scikit-learn

Evaluate several classifiers on the Wine dataset using scikit-learn

Ratio 60-40

Svm

In [1]:

```
from sklearn import svm
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split, cross_val_score

# Loading the dataset
x, y = load_wine(return_X_y=True)

# Splitting the dataset into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=0, train_size=0.6)

# Support Vector Machine (SVM)
clf_svm = svm.SVC()
clf_svm.fit(x_train, y_train)

# Accuracy Scores
svm_acc_test = clf_svm.score(x_test, y_test)
svm_acc_train = clf_svm.score(x_train, y_train)

# Cross-Validation
cv_svm = cross_val_score(clf_svm, x, y, cv=5)
cv_svm3 = cross_val_score(clf_svm, x, y, cv=3)
cv_svm7 = cross_val_score(clf_svm, x, y, cv=7)
# Printing Results
print("SVM Accuracy on Test Set:", svm_acc_test)
print("SVM Accuracy on Training Set:", svm_acc_train)
print("SVM Cross-Validation Scores:", cv_svm)
print("------------------------------------------------------------")
print("SVM Cross-Validation Scores:", cv_svm3.mean())
print("------------------------------------------------------------")
print("SVM Cross-Validation Scores:", cv_svm7.mean())
print("------------------------------------------------------------")
print("SVM Cross-Validation Scores percentage :", cv_svm * 100)
print("SVM Cross-Validation Mean Score:", cv_svm.mean())
print("SVM Cross-Validation Standard Deviation:", cv_svm.std())
print("------------------------------------------------------------")
print("SVM Cross-Validation Standard Deviation:", cv_svm3.std())
print("------------------------------------------------------------")
print("SVM Cross-Validation Standard Deviation:", cv_svm7.std())
```

```
SVM Accuracy on Test Set: 0.6805555555555556
SVM Accuracy on Training Set: 0.6698113207547169
SVM Cross-Validation Scores: [0.63888889 0.61111111 0.63888889 0.68571429 0.74285714]
------------------------------------------------------------
SVM Cross-Validation Scores: 0.657532956685499
------------------------------------------------------------
SVM Cross-Validation Scores: 0.6525274725274725
------------------------------------------------------------
SVM Cross-Validation Scores percentage : [63.88888889 61.11111111 63.88888889 68.57142857 74.28571429]
SVM Cross-Validation Mean Score: 0.6634920634920635
SVM Cross-Validation Standard Deviation: 0.04636170738133653
------------------------------------------------------------
SVM Cross-Validation Standard Deviation: 0.032039704709751565
------------------------------------------------------------
SVM Cross-Validation Standard Deviation: 0.06582041486989956
```

Decision Tree Classifier

In [2]:

```
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.tree import DecisionTreeClassifier

# Loading the dataset
x, y = load_wine(return_X_y=True)

# Splitting the dataset into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=0, train_size=0.6)

# Decision Tree
dt = DecisionTreeClassifier()
dt.fit(x_train, y_train)

# Accuracy Scores
dt_acc_test = dt.score(x_test, y_test)
dt_acc_train = dt.score(x_train, y_train)

# Cross-Validation
cv_dt = cross_val_score(dt, x, y, cv=5)
cv_dt3 = cross_val_score(dt, x, y, cv=3)
cv_dt7= cross_val_score(dt, x, y, cv=7)


# Printing Results
print("Decision Tree Accuracy on Test Set:", dt_acc_test)
print("Decision Tree Accuracy on Training Set:", dt_acc_train)
print("Decision Tree Cross-Validation Scores:", cv_dt)
print("------------------------------------------------------------")
print("Decision Tree Cross-Validation Scores:", cv_dt3.mean())
print("------------------------------------------------------------")
print("Decision Tree Cross-Validation Scores:", cv_dt7.mean())
print("------------------------------------------------------------")
print("Decision Tree-Validation Scores percentage :", cv_svm * 100)
print("Decision Tree-Validation Mean Score:", cv_dt.mean())
print("Decision Tree Cross-Validation Standard Deviation:", cv_dt.std())
print("------------------------------------------------------------")
print("Decision Tree-Validation Standard Deviation:", cv_dt3.std())
print("------------------------------------------------------------")
print("Decision Tree-Validation Standard Deviation:", cv_dt7.std())
```

```
Decision Tree Accuracy on Test Set: 0.9583333333333334
Decision Tree Accuracy on Training Set: 1.0
Decision Tree Cross-Validation Scores: [0.94444444 0.83333333 0.86111111 0.91428571 0.85714286]
------------------------------------------------------------
Decision Tree Cross-Validation Scores: 0.9042372881355932
------------------------------------------------------------
Decision Tree Cross-Validation Scores: 0.8824175824175823
------------------------------------------------------------
Decision Tree-Validation Scores percentage : [63.88888889 61.11111111 63.88888889 68.57142857 74.28571429]
Decision Tree-Validation Mean Score: 0.882063492063492
Decision Tree Cross-Validation Standard Deviation: 0.0409006687162463
------------------------------------------------------------
Decision Tree-Validation Standard Deviation: 0.05263796662125264
------------------------------------------------------------
Decision Tree-Validation Standard Deviation: 0.05878371113739871
```

