In [32]:
```
#Ensemble learning
```

In [31]:
```
import pandas as pd
apple=pd.read_csv("apple.csv")
apple=pd.DataFrame(apple)
X=apple.drop(["Apple"],axis=1)
Y=apple['Apple']
```

In [30]:
```
#soft voting and hard voting
from sklearn.ensemble import VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import load_iris
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test=train_test_split(X,Y,test_size=0.20,random_s
estimator=[] #empty list
estimator.append(('LR',LogisticRegression(solver='lbfgs',multi_class='mult
estimator.append(('SVC',SVC(gamma='auto',probability=True)))
estimator.append(('DTC',DecisionTreeClassifier()))
vot_hard=VotingClassifier(estimators=estimator,voting='hard')
vot_hard.fit(X_train, y_train)
y_pred=vot_hard.predict(X_test)
score=accuracy_score(y_test,y_pred)
print("Hard Voting Score %d"%score)
vot_soft=VotingClassifier(estimators=estimator,voting='soft')
vot_soft.fit(X_train,y_train)
y_pred=vot_soft.predict(X_test)
score=accuracy_score(y_test,y_pred)
print("Soft Voting Score %d"%score)
```

```
C:\Users\Nikhil\anaconda3\anaconda\envs\bb\Lib\site-packages\sklearn\linear_m
odel\_logistic.py:460: ConvergenceWarning: lbfgs 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 (https://sciki
t-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-regres
sion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession)
  n_iter_i = _check_optimize_result(

Hard Voting Score 1
Soft Voting Score 1

C:\Users\Nikhil\anaconda3\anaconda\envs\bb\Lib\site-packages\sklearn\linear_m
odel\_logistic.py:460: ConvergenceWarning: lbfgs 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 (https://sciki
t-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-regres
sion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession)
  n_iter_i = _check_optimize_result(
```

In [29]:
```
#weighted average
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
X=apple.drop(["Apple"],axis=1)
Y=apple['Apple']
X_train,X_test,y_train,y_test=train_test_split(X,Y,test_size=0.20,random_s
m1=DecisionTreeClassifier()
m2=KNeighborsClassifier()
m3=LogisticRegression()
m1.fit(x_train,y_train)
m2.fit(x_train,y_train)
m3.fit(x_train,y_train)
pred1=m1.predict_proba(x_test)
pred2=m2.predict_proba(x_test)
pred3=m3.predict_proba(x_test)
finalpred=(pred1*0.5+pred2*0.3+pred3*0.2)
print(finalpred)
```

```
[[6.75006810e-30 2.95547744e-22 1.32451939e-19 ... 5.30015882e-11
  1.06739232e-12 3.21181770e-13]
 [1.01173442e-48 2.61438135e-37 2.85940413e-33 ... 1.57504008e-06
  1.00462569e-07 4.18533417e-08]
 [1.49772692e-28 3.36261017e-21 1.17863213e-18 ... 1.89809369e-11
  3.49815608e-13 1.02658955e-13]
 ...
 [8.37187537e-77 9.22301086e-61 4.69483510e-55 ... 8.53980493e-03
  4.45367568e-03 3.36695188e-03]
 [8.37187537e-77 9.22301086e-61 4.69483510e-55 ... 8.53980493e-03
  4.45367568e-03 3.36695188e-03]
 [3.66525497e-05 1.42738252e-03 6.06246773e-03 ... 1.56329563e-28
  1.85747837e-31 2.56110260e-32]]

C:\Users\Nikhil\anaconda3\anaconda\envs\bb\Lib\site-packages\sklearn\linear_m
odel\_logistic.py:460: ConvergenceWarning: lbfgs 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 (https://sciki
t-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-regres
sion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession)
  n_iter_i = _check_optimize_result(
```

