
    if str(split) == '0.3':
      RocAucSvm['max'] = {'y_test': y_test, 'y_pred': y_pred}
  reports(y_test, y_pred)
```

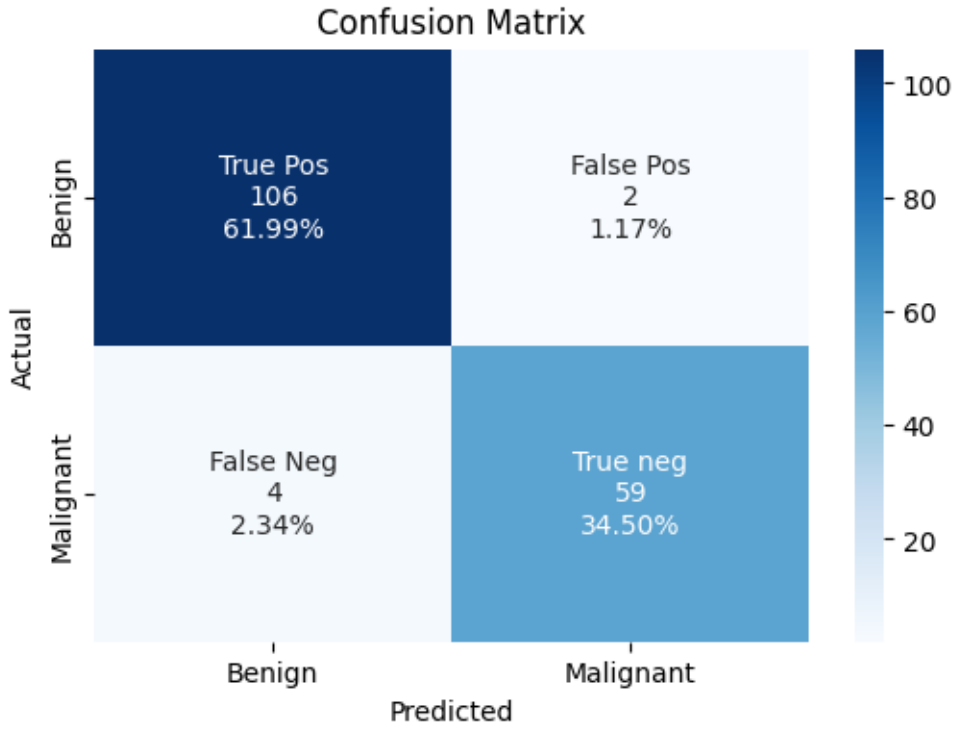
[35]:
```
#Train - Test split 70-30
SVMClassifier(0.3, 'rbf', 3)
SVMClassifier(0.3, 'linear', 3,)
SVMClassifier(0.3, 'poly', 2, )
SVMClassifier(0.3, 'sigmoid', 3, 0.01)
```

Confusion Matrix :

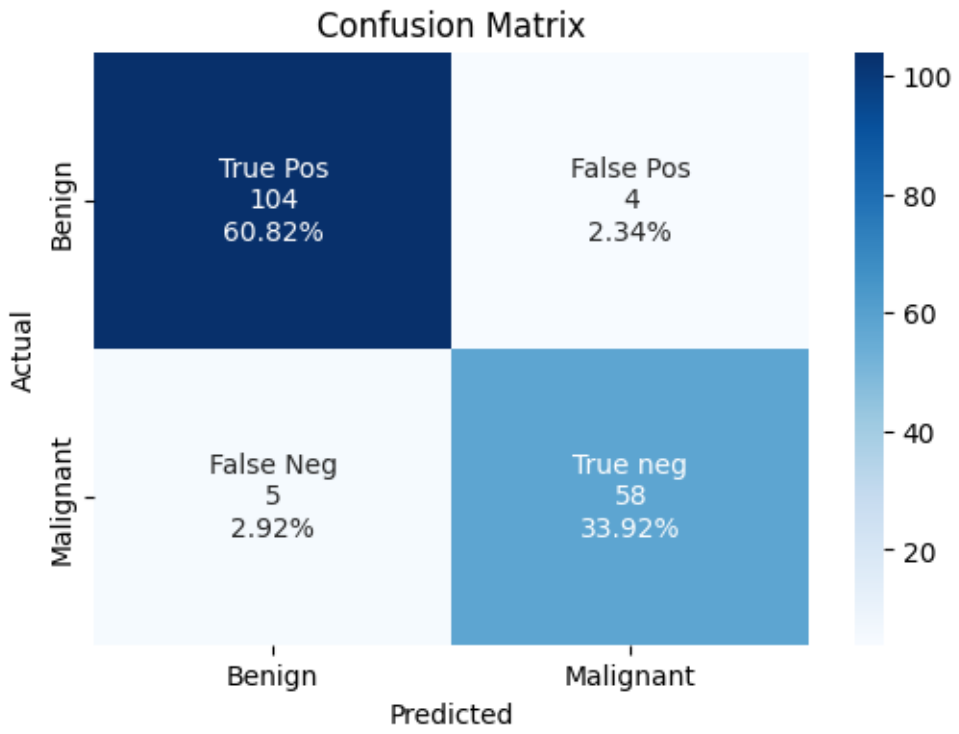



```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.96      0.96      0.96       108
```

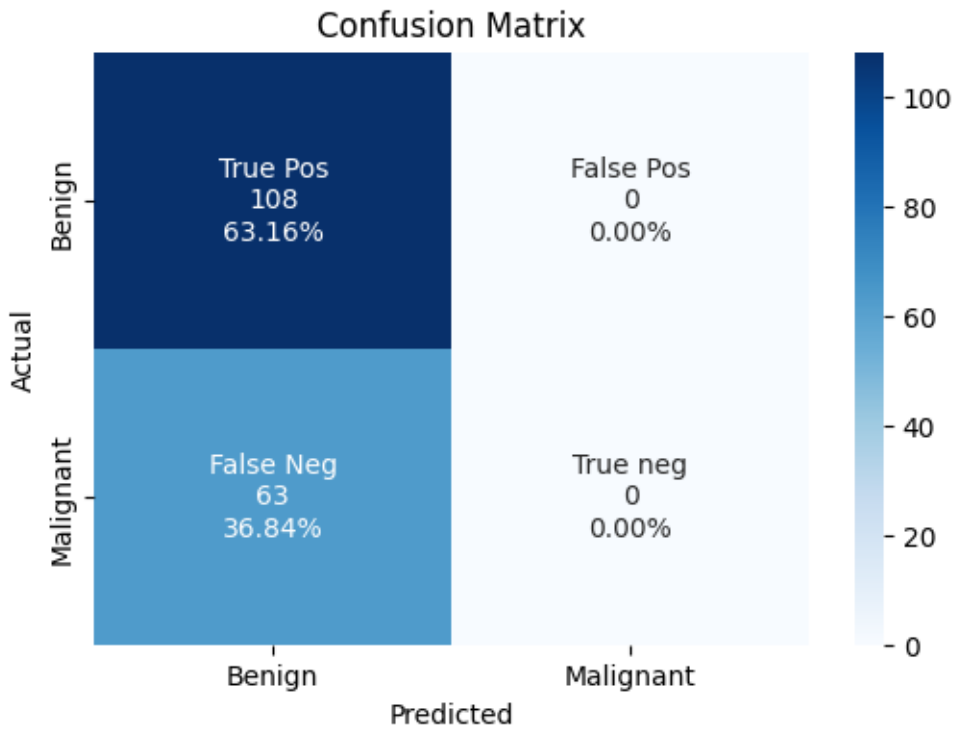
```
           1       0.94      0.94      0.94        63

    accuracy                           0.95       171
   macro avg       0.95      0.95      0.95       171
weighted avg       0.95      0.95      0.95       171

Confusion Matrix :
```




```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.96      0.98      0.97       108
           1       0.97      0.94      0.95        63

    accuracy                           0.96       171
   macro avg       0.97      0.96      0.96       171
weighted avg       0.96      0.96      0.96       171

Confusion Matrix :
```



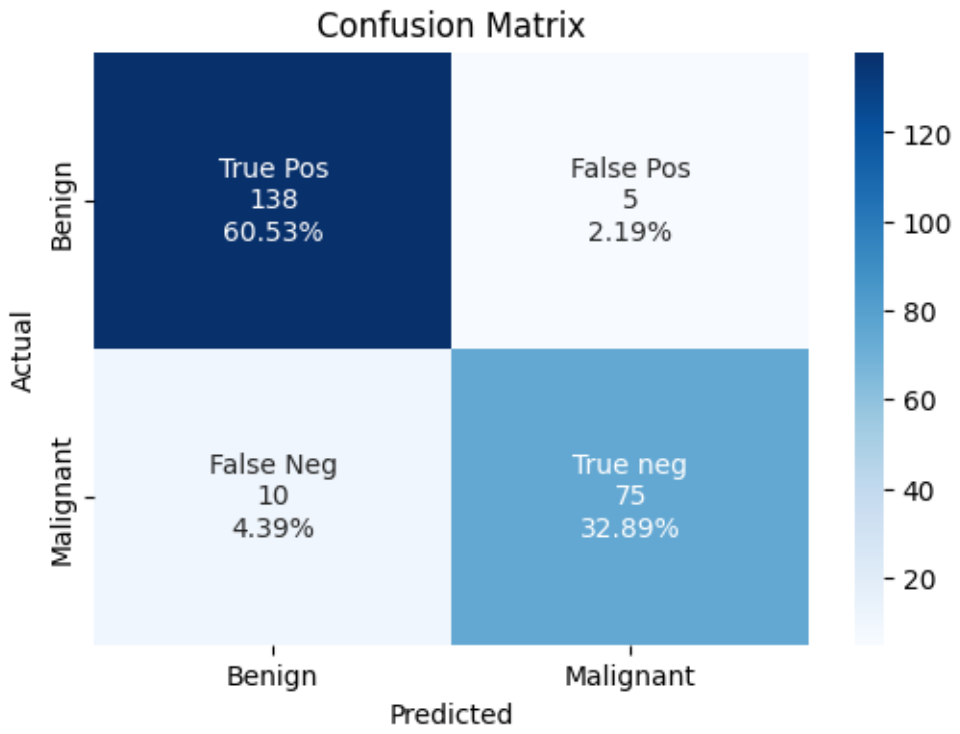

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.95      0.96      0.96       108
           1       0.94      0.92      0.93        63

    accuracy                           0.95       171
   macro avg       0.94      0.94      0.94       171
weighted avg       0.95      0.95      0.95       171

Confusion Matrix :
```