Random Forest Classifier

In [3]:

```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split, cross_val_score

# Loading the dataset
x, y = load_wine(return_X_y=True)

# Splitting the dataset into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=0, train_size=0.6)

# Random Forest
rf = RandomForestClassifier()
rf.fit(x_train, y_train)

# Accuracy Scores
rf_acc_test = rf.score(x_test, y_test)
rf_acc_train = rf.score(x_train, y_train)

# Cross-Validation
cv_rf = cross_val_score(rf, x, y, cv=5)
cv_rf3 = cross_val_score(rf, x, y, cv=3)
cv_rf7 = cross_val_score(rf, x, y, cv=7)
# Printing Results
print("Random Forest Accuracy on Test Set:", rf_acc_test)
print("Random Forest Accuracy on Training Set:", rf_acc_train)
print("Random Forest Cross-Validation Scores:", cv_rf)
print("Random Forest -Validation Scores percentage :", cv_rf * 100)
print("Random Forest -Validation Mean Score:", cv_rf.mean())
print("-----------------------------------------------------------")
print("Random Forest -Validation Scores:", cv_rf3.mean())
print("-----------------------------------------------------------")
print("Random Forest -Validation Scores:", cv_rf7.mean())
print("-----------------------------------------------------------")
print("Random Forest Cross-Validation Standard Deviation:", cv_rf.std())
print("-----------------------------------------------------------")
print("Random Forest Cross-Validation Standard Deviation:", cv_rf3.std())
print("-----------------------------------------------------------")
print("Random Forest -Validation Standard Deviation:", cv_rf7.std())
```

```
Random Forest Accuracy on Test Set: 0.9583333333333334
Random Forest Accuracy on Training Set: 1.0
Random Forest Cross-Validation Scores: [0.97222222 0.94444444 1.         0.97142857 1.        ]
Random Forest -Validation Scores percentage : [ 97.22222222  94.44444444 100.          97.14285714 100.        ]
Random Forest -Validation Mean Score: 0.9776190476190475
----------------------------------------------------------
Random Forest -Validation Scores: 0.9494350282485876
----------------------------------------------------------
Random Forest -Validation Scores: 0.9725274725274725
----------------------------------------------------------
Random Forest Cross-Validation Standard Deviation: 0.020831783767013237
----------------------------------------------------------
Random Forest Cross-Validation Standard Deviation: 0.027680733160189507
----------------------------------------------------------
Random Forest -Validation Standard Deviation: 0.039621442587516376
```

