OUTPUT SUMMARY :

|  |  |  |  |
| --- | --- | --- | --- |
| S.no | Classifier | Best parameters | Accuracy |
| 1 | SVM | {'C': 10, 'degree': 1, 'gamma': 0.001, 'kernel': 'rbf'} | 0.98 |
| 2 | Decision Tree | {'max\_depth': 50, 'max\_features': 'auto', 'min\_samples\_leaf': 4, 'min\_samples\_split': 10} | 0.80 |
| 3 | Neural Network | {'activation': 'relu', 'alpha': 0.01, 'early\_stopping': True, 'learning\_rate': 'invscaling', 'max\_iter': 100, 'solver': 'lbfgs'} | 0.97 |
| 4 | GaussianNB | {'priors': None} | 0.84 |
| 5 | LogisticRegression | {'C': 1, 'class\_weight': None, 'fit\_intercept': True, 'penalty': 'l2'} | 0.96 |
| 6 | KNeighborsClassifier | {'algorithm': 'auto', 'n\_neighbors': 1, 'p': 2, 'weights': 'uniform'} | 0.99 |

OUTPUT :

================Classifiers============

SVC(C=1.0, cache\_size=200, class\_weight=None, coef0=0.0,

decision\_function\_shape='ovr', degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, probability=False, random\_state=None, shrinking=True,

tol=0.001, verbose=False)

============Best Parameters=============

{'C': 10, 'degree': 1, 'gamma': 0.001, 'kernel': 'rbf'}

============Best Score=============

0.991833704529

==============Scores====================

[ 0.98905109 0.99261993 0.99261993 0.98501873 0.99621212]

==============Accuracy==================

Accuracy: 0.99 (+/- 0.01)

=============Confusion matrix==============

[[37 0 0 0 0 0 0 0 0 0]

[ 0 43 0 0 0 0 0 0 0 0]

[ 0 0 44 0 0 0 0 0 0 0]

[ 0 0 0 45 0 0 0 0 0 0]

[ 0 0 0 0 38 0 0 0 0 0]

[ 0 0 0 0 0 47 0 0 0 1]

[ 0 0 0 0 0 0 52 0 0 0]

[ 0 0 0 0 0 0 0 48 0 0]

[ 0 1 0 0 0 0 0 0 47 0]

[ 0 0 0 0 0 1 0 0 0 46]]

==========classification report=================

precision recall f1-score support

0 1.00 1.00 1.00 37

1 0.98 1.00 0.99 43

2 1.00 1.00 1.00 44

3 1.00 1.00 1.00 45

4 1.00 1.00 1.00 38

5 0.98 0.98 0.98 48

6 1.00 1.00 1.00 52

7 1.00 1.00 1.00 48

8 1.00 0.98 0.99 48

9 0.98 0.98 0.98 47

avg / total 0.99 0.99 0.99 450

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

================Classifiers============

DecisionTreeClassifier(class\_weight=None, criterion='gini', max\_depth=None,

max\_features=None, max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0, min\_impurity\_split=None,

min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0, presort=False, random\_state=None,

splitter='best')

============Best Parameters=============

{'max\_depth': 50, 'max\_features': 'auto', 'min\_samples\_leaf': 4, 'min\_samples\_split': 10}

============Best Score=============

0.804751299183

==============Scores====================

[ 0.82116788 0.81180812 0.8302583 0.7340824 0.75757576]

==============Accuracy==================

Accuracy: 0.79 (+/- 0.08)

=============Confusion matrix==============

[[36 0 0 0 0 1 0 0 0 0]

[ 0 35 4 3 1 0 0 0 0 0]

[ 1 1 34 4 2 0 0 0 2 0]

[ 0 2 1 38 0 1 0 0 0 3]

[ 0 2 0 0 26 1 3 5 1 0]

[ 1 0 1 2 1 33 3 0 2 5]

[ 2 0 1 1 0 2 42 0 4 0]

[ 1 1 0 1 3 1 1 34 5 1]