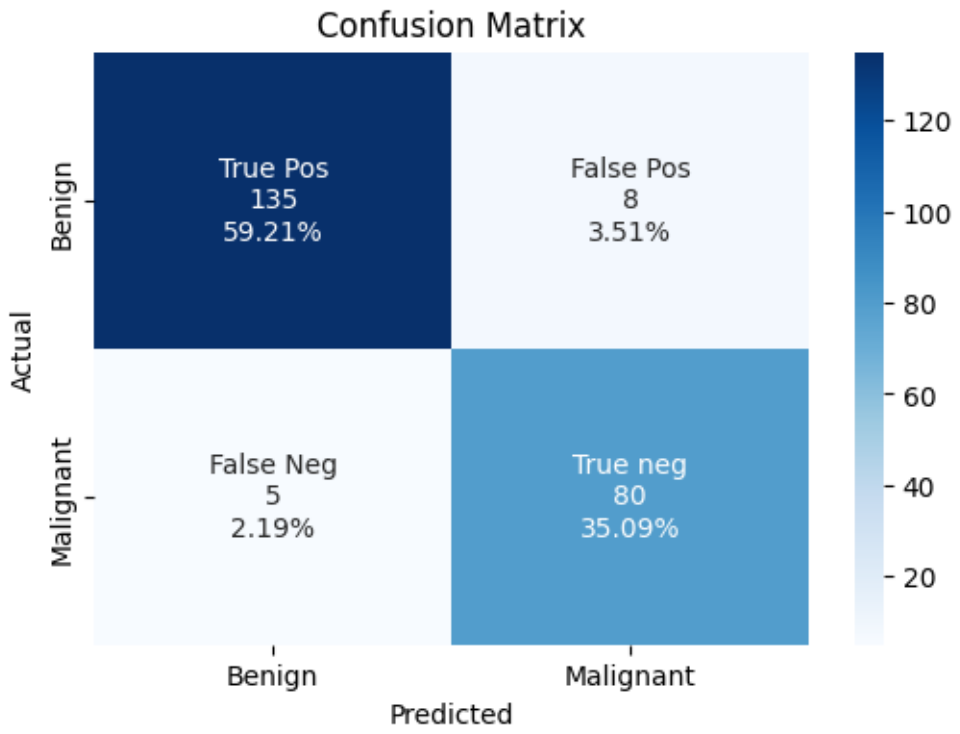

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.63      1.00      0.77       108
           1       0.00      0.00      0.00        63

    accuracy                           0.63       171
   macro avg       0.32      0.50      0.39       171
weighted avg       0.40      0.63      0.49       171
```

[36]:
```
#Train - Test split 60-40
SVMClassifier(0.4, 'rbf', 3,)
SVMClassifier(0.4, 'linear', 3,)
SVMClassifier(0.4, 'poly', 5, )
SVMClassifier(0.4, 'sigmoid', 3, 0.1)
```

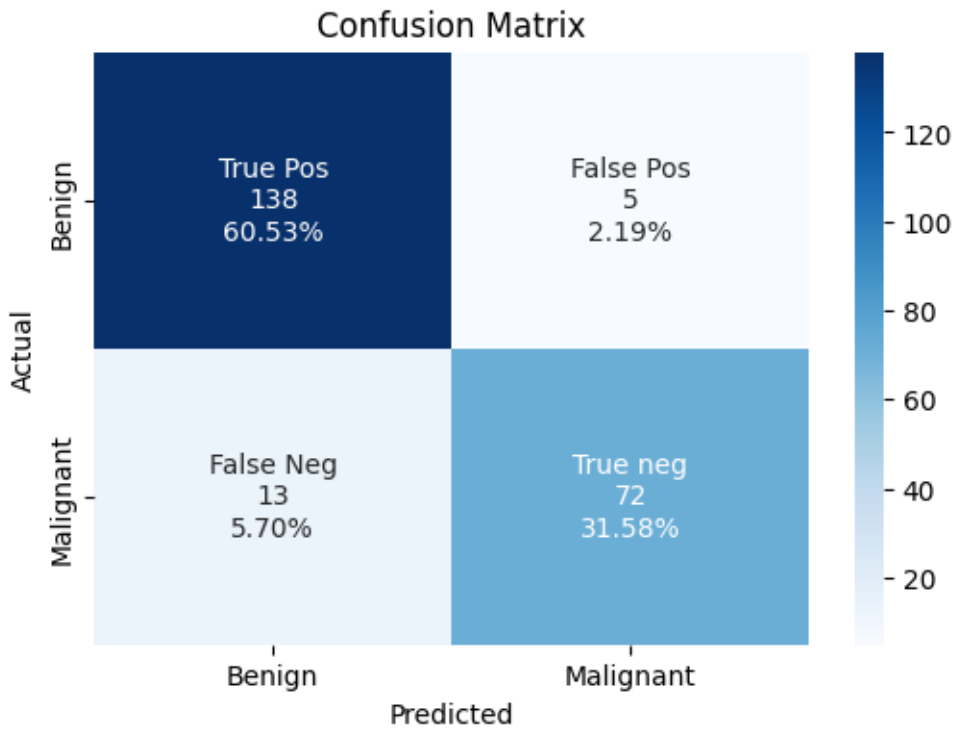
```
Confusion Matrix :
```



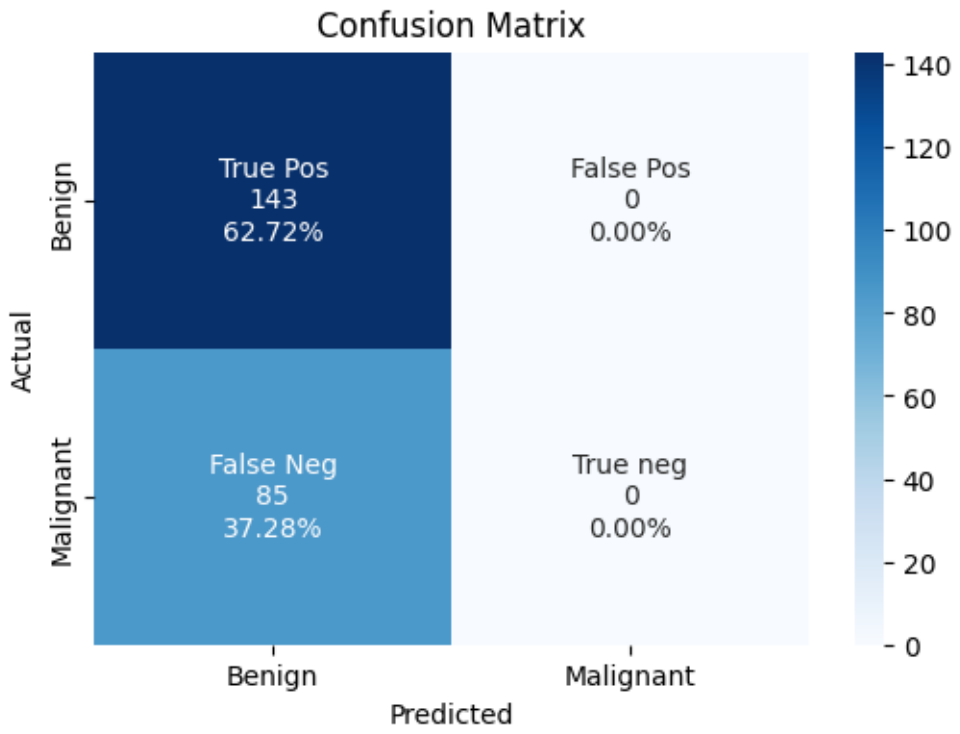

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.93      0.97      0.95       143
           1       0.94      0.88      0.91        85

    accuracy                           0.93       228
   macro avg       0.93      0.92      0.93       228
weighted avg       0.93      0.93      0.93       228

Confusion Matrix :
```




```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.96      0.94      0.95       143
           1       0.91      0.94      0.92        85

    accuracy                           0.94       228
   macro avg       0.94      0.94      0.94       228
weighted avg       0.94      0.94      0.94       228

Confusion Matrix :
```



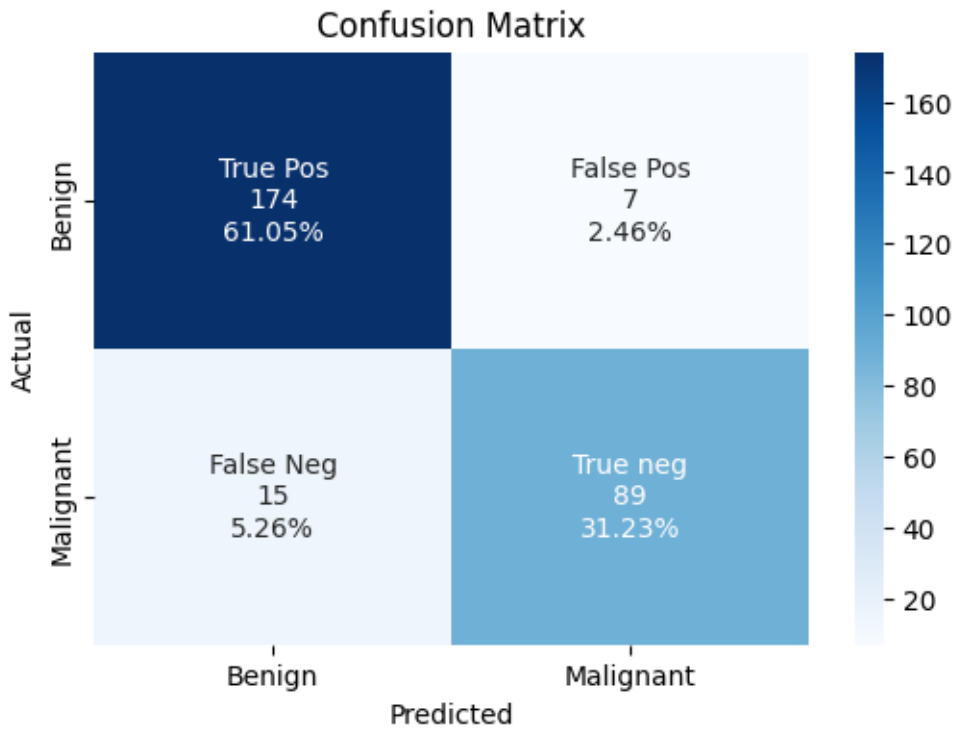

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.91      0.97      0.94       143
           1       0.94      0.85      0.89        85

    accuracy                           0.92       228
   macro avg       0.92      0.91      0.91       228
weighted avg       0.92      0.92      0.92       228

Confusion Matrix :
```



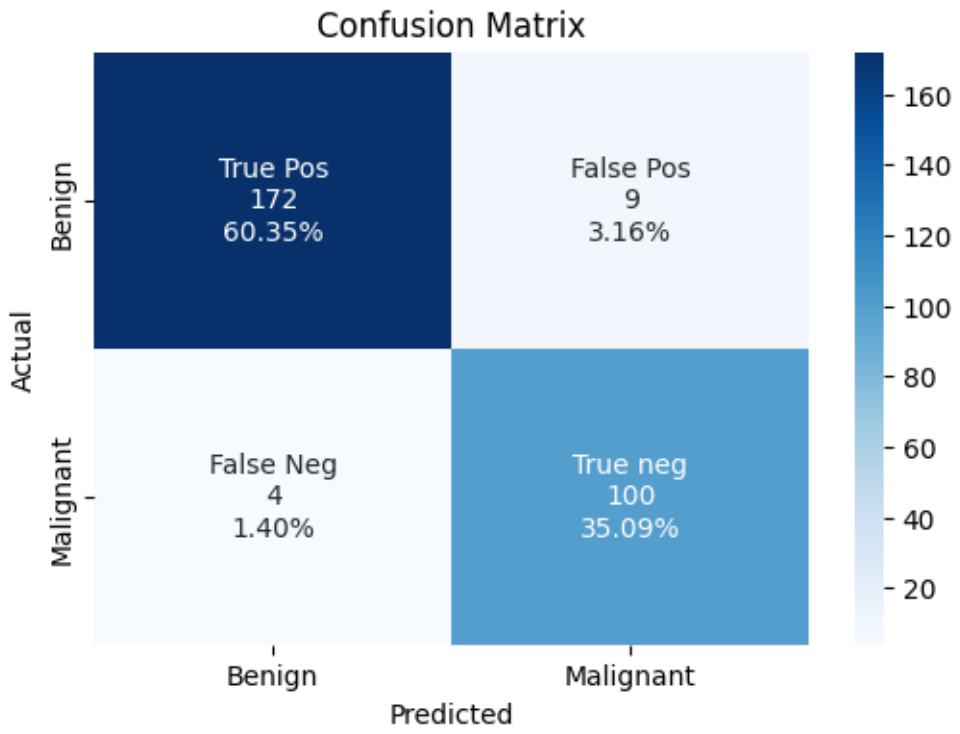

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.63      1.00      0.77       143
           1       0.00      0.00      0.00        85

    accuracy                           0.63       228
   macro avg       0.31      0.50      0.39       228
weighted avg       0.39      0.63      0.48       228
```