Logistic Regression

In [4]:

```python
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split, cross_val_score

# Loading the dataset
x, y = load_wine(return_X_y=True)

# Splitting the dataset into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=0, train_size=0.6)

# Logistic Regression
lr = LogisticRegression()
lr.fit(x_train, y_train)

# Accuracy Scores
lr_acc_test = lr.score(x_test, y_test)
lr_acc_train = lr.score(x_train, y_train)

# Cross-Validation
cv_lr = cross_val_score(lr, x, y, cv=5)
cv_lr3 = cross_val_score(lr, x, y, cv=3)
cv_lr7 = cross_val_score(lr, x, y, cv=7)
# Printing Results
print("Logistic Regression Accuracy on Test Set:", lr_acc_test)
print("Logistic Regression Accuracy on Training Set:", lr_acc_train)
print("Logistic Regression Cross-Validation Scores:", cv_lr)
print("Logistic Regression -Validation Mean Score:", cv_lr.mean())
print("-----------------------------------------------------------")
print("Logistic Regression -Validation Scores:", cv_lr3.mean())
print("-----------------------------------------------------------")
print("Logistic Regression -Validation Scores:", cv_lr7.mean())
print("-----------------------------------------------------------")
print("Logistic Regression Cross-Validation Standard Deviation:", cv_lr.std())
print("-----------------------------------------------------------")
print("Logistic Regression Cross-Validation Standard Deviation:", cv_lr3.std())
print("-----------------------------------------------------------")
print("Logistic Regression -Validation Standard Deviation:", cv_lr7.std())
```

```
C:\Users\Arooj\AppData\Roaming\Python\Python311\site-packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfg
s failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\Arooj\AppData\Roaming\Python\Python311\site-packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfg
s failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\Arooj\AppData\Roaming\Python\Python311\site-packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfg
s failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\Arooj\AppData\Roaming\Python\Python311\site-packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfg
s failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\Arooj\AppData\Roaming\Python\Python311\site-packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfg
s failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
```

```
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\Arooj\AppData\Roaming\Python\Python311\site-packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfg
s failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\Arooj\AppData\Roaming\Python\Python311\site-packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfg
s failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\Arooj\AppData\Roaming\Python\Python311\site-packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfg
s failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\Arooj\AppData\Roaming\Python\Python311\site-packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfg
s failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\Arooj\AppData\Roaming\Python\Python311\site-packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfg
s failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

```
Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\Arooj\AppData\Roaming\Python\Python311\site-packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfg
s failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
Logistic Regression Accuracy on Test Set: 0.9305555555555556
Logistic Regression Accuracy on Training Set: 0.9905660377358491
Logistic Regression Cross-Validation Scores: [0.88888889 0.94444444 0.94444444 1.         1.        ]
Logistic Regression -Validation Mean Score: 0.9555555555555555
------------------------------------------------------------
Logistic Regression -Validation Scores: 0.9052730696798493
------------------------------------------------------------
Logistic Regression -Validation Scores: 0.945054945054945
------------------------------------------------------------
Logistic Regression Cross-Validation Standard Deviation: 0.041573970964154924
------------------------------------------------------------
Logistic Regression Cross-Validation Standard Deviation: 0.1001837219112134
------------------------------------------------------------
Logistic Regression -Validation Standard Deviation: 0.06454582485972929
```

```
C:\Users\Arooj\AppData\Roaming\Python\Python311\site-packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfg
s failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\Arooj\AppData\Roaming\Python\Python311\site-packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfg
s failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\Arooj\AppData\Roaming\Python\Python311\site-packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfg
s failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\Arooj\AppData\Roaming\Python\Python311\site-packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfg
s failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\Arooj\AppData\Roaming\Python\Python311\site-packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfg
s failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
```

```
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

GaussianNB

In [11]:

```
from sklearn.naive_bayes import GaussianNB
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split, cross_val_score

# Loading the dataset
x, y = load_wine(return_X_y=True)

# Splitting the dataset into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=0, train_size=0.6)

# Naive Bayes
nb = GaussianNB()
nb.fit(x_train, y_train)

# Accuracy Scores
nb_acc_test = nb.score(x_test, y_test)
nb_acc_train = nb.score(x_train, y_train)

# Cross-Validation
cv_nb = cross_val_score(nb, x, y, cv=5)
cv_nb3 = cross_val_score(nb, x, y, cv=3)
cv_nb7 = cross_val_score(nb, x, y, cv=7)

# Printing Results
print("Naive Bayes Accuracy on Test Set:", nb_acc_test)
print("Naive Bayes Accuracy on Training Set:", nb_acc_train)
print("Naive Bayes Cross-Validation Scores:", cv_nb)
print("Naive Bayes -Validation Mean Score:", cv_nb.mean())
print("-------------------------------------------------------------")
print("Naive Bayes -Validation mean Scores:", cv_nb3.mean())
print("-------------------------------------------------------------")
print("Naive Bayes -Validation mean Scores:", cv_nb7.mean())
print("-------------------------------------------------------------")
print("Naive Bayes Cross-Validation Standard Deviation:", cv_nb.std())
print("-------------------------------------------------------------")
print("Naive Bayes Cross-Validation Standard Deviation:", cv_nb3.std())
print("-------------------------------------------------------------")
print("Naive Bayes-Validation Standard Deviation:", cv_nb7.std())
```

```
Naive Bayes Accuracy on Test Set: 0.9444444444444444
Naive Bayes Accuracy on Training Set: 0.9905660377358491
Naive Bayes Cross-Validation Scores: [0.94444444 0.97222222 0.97222222 0.94285714 1.        ]
Naive Bayes -Validation Mean Score: 0.9663492063492063
------------------------------------------------------------
Naive Bayes -Validation mean Scores: 0.9607344632768361
------------------------------------------------------------
Naive Bayes -Validation mean Scores: 0.9613186813186813
------------------------------------------------------------
Naive Bayes Cross-Validation Standard Deviation: 0.02113317858457236
------------------------------------------------------------
Naive Bayes Cross-Validation Standard Deviation: 0.007590411775448826
------------------------------------------------------------
Naive Bayes-Validation Standard Deviation: 0.03561253468224893
```