[ 0 5 4 2 1 0 1 1 34 0]

[ 0 0 1 6 0 2 1 0 2 35]]

==========classification report=================

precision recall f1-score support

0 0.88 0.97 0.92 37

1 0.76 0.81 0.79 43

2 0.74 0.77 0.76 44

3 0.67 0.84 0.75 45

4 0.76 0.68 0.72 38

5 0.80 0.69 0.74 48

6 0.82 0.81 0.82 52

7 0.85 0.71 0.77 48

8 0.68 0.71 0.69 48

9 0.80 0.74 0.77 47

avg / total 0.78 0.77 0.77 450

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

================Classifiers============

MLPClassifier(activation='relu', alpha=0.0001, batch\_size='auto', beta\_1=0.9,

beta\_2=0.999, early\_stopping=False, epsilon=1e-08,

hidden\_layer\_sizes=(100,), learning\_rate='constant',

learning\_rate\_init=0.001, max\_iter=200, momentum=0.9,

nesterovs\_momentum=True, power\_t=0.5, random\_state=None,

shuffle=True, solver='adam', tol=0.0001, validation\_fraction=0.1,

verbose=False, warm\_start=False)

============Best Parameters=============

{'activation': 'relu', 'alpha': 0.01, 'early\_stopping': True, 'learning\_rate': 'invscaling', 'max\_iter': 100, 'solver': 'lbfgs'}

============Best Score=============

0.974758723088

==============Scores====================

[ 0.98540146 0.97785978 0.96309963 0.96254682 0.98106061]

==============Accuracy==================

Accuracy: 0.97 (+/- 0.02)

=============Confusion matrix==============

[[37 0 0 0 0 0 0 0 0 0]

[ 1 41 0 0 0 0 1 0 0 0]

[ 0 0 44 0 0 0 0 0 0 0]

[ 0 0 1 43 0 1 0 0 0 0]

[ 0 0 0 0 37 0 0 1 0 0]

[ 0 0 0 0 0 47 0 0 0 1]

[ 0 1 0 0 0 0 51 0 0 0]

[ 0 0 0 0 1 0 0 46 0 1]

[ 0 3 1 0 0 1 0 0 43 0]

[ 0 0 0 0 0 1 0 0 0 46]]

==========classification report=================

precision recall f1-score support

0 0.97 1.00 0.99 37

1 0.91 0.95 0.93 43

2 0.96 1.00 0.98 44

3 1.00 0.96 0.98 45

4 0.97 0.97 0.97 38

5 0.94 0.98 0.96 48

6 0.98 0.98 0.98 52

7 0.98 0.96 0.97 48

8 1.00 0.90 0.95 48

9 0.96 0.98 0.97 47

avg / total 0.97 0.97 0.97 450

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

================Classifiers============

GaussianNB(priors=None)

============Best Parameters=============

{'priors': None}

============Best Score=============

0.835931700074

==============Scores====================

[ 0.84306569 0.83763838 0.85239852 0.84644195 0.84469697]

==============Accuracy==================

Accuracy: 0.84 (+/- 0.01)

=============Confusion matrix==============

[[37 0 0 0 0 0 0 0 0 0]

[ 0 39 0 0 0 0 0 0 4 0]

[ 0 7 20 2 0 0 0 0 15 0]

[ 0 0 0 39 0 0 0 1 5 0]

[ 0 1 0 0 31 0 0 6 0 0]

[ 0 1 0 1 0 43 0 3 0 0]

[ 0 0 1 0 0 0 51 0 0 0]

[ 0 0 0 0 1 0 0 47 0 0]

[ 0 6 0 1 0 1 0 2 38 0]

[ 0 2 0 4 1 0 0 3 7 30]]