[37]:
```
#Train - Test split 50-50
SVMClassifier(0.5, 'rbf', 3,)
SVMClassifier(0.5, 'linear', 3, )
SVMClassifier(0.5, 'poly', 4, )
SVMClassifier(0.5, 'sigmoid', 3, 0.3 ) #wrost performance
```

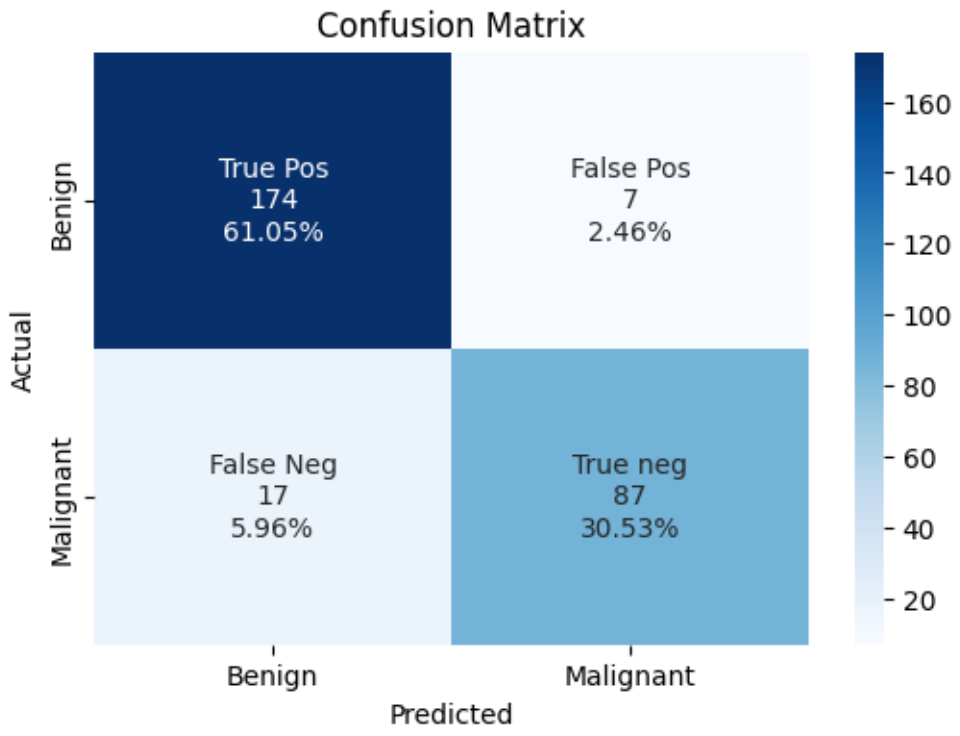
```
Confusion Matrix :
```



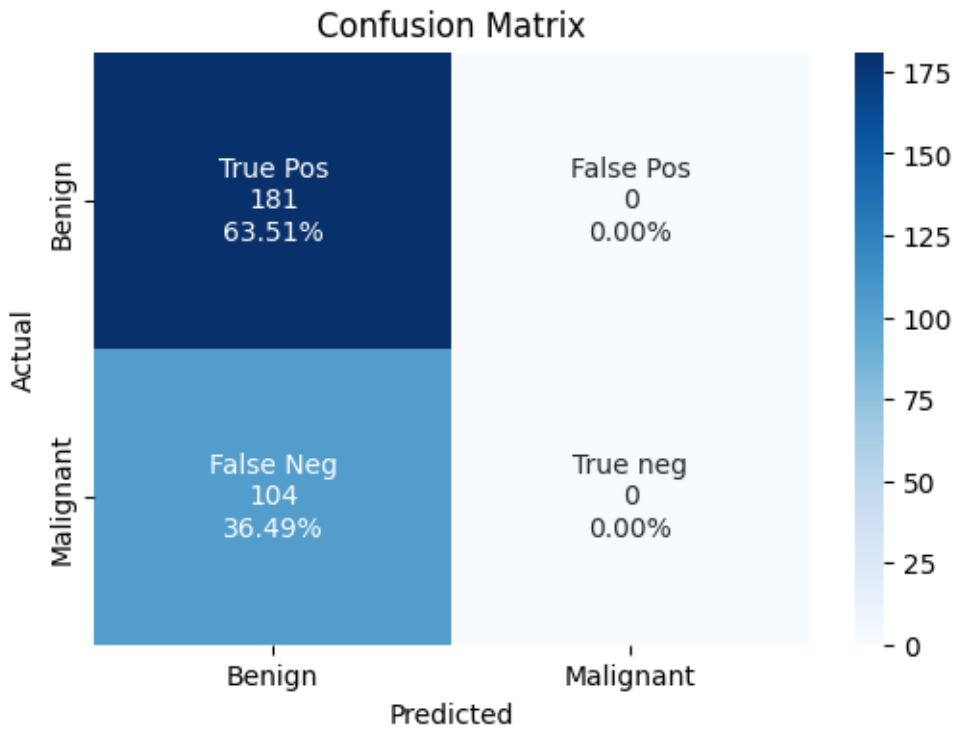

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.92      0.96      0.94       181
           1       0.93      0.86      0.89       104

    accuracy                           0.92       285
   macro avg       0.92      0.91      0.92       285
weighted avg       0.92      0.92      0.92       285

Confusion Matrix :
```




```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.98      0.95      0.96       181
           1       0.92      0.96      0.94       104

    accuracy                           0.95       285
   macro avg       0.95      0.96      0.95       285
weighted avg       0.96      0.95      0.95       285

Confusion Matrix :
```



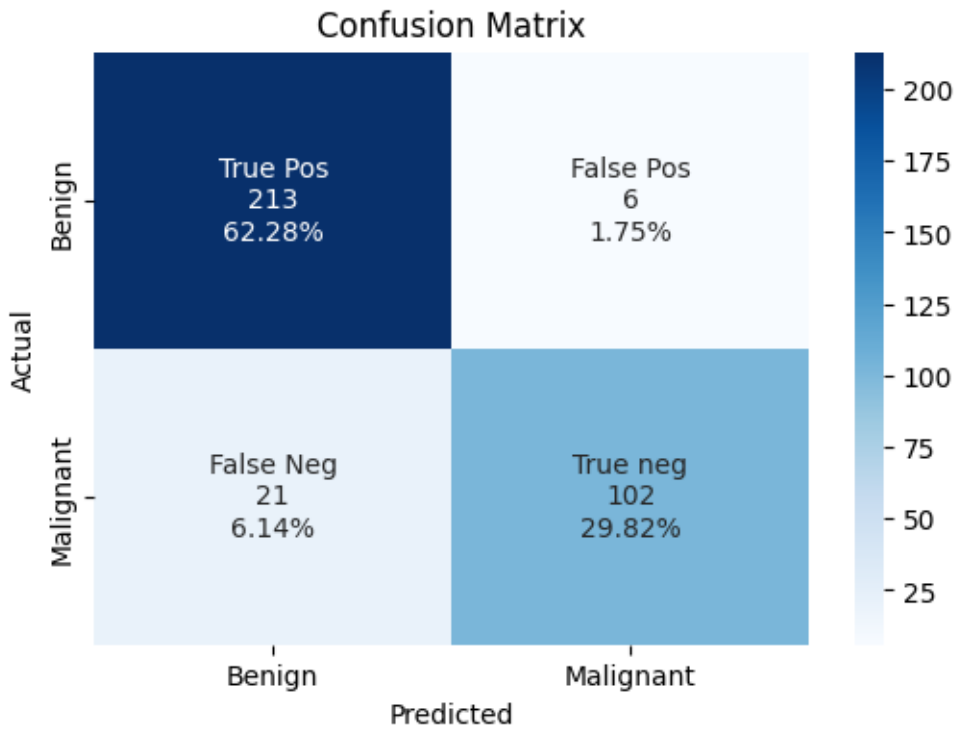

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.91      0.96      0.94       181
           1       0.93      0.84      0.88       104

    accuracy                           0.92       285
   macro avg       0.92      0.90      0.91       285
weighted avg       0.92      0.92      0.91       285

Confusion Matrix :
```



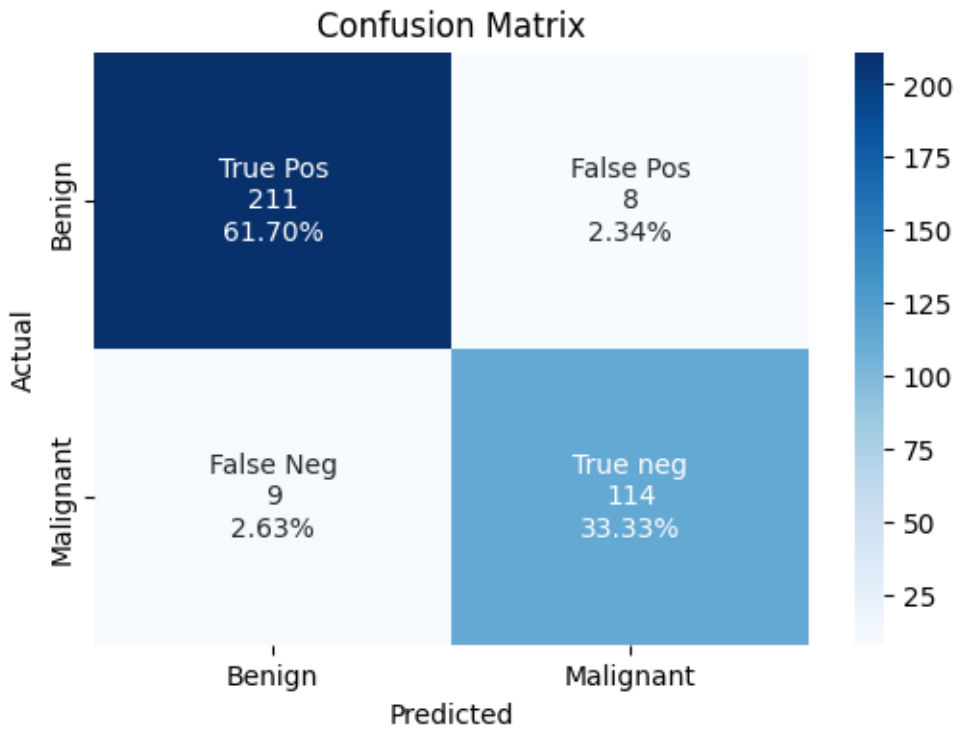

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.64      1.00      0.78       181
           1       0.00      0.00      0.00       104