K Neighbors Classifier

In [7]:

```python
from sklearn.neighbors import KNeighborsClassifier
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split, cross_val_score

# Loading the dataset
x, y = load_wine(return_X_y=True)

# Splitting the dataset into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=0, train_size=0.6)

# K-Nearest Neighbors (KNN)
knn = KNeighborsClassifier()
knn.fit(x_train, y_train)

# Accuracy Scores
knn_acc_test = knn.score(x_test, y_test)
knn_acc_train = knn.score(x_train, y_train)

# Cross-Validation
cv_knn = cross_val_score(knn, x, y, cv=5)
cv_knn3 = cross_val_score(knn, x, y, cv=3)
cv_knn7 = cross_val_score(knn, x, y, cv=7)


# Printing Results
print("K-Nearest Neighbors Accuracy on Test Set:", knn_acc_test)
print("K-Nearest Neighbors Accuracy on Training Set:", knn_acc_train)
print("K-Nearest Neighbors Cross-Validation Scores:", cv_knn)
print("K-Nearest Neighbors-Validation Mean Score:", cv_knn.mean())
print("-------------------------------------------------------------")
print("K-Nearest Neighbors-Validation mean Scores:", cv_knn3.mean())
print("-------------------------------------------------------------")
print("K-Nearest Neighbors-Validation mean Scores:", cv_knn7.mean())
print("-------------------------------------------------------------")
print("K-Nearest Neighbors-Validation Standard Deviation:", cv_knn.std())
print("-------------------------------------------------------------")
print("K-Nearest Neighbors Cross-Validation Standard Deviation:", cv_knn3.std())
print("-------------------------------------------------------------")
print("Naive Bayes-Validation Standard Deviation:", cv_knn7.std())
```

```
K-Nearest Neighbors Accuracy on Test Set: 0.6666666666666666
K-Nearest Neighbors Accuracy on Training Set: 0.8018867924528302
K-Nearest Neighbors Cross-Validation Scores: [0.72222222 0.66666667 0.63888889 0.65714286 0.77142857]
```

AdaBoost Classifier

In [ ]:

```python
from sklearn.ensemble import AdaBoostClassifier
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split, cross_val_score

# Loading the dataset
x, y = load_wine(return_X_y=True)

# Splitting the dataset into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=0, train_size=0.6)

# AdaBoost
adaboost = AdaBoostClassifier()
adaboost.fit(x_train, y_train)

# Accuracy Scores
adaboost_acc_test = adaboost.score(x_test, y_test)
adaboost_acc_train = adaboost.score(x_train, y_train)

# Cross-Validation
cv_adaboost = cross_val_score(adaboost, x, y, cv=5)

# Printing Results
print("AdaBoost Accuracy on Test Set:", adaboost_acc_test)
print("AdaBoost Accuracy on Training Set:", adaboost_acc_train)
print("AdaBoost Cross-Validation Scores:", cv_adaboost)
print("AdaBoost Cross-Validation Mean Score:", cv_adaboost.mean())
print("AdaBoost Cross-Validation Standard Deviation:", cv_adaboost.std())
```

Ratio 70-30

SVM

In [ ]:
```
from sklearn.svm import SVC
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split, cross_val_score

# Loading the dataset
x, y = load_wine(return_X_y=True)

# Splitting the dataset into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=0, train_size=0.7)

# Support Vector Machine (SVM)
clf_svm = SVC()
clf_svm.fit(x_train, y_train)

# Accuracy Scores
svm_acc_test = clf_svm.score(x_test, y_test)
svm_acc_train = clf_svm.score(x_train, y_train)

# Cross-Validation
cv_svm = cross_val_score(clf_svm, x, y, cv=5)

# Printing Results
print("SVM Accuracy on Test Set:", svm_acc_test)
print("SVM Accuracy on Training Set:", svm_acc_train)
print("SVM Cross-Validation Scores:", cv_svm)
print("SVM Cross-Validation Scores percentage :", cv_svm * 100)
print("SVM Cross-Validation Mean Score:", cv_svm.mean())
print("SVM Cross-Validation Standard Deviation:", cv_svm.std())
```