==========classification report=================

precision recall f1-score support

0 1.00 1.00 1.00 37

1 0.70 0.91 0.79 43

2 0.95 0.45 0.62 44

3 0.83 0.87 0.85 45

4 0.94 0.82 0.87 38

5 0.98 0.90 0.93 48

6 1.00 0.98 0.99 52

7 0.76 0.98 0.85 48

8 0.55 0.79 0.65 48

9 1.00 0.64 0.78 47

avg / total 0.87 0.83 0.83 450

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

================Classifiers============

LogisticRegression(C=1.0, class\_weight=None, dual=False, fit\_intercept=True,

intercept\_scaling=1, max\_iter=100, multi\_class='ovr', n\_jobs=1,

penalty='l2', random\_state=None, solver='liblinear', tol=0.0001,

verbose=0, warm\_start=False)

============Best Parameters=============

{'C': 1, 'class\_weight': None, 'fit\_intercept': True, 'penalty': 'l2'}

============Best Score=============

0.956198960653

==============Scores====================

[ 0.94525547 0.96309963 0.95571956 0.96254682 0.96590909]

==============Accuracy==================

Accuracy: 0.96 (+/- 0.01)

=============Confusion matrix==============

[[37 0 0 0 0 0 0 0 0 0]

[ 0 39 0 0 0 0 2 0 2 0]

[ 0 0 41 3 0 0 0 0 0 0]

[ 0 0 1 43 0 0 0 0 0 1]

[ 0 0 0 0 38 0 0 0 0 0]

[ 0 1 0 0 0 47 0 0 0 0]

[ 0 0 0 0 0 0 52 0 0 0]

[ 0 1 0 1 1 0 0 45 0 0]

[ 0 3 1 0 0 0 0 0 43 1]

[ 0 0 0 1 0 1 0 0 1 44]]

==========classification report=================

precision recall f1-score support

0 1.00 1.00 1.00 37

1 0.89 0.91 0.90 43

2 0.95 0.93 0.94 44

3 0.90 0.96 0.92 45

4 0.97 1.00 0.99 38

5 0.98 0.98 0.98 48

6 0.96 1.00 0.98 52

7 1.00 0.94 0.97 48

8 0.93 0.90 0.91 48

9 0.96 0.94 0.95 47

avg / total 0.95 0.95 0.95 450

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

================Classifiers============

KNeighborsClassifier(algorithm='auto', leaf\_size=30, metric='minkowski',

metric\_params=None, n\_jobs=1, n\_neighbors=5, p=2,

weights='uniform')

============Best Parameters=============

{'algorithm': 'auto', 'n\_neighbors': 1, 'p': 2, 'weights': 'uniform'}

============Best Score=============

0.984409799555

==============Scores====================

[ 0.98540146 0.99630996 0.98523985 0.98876404 0.98863636]

==============Accuracy==================

Accuracy: 0.99 (+/- 0.01)

=============Confusion matrix==============

[[37 0 0 0 0 0 0 0 0 0]

[ 0 43 0 0 0 0 0 0 0 0]

[ 0 0 43 1 0 0 0 0 0 0]

[ 0 0 0 45 0 0 0 0 0 0]

[ 0 0 0 0 38 0 0 0 0 0]

[ 0 0 0 0 0 47 0 0 0 1]

[ 0 0 0 0 0 0 52 0 0 0]

[ 0 0 0 0 0 0 0 48 0 0]

[ 0 0 0 0 0 0 0 0 48 0]

[ 0 0 0 1 0 1 0 0 0 45]]

==========classification report=================

precision recall f1-score support

0 1.00 1.00 1.00 37

1 1.00 1.00 1.00 43

2 1.00 0.98 0.99 44

3 0.96 1.00 0.98 45

4 1.00 1.00 1.00 38

5 0.98 0.98 0.98 48

6 1.00 1.00 1.00 52

7 1.00 1.00 1.00 48

8 1.00 1.00 1.00 48

9 0.98 0.96 0.97 47

avg / total 0.99 0.99 0.99 450

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

KNeighborsClassifier(algorithm='auto', leaf\_size=30, metric='minkowski',

metric\_params=None, n\_jobs=1, n\_neighbors=1, p=2,

weights='uniform')