In [22]:

```
#Stacking
import pandas as pd
import matplotlib.pyplot as plt
from mlxtend.plotting import plot_confusion_matrix
from mlxtend.classifier import StackingClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder
label_encoder= LabelEncoder()

df = pd.read_csv("apple.csv")
df.head()

X = df.drop('Apple',axis=1)
y = df['Apple']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, r

sc = StandardScaler()  # StandardScaler initialization
var_transform = ['Humidity','Temperature','Rainfall']

X_train[var_transform] = sc.fit_transform(X_train[var_transform])  # Stand
X_test[var_transform] = sc.transform(X_test[var_transform])

KNC = KNeighborsClassifier()  # KNeighborsClassifier initialization
NB = GaussianNB()  # GaussianNB initialization

model_kNeighborsClassifier = KNC.fit(X_train, y_train)
pred_knc = model_kNeighborsClassifier.predict(X_test)
acc_knc = accuracy_score(y_test, pred_knc)  # Evaluating accuracy score
print('Accuracy score of KNeighbors Classifier is:', acc_knc * 100)

model_NaiveBayes = NB.fit(X_train, y_train)
pred_nb = model_NaiveBayes.predict(X_test)
acc_nb = accuracy_score(y_test, pred_nb)
print('Accuracy of Naive Bayes Classifier:', acc_nb * 100)
```

```
Accuracy score of KNeighbors Classifier is: 100.0
Accuracy of Naive Bayes Classifier: 100.0
```

In [23]:

```
#Logistic regression
lr = LogisticRegression()
clf_stack = StackingClassifier(classifiers =[KNC, NB],
                               meta_classifier = lr,
                               use_probas = True,
                               use_features_in_secondary = True)
model_stack = clf_stack.fit(X_train, y_train)
pred_stack = model_stack.predict(X_test)
acc_stack = accuracy_score(y_test, pred_stack)
print('accuracy score of Stacked model:', acc_stack * 100)
```

```
accuracy score of Stacked model: 100.0
```

In [28]:

```
#Bagging
import sys
import pandas as pd
import matplotlib
matplotlib.use('TkAgg')
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.ensemble import BaggingClassifier
from sklearn.preprocessing import LabelEncoder
data=pd.read_csv("apple.csv")
data.head()
data.columns
label_encoder=LabelEncoder()
X = df.drop('Apple',axis=1)
y = df['Apple']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25
estimator_range = [2,4,6,8,10,12,14,16]
models = []
scores = []
for n_estimators in estimator_range:

    # Create bagging classifier
    clf = BaggingClassifier(n_estimators = n_estimators, random_state = 22
    print(clf)
    # Fit the model
    clf.fit(X_train, y_train)

    # Append the model and score to their respective list
    models.append(clf)
    scores.append(accuracy_score(y_true = y_test, y_pred = clf.predict(X_t
plt.figure(figsize=(9,6))
plt.plot(estimator_range, scores)

# Adjust labels and font (to make visable)
plt.xlabel("n_estimators", fontsize = 18)
plt.ylabel("score", fontsize = 18)
plt.tick_params(labelsize = 16)

# Visualize plot
plt.show()

#Two  lines to make our compiler able to draw:
#plt.savefig(sys.stdout.buffer)
#sys.stdout.flush()
#print(scores)
```

```
BaggingClassifier(n_estimators=2, random_state=22)
BaggingClassifier(n_estimators=4, random_state=22)
BaggingClassifier(n_estimators=6, random_state=22)
BaggingClassifier(n_estimators=8, random_state=22)
BaggingClassifier(random_state=22)
BaggingClassifier(n_estimators=12, random_state=22)
BaggingClassifier(n_estimators=14, random_state=22)
BaggingClassifier(n_estimators=16, random_state=22)
```

In [27]:

```
#Blending
#Boosting
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClass
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Load your dataset
data = pd.read_csv("apple.csv")

# Handle missing values by filling with column means
numerical_cols = data.select_dtypes(include=['number']).columns
data[numerical_cols] = data[numerical_cols].fillna(data[numerical_cols].me

# Assuming 'Potability' is the target variable, and other columns are feat
X = df.drop('Apple',axis=1)
y = df['Apple']
# Split the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, r

# Base models
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
gb_model = GradientBoostingClassifier(n_estimators=100, random_state=42)
lr_model = LogisticRegression()

# Fit the base models
rf_model.fit(X_train, y_train)
gb_model.fit(X_train, y_train)
lr_model.fit(X_train, y_train)

# Make predictions on the test set
rf_preds = rf_model.predict(X_test)
gb_preds = gb_model.predict(X_test)
lr_preds = lr_model.predict(X_test)

# Weighted Averaging
weighted_avg_preds = (0.4 * rf_preds + 0.4 * gb_preds + 0.2 * lr_preds)

# Stacking
stacked_model = StackingClassifier(estimators=[('rf', rf_model), ('gb', gb
stacked_model.fit(X_train, y_train)
stacked_preds = stacked_model.predict(X_test)

# Blending
blend_preds = (rf_preds + gb_preds) / 2

# Boosting (using GradientBoostingClassifier as an example)
boosted_model = GradientBoostingClassifier(n_estimators=100, random_state=
boosted_model.fit(X_train, y_train)
boosted_preds = boosted_model.predict(X_test)

import numpy as np  # Import NumPy for array operations

# Convert weighted_avg_preds to binary predictions
threshold = 0.5  # You can adjust the threshold as needed
binary_weighted_avg_preds = np.where(weighted_avg_preds >= threshold, 1, 0
```

```
58
59 # Convert blend_preds to binary predictions
60 binary_blend_preds = np.where(blend_preds >= threshold, 1, 0)
61
62 # Evaluate the models with binary predictions
63 print("Weighted Averaging Accuracy:", accuracy_score(y_test, binary_weight
64 print("Stacking Accuracy:", accuracy_score(y_test, stacked_preds))
65 print("Blending Accuracy:", accuracy_score(y_test, binary_blend_preds))
66 print("Boosting Accuracy:", accuracy_score(y_test, boosted_preds))
```

```
C:\Users\Nikhil\anaconda3\anaconda\envs\bb\Lib\site-packages\sklearn\linear_m
odel\_logistic.py:460: ConvergenceWarning: lbfgs 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 (https://sciki
t-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-regres
sion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession)
  n_iter_i = _check_optimize_result(
C:\Users\Nikhil\anaconda3\anaconda\envs\bb\Lib\site-packages\sklearn\linear_m
odel\_logistic.py:460: ConvergenceWarning: lbfgs 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 (https://sciki
t-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-regres
sion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession)
  n_iter_i = _check_optimize_result(
C:\Users\Nikhil\anaconda3\anaconda\envs\bb\Lib\site-packages\sklearn\model_se
lection\_split.py:725: UserWarning: The least populated class in y has only 3
members, which is less than n_splits=5.
  warnings.warn(
C:\Users\Nikhil\anaconda3\anaconda\envs\bb\Lib\site-packages\sklearn\model_se
lection\_split.py:725: UserWarning: The least populated class in y has only 3
members, which is less than n_splits=5.
  warnings.warn(
C:\Users\Nikhil\anaconda3\anaconda\envs\bb\Lib\site-packages\sklearn\model_se
lection\_split.py:725: UserWarning: The least populated class in y has only 3
members, which is less than n_splits=5.
  warnings.warn(
C:\Users\Nikhil\anaconda3\anaconda\envs\bb\Lib\site-packages\sklearn\linear_m
odel\_logistic.py:460: ConvergenceWarning: lbfgs 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 (https://sciki
t-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-regres
sion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession)
  n_iter_i = _check_optimize_result(
C:\Users\Nikhil\anaconda3\anaconda\envs\bb\Lib\site-packages\sklearn\linear_m
odel\_logistic.py:460: ConvergenceWarning: lbfgs 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 (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 (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
C:\Users\Nikhil\anaconda3\anaconda\envs\bb\Lib\site-packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfgs 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 (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 (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
C:\Users\Nikhil\anaconda3\anaconda\envs\bb\Lib\site-packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfgs 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 (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 (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
C:\Users\Nikhil\anaconda3\anaconda\envs\bb\Lib\site-packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfgs 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 (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 (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(

Weighted Averaging Accuracy: 0.0
Stacking Accuracy: 1.0
Blending Accuracy: 0.0
Boosting Accuracy: 1.0
```

In [ ]:

```
1
```