    accuracy                           0.64       285
   macro avg       0.32      0.50      0.39       285
weighted avg       0.40      0.64      0.49       285
```

[38]:
```
#Train - Test split 40-60
SVMClassifier(0.6, 'rbf', 3,)
SVMClassifier(0.6, 'linear', 3, )
SVMClassifier(0.6, 'poly', 2, 0.14)
SVMClassifier(0.6, 'sigmoid', 3, 0.2) #wrost performance
```

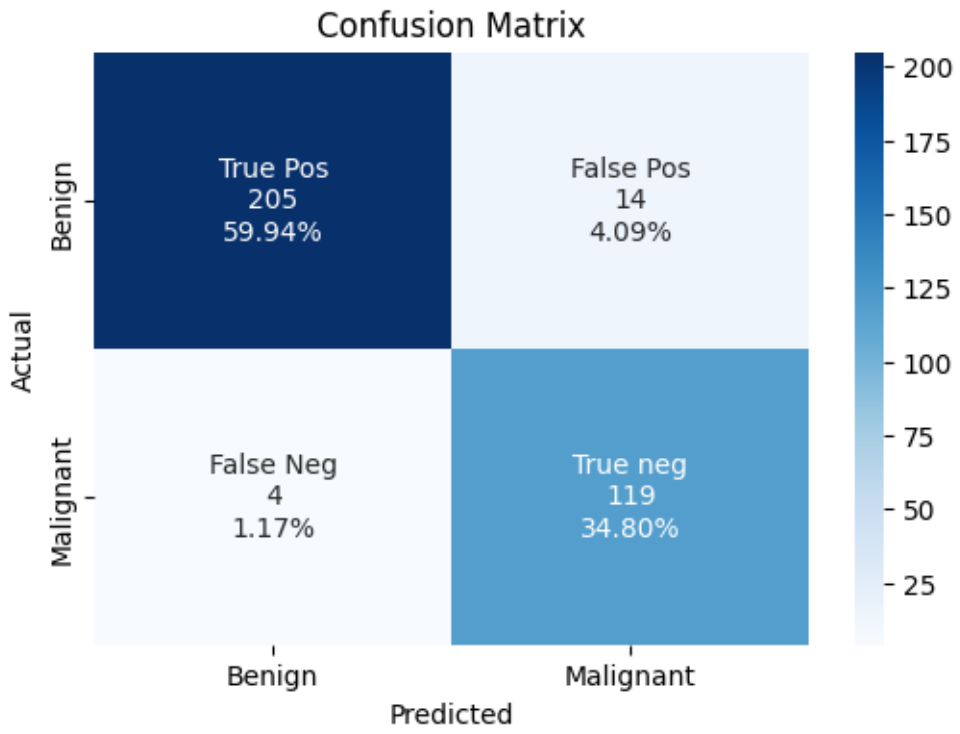
```
Confusion Matrix :
```



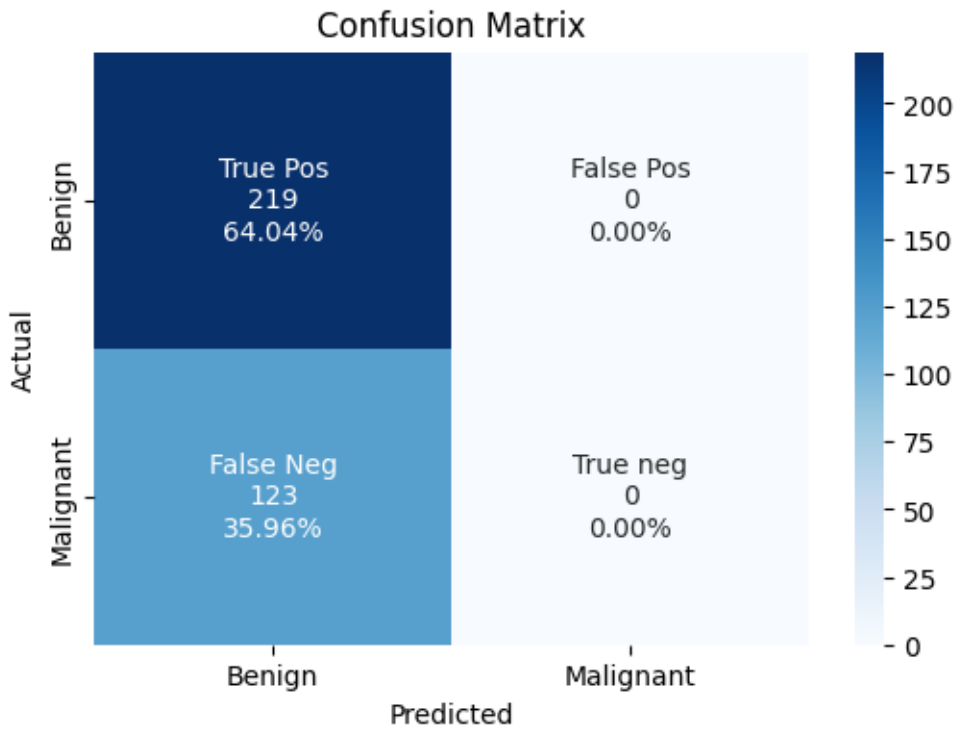

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.91      0.97      0.94       219
           1       0.94      0.83      0.88       123

    accuracy                           0.92       342
   macro avg       0.93      0.90      0.91       342
weighted avg       0.92      0.92      0.92       342

Confusion Matrix :
```




```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.96      0.96      0.96       219
           1       0.93      0.93      0.93       123

    accuracy                           0.95       342
   macro avg       0.95      0.95      0.95       342
weighted avg       0.95      0.95      0.95       342

Confusion Matrix :
```



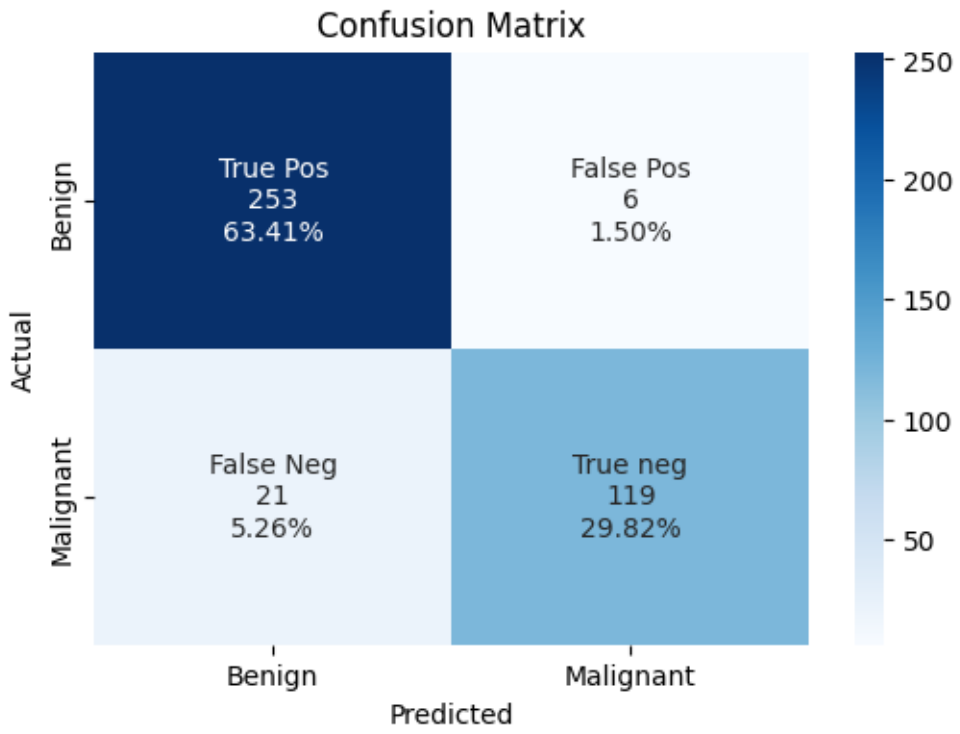

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.98      0.94      0.96       219
           1       0.89      0.97      0.93       123

    accuracy                           0.95       342
   macro avg       0.94      0.95      0.94       342
weighted avg       0.95      0.95      0.95       342

Confusion Matrix :
```



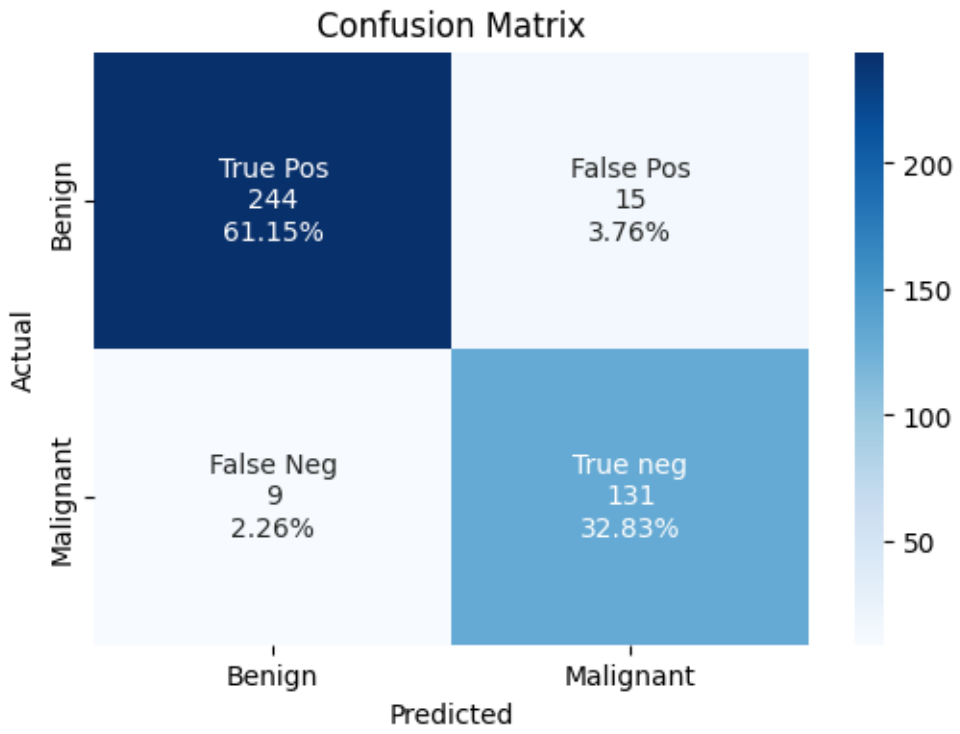

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.64      1.00      0.78       219
           1       0.00      0.00      0.00       123

    accuracy                           0.64       342
   macro avg       0.32      0.50      0.39       342
weighted avg       0.41      0.64      0.50       342
```

[39]:
```
#Train - Test split 30-70
SVMClassifier(0.7, 'rbf', 3,)
SVMClassifier(0.7, 'linear')
SVMClassifier(0.7, 'poly', 2,)
SVMClassifier(0.7, 'sigmoid', 3, 0.2 ) #wrost performance
```