Decision Tree Classifier

In [ ]:

```
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.tree import DecisionTreeClassifier

# Loading the dataset
x, y = load_wine(return_X_y=True)

# Splitting the dataset into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=0, train_size=0.7)

# Decision Tree
dt = DecisionTreeClassifier()
dt.fit(x_train, y_train)

# Accuracy Scores
dt_acc_test = dt.score(x_test, y_test)
dt_acc_train = dt.score(x_train, y_train)

# Cross-Validation
cv_dt = cross_val_score(dt, x, y, cv=5)

# Printing Results
print("Decision Tree Accuracy on Test Set:", dt_acc_test)
print("Decision Tree Accuracy on Training Set:", dt_acc_train)
print("Decision Tree Cross-Validation Scores:", cv_dt)
print("Decision Tree Cross-Validation Mean Score:", cv_dt.mean())
print("Decision Tree Cross-Validation Standard Deviation:", cv_dt.std())
```

Logistic Regression

In [ ]:

```python
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split, cross_val_score

# Loading the dataset
x, y = load_wine(return_X_y=True)

# Splitting the dataset into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=0, train_size=0.7)

# Logistic Regression
lr = LogisticRegression()
lr.fit(x_train, y_train)

# Accuracy Scores
lr_acc_test = lr.score(x_test, y_test)
lr_acc_train = lr.score(x_train, y_train)

# Cross-Validation
cv_lr = cross_val_score(lr, x, y, cv=5)

# Printing Results
print("Logistic Regression Accuracy on Test Set:", lr_acc_test)
print("Logistic Regression Accuracy on Training Set:", lr_acc_train)
print("Logistic Regression Cross-Validation Scores:", cv_lr)
print("Logistic Regression Cross-Validation Mean Score:", cv_lr.mean())
print("Logistic Regression Cross-Validation Standard Deviation:", cv_lr.std())
```

Random Forest Classifier

In [ ]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split, cross_val_score

# Loading the dataset
x, y = load_wine(return_X_y=True)

# Splitting the dataset into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=0, train_size=0.7)

# Random Forest
rf = RandomForestClassifier()
rf.fit(x_train, y_train)

# Accuracy Scores
rf_acc_test = rf.score(x_test, y_test)
rf_acc_train = rf.score(x_train, y_train)

# Cross-Validation
cv_rf = cross_val_score(rf, x, y, cv=5)

# Printing Results
print("Random Forest Accuracy on Test Set:", rf_acc_test)
print("Random Forest Accuracy on Training Set:", rf_acc_train)
print("Random Forest Cross-Validation Scores:", cv_rf)
print("Random Forest Cross-Validation Mean Score:", cv_rf.mean())
print("Random Forest Cross-Validation Standard Deviation:", cv_rf.std())
```

GaussianNB

In [ ]:

```
from sklearn.naive_bayes import GaussianNB
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split, cross_val_score

# Loading the dataset
x, y = load_wine(return_X_y=True)

# Splitting the dataset into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=0, train_size=0.7)

# Naive Bayes
nb = GaussianNB()
nb.fit(x_train, y_train)

# Accuracy Scores
nb_acc_test = nb.score(x_test, y_test)
nb_acc_train = nb.score(x_train, y_train)

# Cross-Validation
cv_nb = cross_val_score(nb, x, y, cv=5)

# Printing Results
print("Naive Bayes Accuracy on Test Set:", nb_acc_test)
print("Naive Bayes Accuracy on Training Set:", nb_acc_train)
print("Naive Bayes Cross-Validation Scores:", cv_nb)
print("Naive Bayes Cross-Validation Mean Score:", cv_nb.mean())
print("Naive Bayes Cross-Validation Standard Deviation:", cv_nb.std())
```

KNeighbors Classifier

In [ ]:

```python
from sklearn.neighbors import KNeighborsClassifier
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split, cross_val_score

