Bahria University,

Karachi Campus

## LAB EXPERIMENT NO.

12

## LIST OF TASKS

|  |  |
| --- | --- |
| **TASK NO** | **OBJECTIVE** |
| **1** | Optimization of Ensemble Models using Decision Trees Enhance the performance of existing Adaboost and Bagging algorithms which utilize Decision Trees as base learners. The objective is to optimize these models to achieve 100% efficiency, signifying perfect accuracy on the dataset. This will involve analyzing the current model configurations, identifying inefficiencies, and adjusting the Decision Tree parameters and training process to maximize predictive accuracy. |
| **2** | Implementation of Adaboost and Bagging Algorithms with Support Vector Machine Implement Adaboost and Bagging ensemble techniques using Support Vector Machines (SVM) as the base learner. This task will explore how the robust features of SVM can be leveraged within ensemble frameworks to potentially improve model performance. The focus will be on configuring the SVM settings appropriately within each ensemble method, evaluating their effectiveness, and comparing the results to those obtained with Decision Trees. |
|  |  |

Submitted On:

22 May 2024

(Date: DD/MM/YY)

**TASK #1**: Optimization of Ensemble Models using Decision Trees Enhance the performance of existing Adaboost and Bagging algorithms which utilize Decision Trees as base learners. The objective is to optimize these models to achieve 100% efficiency, signifying perfect accuracy on the dataset. This will involve analyzing the current model configurations, identifying inefficiencies, and adjusting the Decision Tree parameters and training process to maximize predictive accuracy.

import numpy as np

import pandas as pd

from sklearn.ensemble import AdaBoostClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score, confusion\_matrix

from sklearn.datasets import load\_iris

iris = load\_iris()

X = iris.data

y = iris.target

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.5, random\_state=42)

base\_classifier = DecisionTreeClassifier(max\_depth=1)

boosting\_clf = AdaBoostClassifier(base\_classifier, n\_estimators=300, random\_state=42)

boosting\_clf.fit(X\_train, y\_train)

y\_pred = boosting\_clf.predict(X\_test)

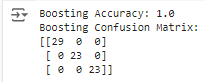
accuracy = accuracy\_score(y\_test, y\_pred)

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

print(f'Boosting Accuracy: {accuracy}')

print('Boosting Confusion Matrix:')

print(conf\_matrix)



from sklearn.ensemble import BaggingClassifier

from sklearn.svm import SVC

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score, confusion\_matrix

iris = load\_iris()

X = iris.data

y = iris.target

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.4, random\_state=42)

svm\_base\_classifier = SVC(kernel='linear', C=1)

bagging\_svm = BaggingClassifier(base\_estimator=svm\_base\_classifier, n\_estimators=50, random\_state=42)

bagging\_svm.fit(X\_train, y\_train)

bagging\_svm\_preds = bagging\_svm.predict(X\_test)

bagging\_svm\_accuracy = accuracy\_score(y\_test, bagging\_svm\_preds)

bagging\_svm\_conf\_matrix = confusion\_matrix(y\_test, bagging\_svm\_preds)

print("Bagging with SVM accuracy:", bagging\_svm\_accuracy)

print("Bagging with SVM Confusion Matrix:")

print(bagging\_svm\_conf\_matrix)

A screenshot of a computer code

Description automatically generated

**TASK # 2**: Implementation of Adaboost and Bagging Algorithms with Support Vector Machine Implement Adaboost and Bagging ensemble techniques using Support Vector Machines (SVM) as the base learner. This task will explore how the robust features of SVM can be leveraged within ensemble frameworks to potentially improve model performance. The focus will be on configuring the SVM settings appropriately within each ensemble method, evaluating their effectiveness, and comparing the results to those obtained with Decision Trees.

import numpy as np

import pandas as pd

from sklearn.ensemble import AdaBoostClassifier, BaggingClassifier

from sklearn.svm import SVC

from sklearn.tree import DecisionTreeClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score

from sklearn.datasets import load\_iris

iris = load\_iris()

X = iris.data

y = iris.target

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.4, random\_state=42)

dt\_base\_classifier = DecisionTreeClassifier(max\_depth=1)

adaboost\_dt = AdaBoostClassifier(base\_estimator=dt\_base\_classifier, n\_estimators=50, random\_state=42)

adaboost\_dt.fit(X\_train, y\_train)

adaboost\_dt\_preds = adaboost\_dt.predict(X\_test)

adaboost\_dt\_accuracy = accuracy\_score(y\_test, adaboost\_dt\_preds)

print("Adaboost with Decision Tree accuracy:", adaboost\_dt\_accuracy)

svm\_base\_classifier = SVC(kernel='linear', C=1)

adaboost\_svm = AdaBoostClassifier(base\_estimator=svm\_base\_classifier, n\_estimators=50, random\_state=42, algorithm='SAMME')

adaboost\_svm.fit(X\_train, y\_train)

adaboost\_svm\_preds = adaboost\_svm.predict(X\_test)

adaboost\_svm\_accuracy = accuracy\_score(y\_test, adaboost\_svm\_preds)

print("Adaboost with SVM accuracy:", adaboost\_svm\_accuracy)



from sklearn.ensemble import BaggingClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.svm import SVC

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score, confusion\_matrix

iris = load\_iris()

X = iris.data

y = iris.target

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.4, random\_state=42)

dt\_base\_classifier = DecisionTreeClassifier()

bagging\_dt = BaggingClassifier(base\_estimator=dt\_base\_classifier, n\_estimators=50, random\_state=42)

bagging\_dt.fit(X\_train, y\_train)

bagging\_dt\_preds = bagging\_dt.predict(X\_test)

bagging\_dt\_accuracy = accuracy\_score(y\_test, bagging\_dt\_preds)

bagging\_dt\_conf\_matrix = confusion\_matrix(y\_test, bagging\_dt\_preds)

svm\_base\_classifier = SVC(kernel='linear', C=1)

bagging\_svm = BaggingClassifier(base\_estimator=svm\_base\_classifier, n\_estimators=50, random\_state=42)

bagging\_svm.fit(X\_train, y\_train)

bagging\_svm\_preds = bagging\_svm.predict(X\_test)

bagging\_svm\_accuracy = accuracy\_score(y\_test, bagging\_svm\_preds)

bagging\_svm\_conf\_matrix = confusion\_matrix(y\_test, bagging\_svm\_preds)

print("Bagging with Decision Tree accuracy:", bagging\_dt\_accuracy)

print("Bagging with Decision Tree Confusion Matrix:")

print(bagging\_dt\_conf\_matrix)

print("\n")

print("Bagging with SVM accuracy:", bagging\_svm\_accuracy)

print("Bagging with SVM Confusion Matrix:")

print(bagging\_svm\_conf\_matrix)