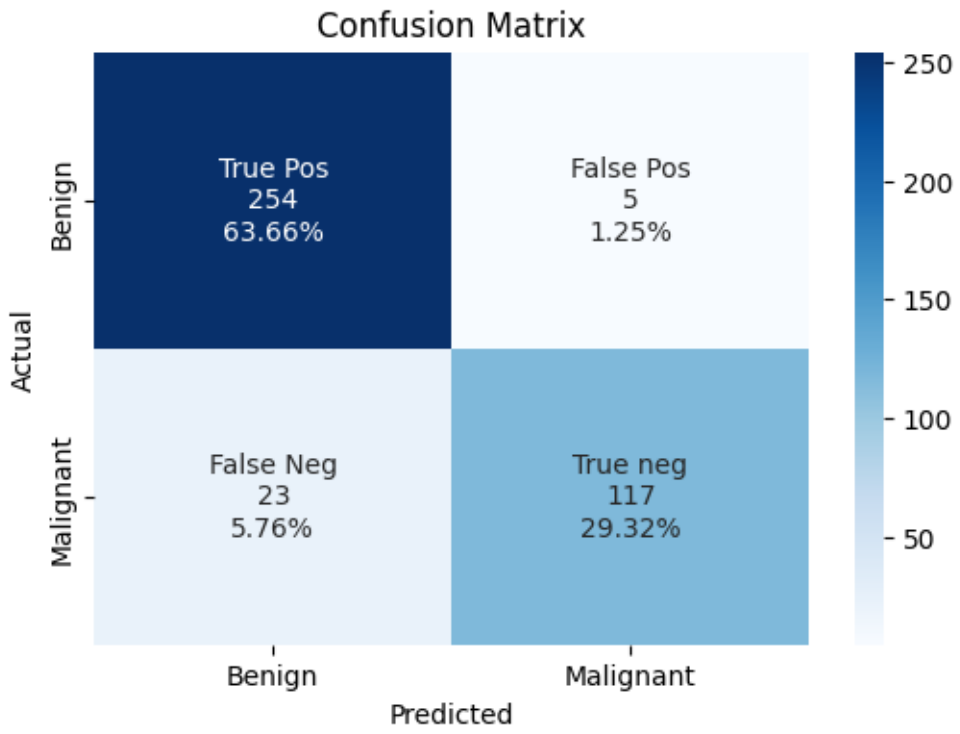
```
Confusion Matrix :
```



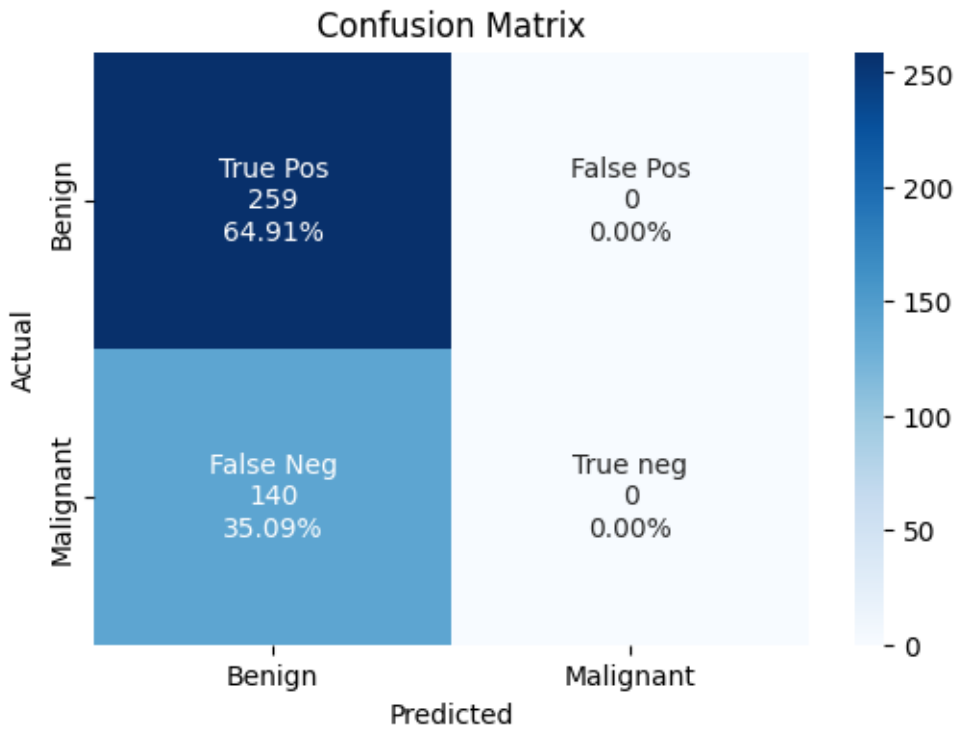

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.92      0.98      0.95       259
           1       0.95      0.85      0.90       140

    accuracy                           0.93       399
   macro avg       0.94      0.91      0.92       399
weighted avg       0.93      0.93      0.93       399

Confusion Matrix :
```




```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.96      0.94      0.95       259
           1       0.90      0.94      0.92       140

    accuracy                           0.94       399
   macro avg       0.93      0.94      0.93       399
weighted avg       0.94      0.94      0.94       399

Confusion Matrix :
```



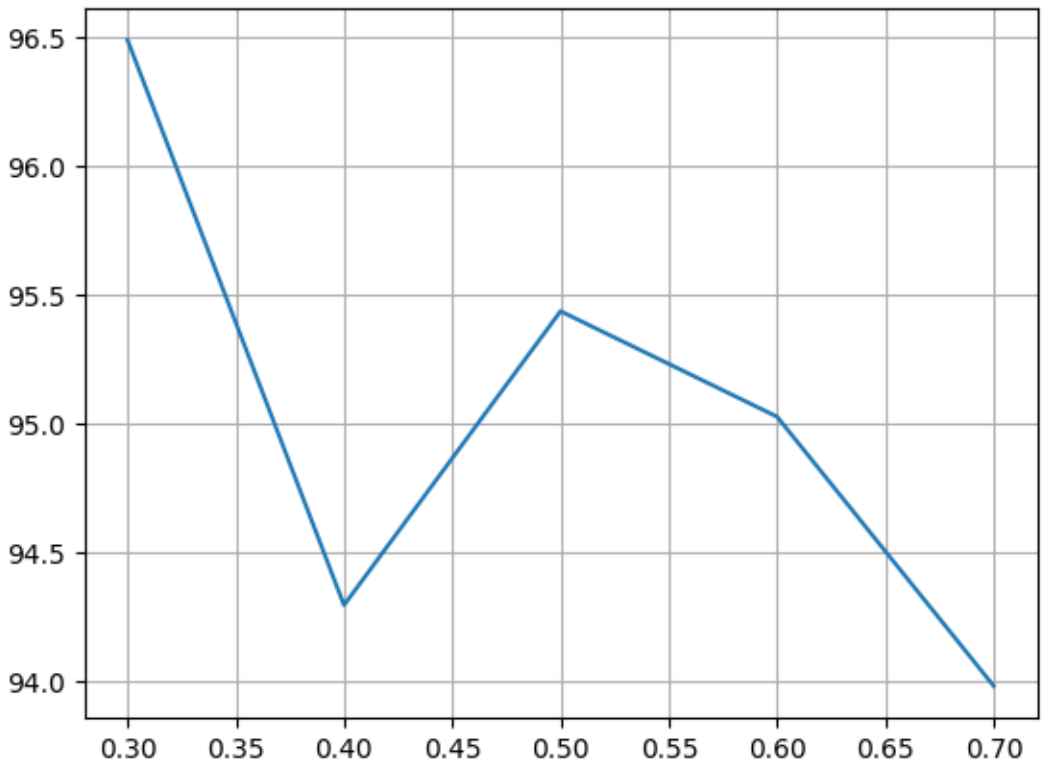

```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.92      0.98      0.95       259
           1       0.96      0.84      0.89       140

    accuracy                           0.93       399
   macro avg       0.94      0.91      0.92       399
weighted avg       0.93      0.93      0.93       399

Confusion Matrix :
```




```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.65      1.00      0.79       259
           1       0.00      0.00      0.00       140

    accuracy                           0.65       399
   macro avg       0.32      0.50      0.39       399
weighted avg       0.42      0.65      0.51       399
```

0.0.5 split vs accuracy graph

[40]:
```
x_points = [float(key) for key in dict_svm]
y_points = [i*100 for i in dict_svm.values()]
plt.plot(x_points, y_points)
plt.grid(True)
plt.show()
```



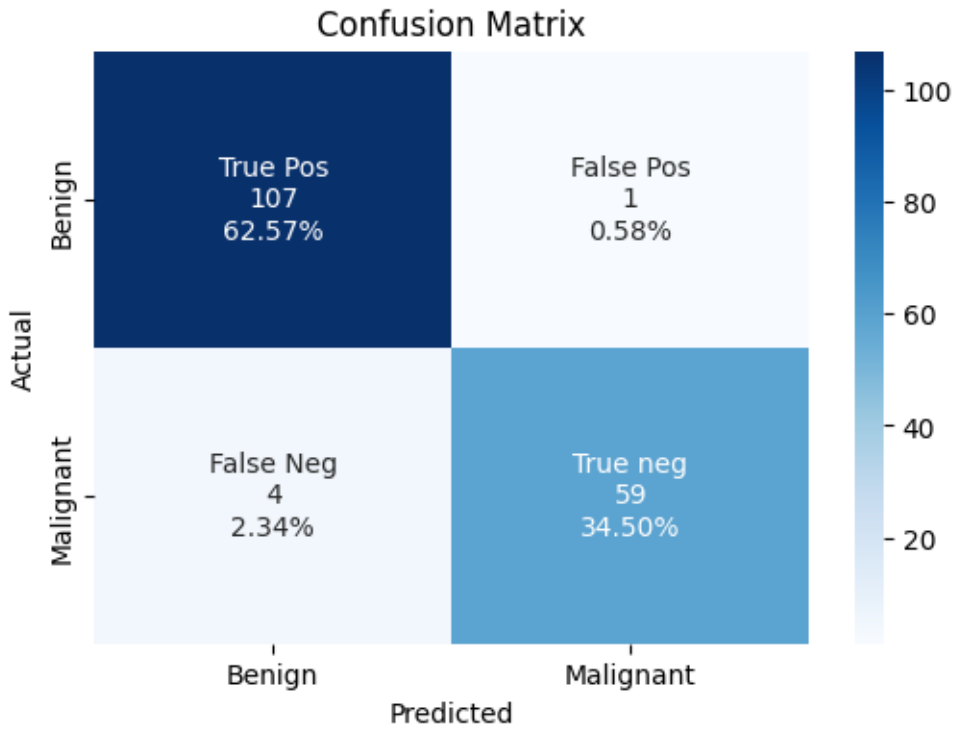
0.0.6 MLP Classifier

[41]:
```
def MLPClassifier(split, hiddenLayerSize = [100, ], activationValue = 'relu',␣
↪solverValue = 'adam'):
 from sklearn.model_selection import train_test_split
 from sklearn.neural_network import MLPClassifier
 from sklearn.metrics import accuracy_score
 from sklearn.preprocessing import StandardScaler
 scaler = StandardScaler()
 X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = split,␣
↪random_state=44)
 scaler.fit_transform(X_train)
 scaler.transform(X_test)
 classifier = MLPClassifier(hidden_layer_sizes = hiddenLayerSize, activation =␣
↪activationValue, solver = solverValue, random_state = 1)
 classifier.fit(X_train, y_train)
 y_pred = classifier.predict(X_test)
 accuracy = accuracy_score(y_test, y_pred)
 if(str(split) in dict_mlp):
   dict_mlp[str(split)] = max(accuracy, dict_mlp[str(split)])
   if(str(split) == '0.3' and accuracy > dict_svm[str(split)]):
     RocAucMlp['max'] = {'y_test': y_test, 'y_pred': y_pred}
```

```
  else:
    dict_mlp[str(split)] = accuracy
    RocAucMlp['max'] = {'y_test': y_test, 'y_pred': y_pred}

  reports(y_test, y_pred)
```

[42]:
```
#Train - Test split 70-30
MLPClassifier(0.3, [100, 60,])
```

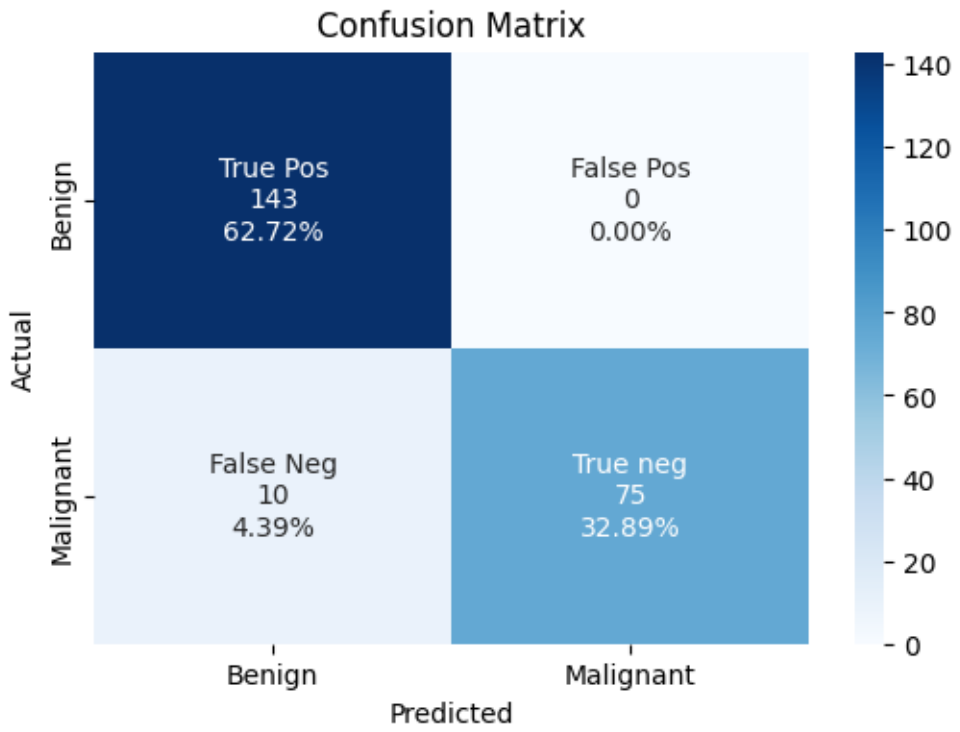
```
Confusion Matrix :
```