# Loading the dataset
x, y = load_wine(return_X_y=True)

# Splitting the dataset into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=0, train_size=0.7)

# K-Nearest Neighbors (KNN)
knn = KNeighborsClassifier()
knn.fit(x_train, y_train)

# Accuracy Scores
knn_acc_test = knn.score(x_test, y_test)
knn_acc_train = knn.score(x_train, y_train)

# Cross-Validation
cv_knn = cross_val_score(knn, x, y, cv=5)

# Printing Results
print("K-Nearest Neighbors Accuracy on Test Set:", knn_acc_test)
print("K-Nearest Neighbors Accuracy on Training Set:", knn_acc_train)
print("K-Nearest Neighbors Cross-Validation Scores:", cv_knn)
print("K-Nearest Neighbors Cross-Validation Mean Score:", cv_knn.mean())
print("K-Nearest Neighbors Cross-Validation Standard Deviation:", cv_knn.std())
```

AdaBoost Classifier

In [ ]:

```python
from sklearn.ensemble import AdaBoostClassifier
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split, cross_val_score

# Loading the dataset
x, y = load_wine(return_X_y=True)

# Splitting the dataset into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=0, train_size=0.7)

# AdaBoost
adaboost = AdaBoostClassifier()
adaboost.fit(x_train, y_train)

# Accuracy Scores
adaboost_acc_test = adaboost.score(x_test, y_test)
adaboost_acc_train = adaboost.score(x_train, y_train)

# Cross-Validation
cv_adaboost = cross_val_score(adaboost, x, y, cv=5)

# Printing Results
print("AdaBoost Accuracy on Test Set:", adaboost_acc_test)
print("AdaBoost Accuracy on Training Set:", adaboost_acc_train)
print("AdaBoost Cross-Validation Scores:", cv_adaboost)
print("AdaBoost Cross-Validation Mean Score:", cv_adaboost.mean())
print("AdaBoost Cross-Validation Standard Deviation:", cv_adaboost.std())
```

With Ratio 80 - 20

SVM

In [ ]:

```python
from sklearn.svm import SVC
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split, cross_val_score

# Loading the dataset
x, y = load_wine(return_X_y=True)

# Splitting the dataset into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=0, train_size=0.8)

# Support Vector Machine (SVM)
clf_svm = SVC()
clf_svm.fit(x_train, y_train)

# Accuracy Scores
svm_acc_test = clf_svm.score(x_test, y_test)
svm_acc_train = clf_svm.score(x_train, y_train)

# Cross-Validation
cv_svm = cross_val_score(clf_svm, x, y, cv=5)

# Printing Results
print("SVM Accuracy on Test Set:", svm_acc_test)
print("SVM Accuracy on Training Set:", svm_acc_train)
print("SVM Cross-Validation Scores:", cv_svm)
print("SVM Cross-Validation Scores percentage :", cv_svm * 100)
print("SVM Cross-Validation Mean Score:", cv_svm.mean())
print("SVM Cross-Validation Standard Deviation:", cv_svm.std())
```

Decision Tree Classifier

In [ ]:

```python
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.tree import DecisionTreeClassifier

# Loading the dataset
x, y = load_wine(return_X_y=True)

# Splitting the dataset into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=0, train_size=0.8)

# Decision Tree
dt = DecisionTreeClassifier()
dt.fit(x_train, y_train)

# Accuracy Scores
dt_acc_test = dt.score(x_test, y_test)
dt_acc_train = dt.score(x_train, y_train)

# Cross-Validation
cv_dt = cross_val_score(dt, x, y, cv=5)

# Printing Results
print("Decision Tree Accuracy on Test Set:", dt_acc_test)
print("Decision Tree Accuracy on Training Set:", dt_acc_train)
print("Decision Tree Cross-Validation Scores:", cv_dt)
print("Decision Tree Cross-Validation Mean Score:", cv_dt.mean())
print("Decision Tree Cross-Validation Standard Deviation:", cv_dt.std())
```

Logistic Regression

In [ ]:

```python
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split, cross_val_score

# Loading the dataset
x, y = load_wine(return_X_y=True)

# Splitting the dataset into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=0, train_size=0.8)

# Logistic Regression
lr = LogisticRegression()
lr.fit(x_train, y_train)

# Accuracy Scores
lr_acc_test = lr.score(x_test, y_test)
lr_acc_train = lr.score(x_train, y_train)

# Cross-Validation
cv_lr = cross_val_score(lr, x, y, cv=5)

# Printing Results
print("Logistic Regression Accuracy on Test Set:", lr_acc_test)
print("Logistic Regression Accuracy on Training Set:", lr_acc_train)
print("Logistic Regression Cross-Validation Scores:", cv_lr)
print("Logistic Regression Cross-Validation Mean Score:", cv_lr.mean())
print("Logistic Regression Cross-Validation Standard Deviation:", cv_lr.std())
```

Random Forest Classifier

In [ ]:

```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split, cross_val_score

# Loading the dataset
x, y = load_wine(return_X_y=True)

# Splitting the dataset into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=0, train_size=0.8)

# Random Forest
rf = RandomForestClassifier()
rf.fit(x_train, y_train)

# Accuracy Scores
rf_acc_test = rf.score(x_test, y_test)
rf_acc_train = rf.score(x_train, y_train)

# Cross-Validation
cv_rf = cross_val_score(rf, x, y, cv=5)

# Printing Results
print("Random Forest Accuracy on Test Set:", rf_acc_test)
print("Random Forest Accuracy on Training Set:", rf_acc_train)
print("Random Forest Cross-Validation Scores:", cv_rf)
print("Random Forest Cross-Validation Mean Score:", cv_rf.mean())
print("Random Forest Cross-Validation Standard Deviation:", cv_rf.std())
```

GaussianNB

In [ ]:

```python
from sklearn.naive_bayes import GaussianNB
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split, cross_val_score

# Loading the dataset
x, y = load_wine(return_X_y=True)

# Splitting the dataset into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=0, train_size=0.8)

# Naive Bayes
nb = GaussianNB()
nb.fit(x_train, y_train)

# Accuracy Scores
nb_acc_test = nb.score(x_test, y_test)
nb_acc_train = nb.score(x_train, y_train)

# Cross-Validation
cv_nb = cross_val_score(nb, x, y, cv=5)

# Printing Results
print("Naive Bayes Accuracy on Test Set:", nb_acc_test)
print("Naive Bayes Accuracy on Training Set:", nb_acc_train)
print("Naive Bayes Cross-Validation Scores:", cv_nb)
print("Naive Bayes Cross-Validation Mean Score:", cv_nb.mean())
print("Naive Bayes Cross-Validation Standard Deviation:", cv_nb.std())
```

KNeighbors Classifier

In [ ]:

```python
from sklearn.neighbors import KNeighborsClassifier
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split, cross_val_score

# Loading the dataset
x, y = load_wine(return_X_y=True)

# Splitting the dataset into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=0, train_size=0.8)

# K-Nearest Neighbors (KNN)
knn = KNeighborsClassifier()
knn.fit(x_train, y_train)

# Accuracy Scores
knn_acc_test = knn.score(x_test, y_test)
knn_acc_train = knn.score(x_train, y_train)

# Cross-Validation
cv_knn = cross_val_score(knn, x, y, cv=5)

# Printing Results
print("K-Nearest Neighbors Accuracy on Test Set:", knn_acc_test)
print("K-Nearest Neighbors Accuracy on Training Set:", knn_acc_train)
print("K-Nearest Neighbors Cross-Validation Scores:", cv_knn)
print("K-Nearest Neighbors Cross-Validation Mean Score:", cv_knn.mean())
print("K-Nearest Neighbors Cross-Validation Standard Deviation:", cv_knn.std())
```

AdaBoost Classifier

In [ ]:
```python
from sklearn.ensemble import AdaBoostClassifier
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split, cross_val_score

# Loading the dataset
x, y = load_wine(return_X_y=True)

# Splitting the dataset into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=0, train_size=0.8)

# AdaBoost
adaboost = AdaBoostClassifier()
adaboost.fit(x_train, y_train)

# Accuracy Scores
adaboost_acc_test = adaboost.score(x_test, y_test)
adaboost_acc_train = adaboost.score(x_train, y_train)

# Cross-Validation
cv_adaboost = cross_val_score(adaboost, x, y, cv=5)

# Printing Results
print("AdaBoost Accuracy on Test Set:", adaboost_acc_test)
print("AdaBoost Accuracy on Training Set:", adaboost_acc_train)
print("AdaBoost Cross-Validation Scores:", cv_adaboost)
print("AdaBoost Cross-Validation Mean Score:", cv_adaboost.mean())
print("AdaBoost Cross-Validation Standard Deviation:", cv_adaboost.std())
```

In [ ]:

In [ ]: