LogisticRegression:

```
print("The report:\n",clf_report)
```

```
The report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        32
           1       1.00      1.00      1.00        48

    accuracy                           1.00        80
   macro avg       1.00      1.00      1.00        80
weighted avg       1.00      1.00      1.00        80
```

```
#best parameters
from sklearn.metrics import f1_score
f1_macro=f1_score(y_test,grid_predictions,average='weighted')
print("The f1_macro value for best parameter {}:".format(grid.best_params_),f1_macro)
```

```
The f1_macro value for best parameter {'penalty': 'l2', 'solver': 'newton-cg'}: 1.0
```

```
from sklearn.metrics import roc_auc_score

roc_auc_score(y_test,grid.predict_proba(X_test)[:,1])
#probability estimates for each class for the input data.(such as logistic regression, random forests, and gradient boosting.)
#Column 0: Predicted probabilities of class 0 (negative class).
#Column 1: Predicted probabilities of class 1 (positive class).
#[:,1] --predicted probabilities for the positive class (y_pred_proba), not the class labels (y_pred).
```

```
1.0
```

SVC:

```
print("The report:\n",clf_reportc)
```

```
The report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        32
           1       1.00      1.00      1.00        48

    accuracy                           1.00        80
   macro avg       1.00      1.00      1.00        80
weighted avg       1.00      1.00      1.00        80
```

```
#from sklearn.metrics import roc_auc_score
#roc_auc_score(y_test,grid.predict_proba(X_test)[:,1])
from sklearn.metrics import roc_auc_score

roc_auc_score(y_test,gridc.predict_proba(X_test)[:,1])
##Column 1: Predicted probabilities of class 1 (positive class)
#AttributeError: predict_proba is not available when  probability=
#to fix use  grid = GridSearchCV(SVC(probability=True)   or #model
```

```
0.9993489583333334
```

```
from sklearn.metrics import f1_score
f1_macroc=f1_score(y_test,grid_predictions,average='weighted')
print("The f1_macro value for best parameter {}:".format(gridc.best_params_),f1_macro)
```

```
The f1_macro value for best parameter {'C': 10, 'gamma': 'auto', 'kernel': 'rbf'}: 1.0
```

DecisionTreeClassifier:

```
from sklearn.metrics import f1_score
f1_macro=f1_score(y_test,grid_predictionsd,average='weighted')
print("The f1_macro value for best parameter {}:".format(gridd.best_params_),f1_macro)
```

```
The f1_macro value for best parameter {'criterion': 'entropy', 'max_features': 'sqrt', 'splitter': 'random'}: 0.962734713529128
6
```

```
print("The report:\n",clf_reportd)
```

```
The report:
               precision    recall  f1-score   support

           0       0.91      1.00      0.96        32
           1       1.00      0.94      0.97        48

    accuracy                           0.96        80
   macro avg       0.96      0.97      0.96        80
weighted avg       0.97      0.96      0.96        80
```

```
from sklearn.metrics import roc_auc_score

roc_auc_score(y_test,gridd.predict_proba(X_test)[:,1])
```

```
0.96875
```

RandomForestClassifier:

```
from sklearn.metrics import f1_score
f1_macror=f1_score(y_test,grid_predictionsd,average='weighted')
print("The f1_macro value for best parameter {}:".format(gridr.best_params_),f1_macro)
```

```
The f1_macro value for best parameter {'criterion': 'entropy', 'max_features': 'sqrt', 'n_estimators': 100}: 0.9627347135291286
```

```
print("The report:\n",clf_reportr)
```

```
The report:
               precision    recall  f1-score   support

           0       1.00      0.97      0.98        32
           1       0.98      1.00      0.99        48

    accuracy                           0.99        80
   macro avg       0.99      0.98      0.99        80
weighted avg       0.99      0.99      0.99        80
```

```
from sklearn.metrics import roc_auc_score

roc_auc_score(y_test,gridr.predict_proba(X_test)[:,1])
```

```
1.0
```