```
*********************************************
*********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.96      0.99      0.98       108
           1       0.98      0.94      0.96        63

    accuracy                           0.97       171
   macro avg       0.97      0.96      0.97       171
weighted avg       0.97      0.97      0.97       171
```

[43]:
```
#Train - Test split 60-40
MLPClassifier(0.4, [100, 66,])
```

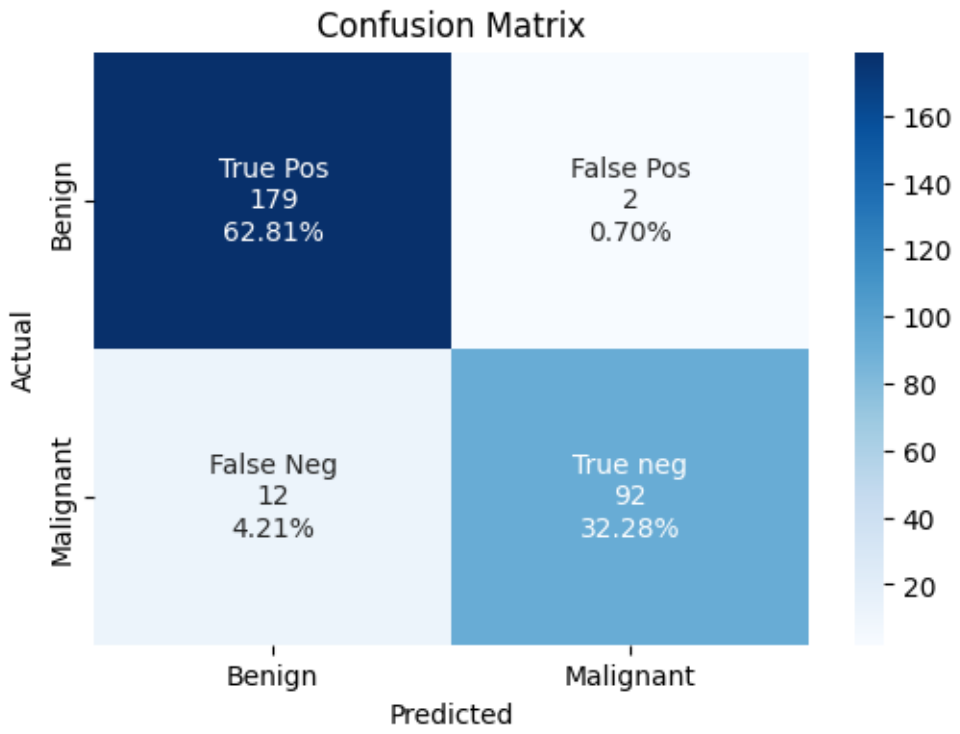
Confusion Matrix :




```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.93      1.00      0.97       143
           1       1.00      0.88      0.94        85

    accuracy                           0.96       228
   macro avg       0.97      0.94      0.95       228
weighted avg       0.96      0.96      0.96       228
```

[44]:
```
#Train - Test split 50-50
MLPClassifier(0.5, [150, 32])
```

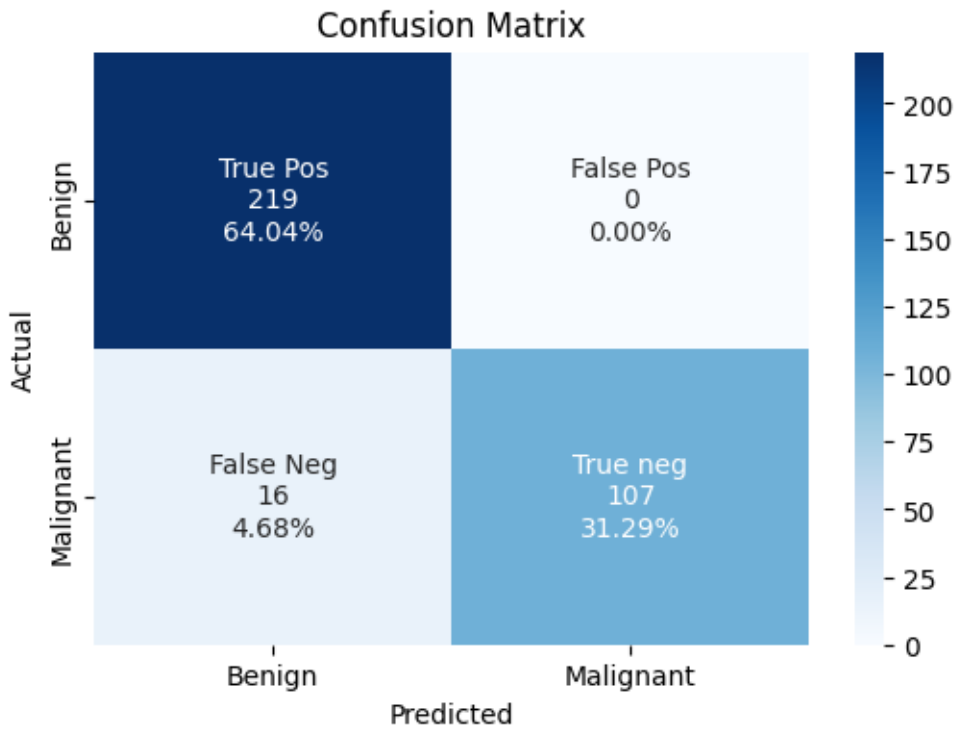
Confusion Matrix :




```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.94      0.99      0.96       181
           1       0.98      0.88      0.93       104

    accuracy                           0.95       285
   macro avg       0.96      0.94      0.95       285
weighted avg       0.95      0.95      0.95       285
```

[45]:
```
#Train - Test split 40-60
MLPClassifier(0.6, [150, 50])
```

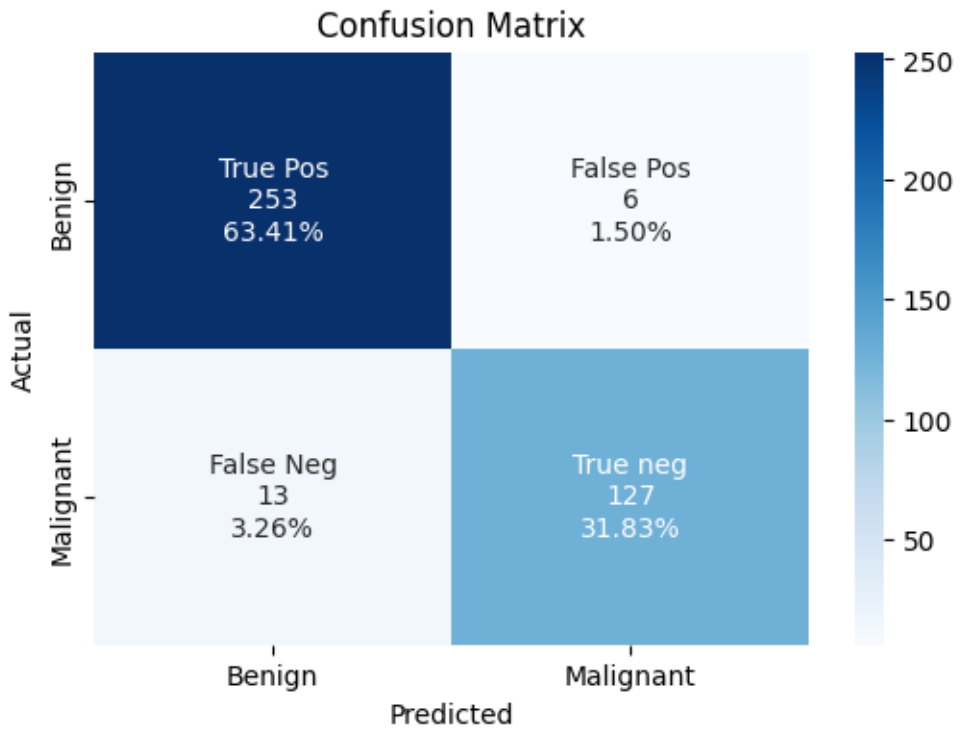
```
Confusion Matrix :
```




```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.93      1.00      0.96       219
           1       1.00      0.87      0.93       123

    accuracy                           0.95       342
   macro avg       0.97      0.93      0.95       342
weighted avg       0.96      0.95      0.95       342
```

[46]:
```
#Train - Test split 30-70
MLPClassifier(0.7, [100, 80])
```

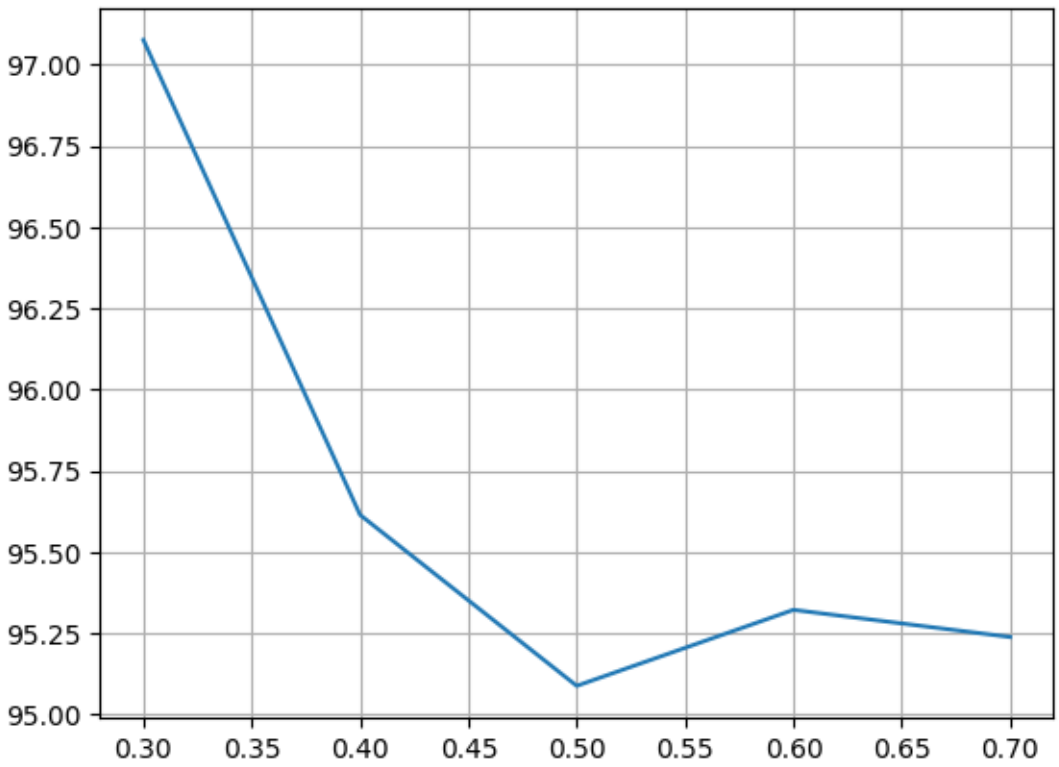
```
/home/aqeel/.local/lib/python3.10/site-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and
the optimization hasn't converged yet.
  warnings.warn(

Confusion Matrix :
```




```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.95      0.98      0.96       259
           1       0.95      0.91      0.93       140

    accuracy                           0.95       399
   macro avg       0.95      0.94      0.95       399
weighted avg       0.95      0.95      0.95       399
```

0.0.7 split vs accuracy graph

[47]:
```
x_points = [float(key) for key in dict_mlp]
y_points = [i*100 for i in dict_mlp.values()]
plt.plot(x_points, y_points)
plt.grid(True)
plt.show()
```



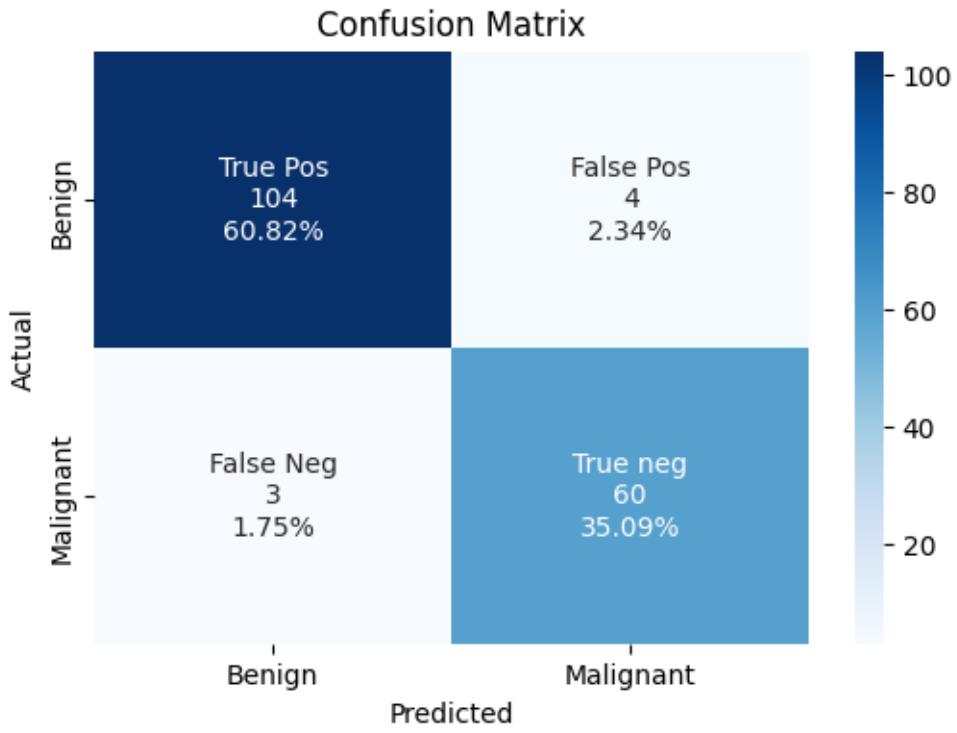
0.0.8 Random Forest Classifier

[48]:
```
def randomForest(split, estimator = 100, criterionValue = 'gini', ):
  from sklearn.model_selection import train_test_split
  from sklearn.ensemble import RandomForestClassifier
  from sklearn.metrics import accuracy_score
  from sklearn.preprocessing import StandardScaler
  scaler = StandardScaler()
  X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = split,␣
 ↪random_state=44)
  scaler.fit_transform(X_train)
  scaler.transform(X_test)
  classifier = RandomForestClassifier(n_estimators = estimator, criterion =␣
 ↪criterionValue)
  classifier.fit(X_train, y_train)
  y_pred = classifier.predict(X_test)
  accuracy = accuracy_score(y_test, y_pred)

  if(str(split) in dict_rfr):
    dict_rfr[str(split)] = max(accuracy, dict_rfr[str(split)])
    if(str(split) == '0.3' and accuracy > dict_svm[str(split)]):
      RocAucRfr['max'] = {'y_test': y_test, 'y_pred': y_pred}
```

```
  else:
    dict_rfr[str(split)] = accuracy
    if(str(split) == '0.3'):
      RocAucRfr['max'] = {'y_test': y_test, 'y_pred': y_pred}

  reports(y_test, y_pred)
```

[49]:
```
randomForest(0.3)
```

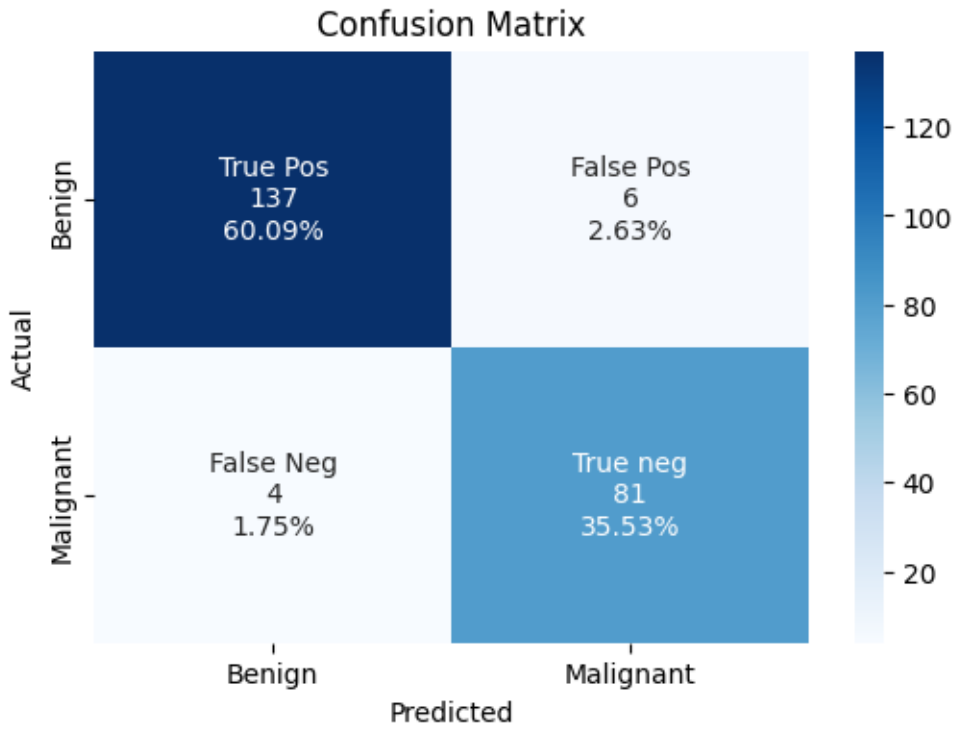
```
Confusion Matrix :
```




```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.97      0.96      0.97       108
           1       0.94      0.95      0.94        63

    accuracy                           0.96       171
   macro avg       0.95      0.96      0.96       171
weighted avg       0.96      0.96      0.96       171
```

[50]:
```
randomForest(0.4, 100,)
```

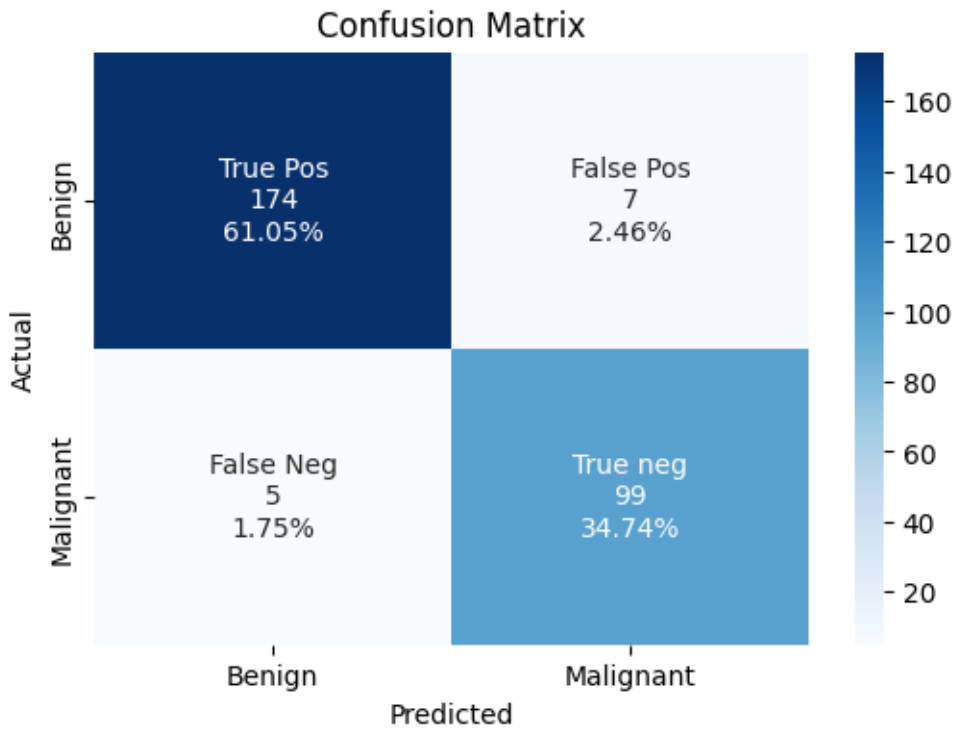
```
Confusion Matrix :
```




```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.97      0.96      0.96       143
           1       0.93      0.95      0.94        85

    accuracy                           0.96       228
   macro avg       0.95      0.96      0.95       228
weighted avg       0.96      0.96      0.96       228
```

[51]:
```
randomForest(0.5)
```

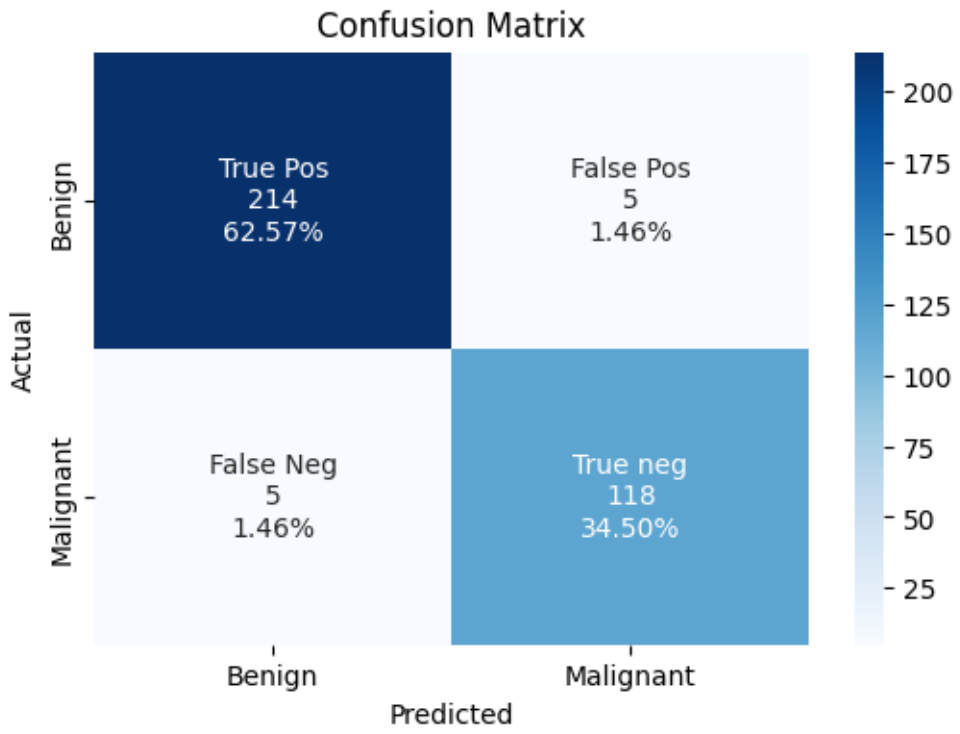
```
Confusion Matrix :
```




```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.97      0.96      0.97       181
           1       0.93      0.95      0.94       104

    accuracy                           0.96       285
   macro avg       0.95      0.96      0.95       285
weighted avg       0.96      0.96      0.96       285
```

[52]:
```
randomForest(0.6, 100, 'entropy')
```

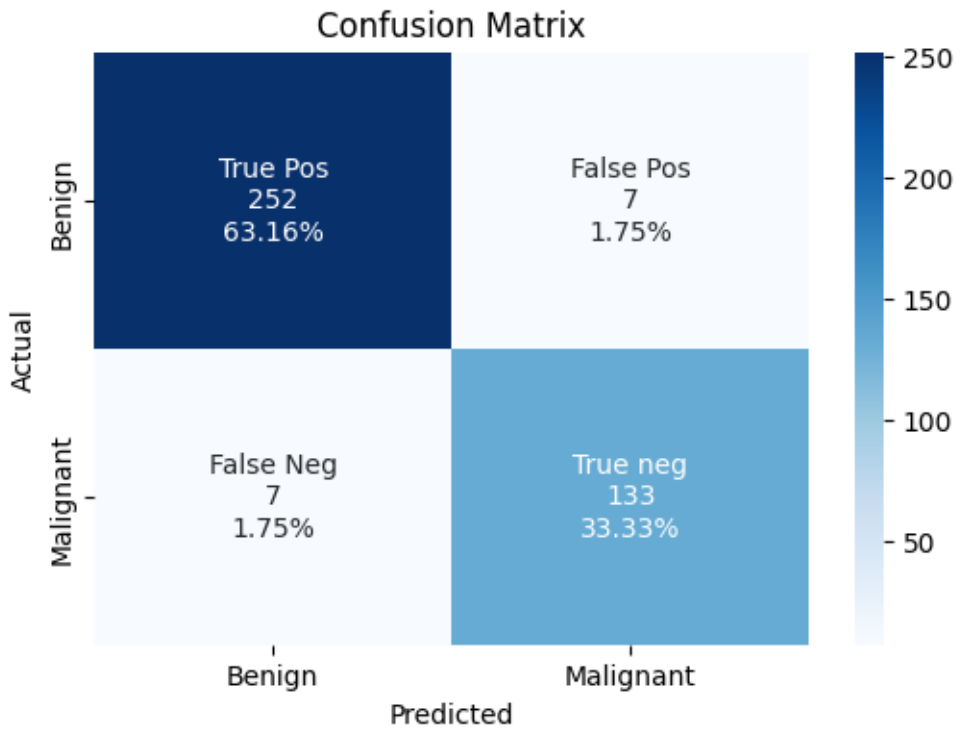
```
Confusion Matrix :
```




```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.98      0.98      0.98       219
           1       0.96      0.96      0.96       123

    accuracy                           0.97       342
   macro avg       0.97      0.97      0.97       342
weighted avg       0.97      0.97      0.97       342
```

[53]:
```
randomForest(0.7, 120)
```

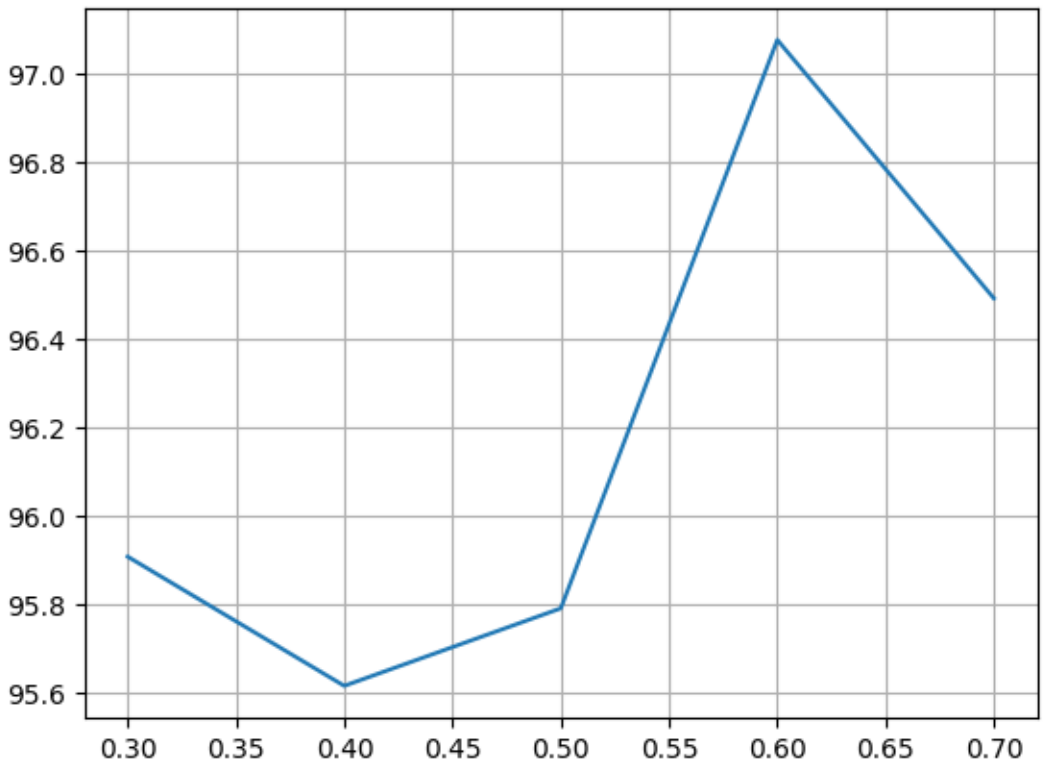
```
Confusion Matrix :
```




```
**********************************************
**********************************************
Classification Evaluation :
              precision    recall  f1-score   support

           0       0.97      0.97      0.97       259
           1       0.95      0.95      0.95       140

    accuracy                           0.96       399
   macro avg       0.96      0.96      0.96       399
weighted avg       0.96      0.96      0.96       399
```

0.0.9 split vs accuracy graph

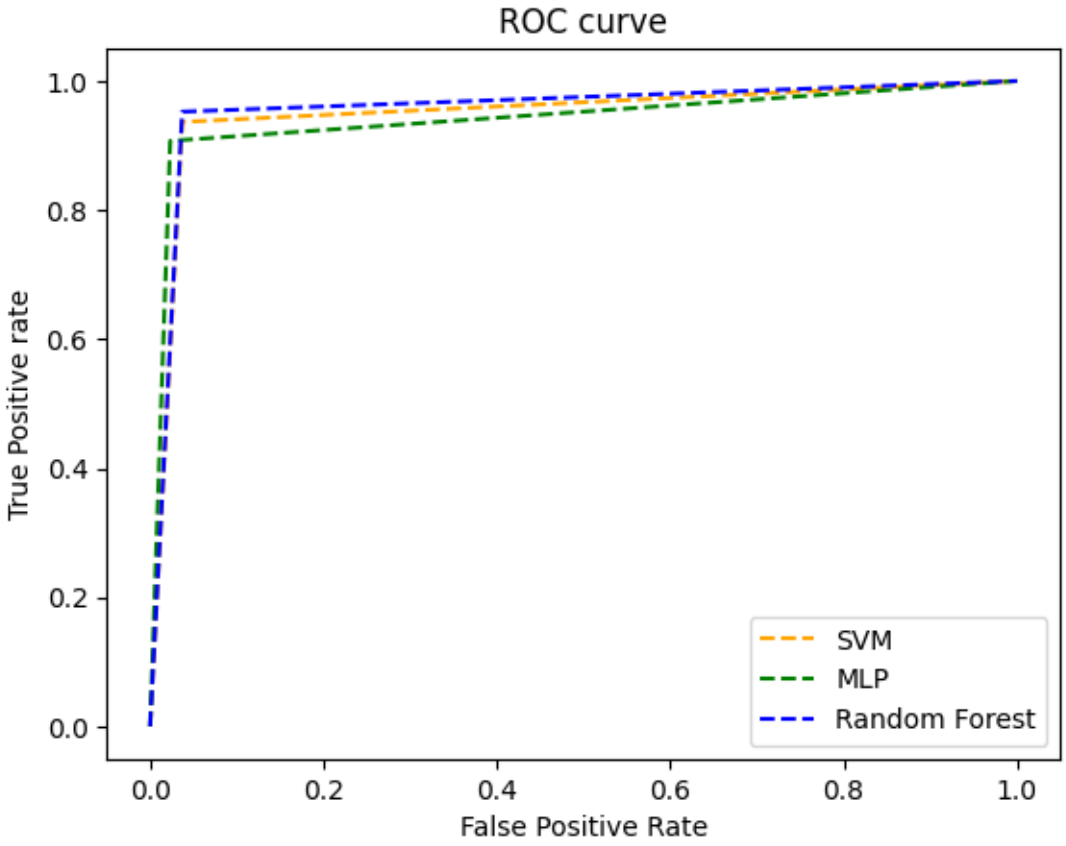
[54]:
```
x_points = [float(key) for key in dict_rfr]
y_points = [i*100 for i in dict_rfr.values()]
plt.plot(x_points, y_points)
plt.grid(True)
plt.show()
```



0.0.10 ROC curve and ROC_AUC score for all the classifier having maximum accuracy when train test split 70-30.

[55]:
```
from sklearn import metrics
def auc_roc():
    fpr1, tpr1, _1 = metrics.roc_curve(RocAucSvm['max']['y_test'],␣
 ↪RocAucSvm['max']['y_pred'], pos_label=1)
    fpr2, tpr2, _2 = metrics.roc_curve(RocAucMlp['max']['y_test'],␣
 ↪RocAucMlp['max']['y_pred'], pos_label=1)
    fpr3, tpr3, _3 = metrics.roc_curve(RocAucRfr['max']['y_test'],␣
 ↪RocAucRfr['max']['y_pred'], pos_label=1)
    plt.plot(fpr1, tpr1, linestyle='--',color='orange', label='SVM')
    plt.plot(fpr2, tpr2, linestyle='--',color='green', label='MLP')
    plt.plot(fpr3, tpr3, linestyle='--', color='blue', label= 'Random Forest')
    plt.title('ROC curve')
    # x label
    plt.xlabel('False Positive Rate')
    # y label
    plt.ylabel('True Positive rate')

    plt.legend(loc='best')
```

```
    plt.savefig('ROC',dpi=300)
    plt.show()
auc_roc()
```



0.0.11 Using PCA

[56]:
```
# from sklearn.datasets import load_breast_cancer
# dataset = load_breast_cancer()
# X = pd.DataFrame(data=dataset.data, columns=dataset.feature_names)
# y = dataset.target
# print(X)


# Standardizing the data (breast cancer dataset is already standardized)
##      from sklearn.preprocessing import StandardScaler
##      scaler = StandardScaler()
##      X_std = scaler.fit_transform(X)
##      print(X_std)
```

```
# Performing PCA
from sklearn.decomposition import PCA
number_of_components = 10  # Number of components to retain (your choice)
pca = PCA(n_components=number_of_components)
transformed_data = pca.fit_transform(X)
print(transformed_data.shape)


## choose train-test split or hyperparameters accordingly
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.3,␣
 ↪random_state=42)
rfc = RandomForestClassifier()
rfc.fit(X_train,y_train)
y_pred = rfc.predict(X_test)
print(classification_report(y_test,y_pred))
```

```
(569, 10)
              precision    recall  f1-score   support

           0       0.96      0.99      0.98       108
           1       0.98      0.94      0.96        63

    accuracy                           0.97       171
   macro avg       0.97      0.96      0.97       171
weighted avg       0.97      0.97      0.97       171
```

[ ]: