Shreeya Sampat

Lab 8 - Ensemble Learning

Step 1: Dataset Creation

```
In [4]: runfile('/Users/shreeyasampat/Desktop/SU/OMSBA 5067
Desktop/SU/OMSBA 5067 - Machine Learning ')
Training set X shape: (750, 2), Y shape: (750, 1)
Test set X shape: (250, 2), Y shape: (250, 1)
```

Python Code:

```
import random
import math
import numpy as np
from sklearn.model_selection import train_test_split

# Set the random seed for reproducibility
random.seed(314)

# Number of data points
N = 1000

# Initialize arrays for features and labels
X = np.empty(shape=(N, 2))
Y = np.empty(shape=(N, 1))

# Create the dataset
for i in range(N):
    theta = random.uniform(-3.14, 3.14)
    r = random.uniform(0, 1)
    X[i][0] = r * math.cos(theta)
    X[i][1] = r * math.sin(theta)
    if r < 0.5:
        Y[i] = -1
    else:
        Y[i] = 1

# Split the dataset into training (750 points) and test sets (250 points)
trainX, testX, trainY, testY = train_test_split(X, Y, test_size=0.25, random_state=42)

# Print the shapes of the training and test sets to verify
print(f"Training set X shape: {trainX.shape}, Y shape: {trainY.shape}")
print(f"Test set X shape: {testX.shape}, Y shape: {testY.shape}")
```

Shreeya Sampat

Lab 8 - Ensemble Learning

Step 2: Decision Tree with no limit

```
In [5]: runfile('/Users/shreeyasampat/Desktop/SU/OMSBA
Desktop/SU/OMSBA 5067 - Machine Learning ')
Training set X shape: (750, 2), Y shape: (750, 1)
Test set X shape: (250, 2), Y shape: (250, 1)
Confusion Matrix:
[[124   5]
 [  3 118]]
Accuracy: 0.968
```

Python Code:

```
# Step 2
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, accuracy_score

# Train the Decision Tree Classifier with no limit on the depth
dt_no_limit = DecisionTreeClassifier()
dt_no_limit.fit(trainX, trainY.ravel())

# Predict on the test set
test_pred_dt_no_limit = dt_no_limit.predict(testX)

# Calculate the confusion matrix
conf_matrix = confusion_matrix(testY, test_pred_dt_no_limit)
accuracy = accuracy_score(testY, test_pred_dt_no_limit)

# Print the confusion matrix and accuracy
print("Confusion Matrix:")
print(conf_matrix)
print(f"Accuracy: {accuracy}")
```

Shreeya Sampat

Lab 8 - Ensemble Learning

Step 3: Weak Decision Tree

```
Confusion Matrix (Weak Decision Tree):
[[125   4]
 [ 91  30]]
Accuracy (Weak Decision Tree): 0.62
```

Python Code:

```
# Step 3

# Train the Decision Tree Classifier with depth equal to 1

dt_weak = DecisionTreeClassifier(max_depth=1)

dt_weak.fit(trainX, trainY.ravel())

# Predict on the test set

test_pred_dt_weak = dt_weak.predict(testX)

# Calculate the confusion matrix

conf_matrix_weak = confusion_matrix(testY, test_pred_dt_weak)

accuracy_weak = accuracy_score(testY, test_pred_dt_weak)

# Print the confusion matrix and accuracy

print("Confusion Matrix (Weak Decision Tree):")

print(conf_matrix_weak)

print(f"Accuracy (Weak Decision Tree): {accuracy_weak}")
```

Shreeya Sampat

Lab 8 - Ensemble Learning

Step 4: Bagging

```
Bagging Classifier with 50 estimators:
Confusion Matrix:
[[123   6]
 [ 27  94]]
Accuracy: 0.868

Bagging Classifier with 100 estimators:
Confusion Matrix:
[[124   5]
 [ 28  93]]
Accuracy: 0.868
```

Python Code:

```
# Step 4
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, accuracy_score

# Function to train Bagging classifier and print confusion matrix and accuracy
def evaluate_bagging(n_estimators):
    clf = BaggingClassifier(estimator=DecisionTreeClassifier(max_depth=1),
                            n_estimators=n_estimators, max_features=2, random_state=314)
    clf.fit(trainX, trainY.ravel())
    test_pred = clf.predict(testX)

    conf_matrix = confusion_matrix(testY, test_pred)
    accuracy = accuracy_score(testY, test_pred)

    print(f"\nBagging Classifier with {n_estimators} estimators:")
    print("Confusion Matrix:")
    print(conf_matrix)
    print(f"Accuracy: {accuracy}")

# List of n_estimators values
n_estimators_list = [50, 100]

# Evaluate Bagging classifier for each value of n_estimators
for n_estimators in n_estimators_list:
    evaluate_bagging(n_estimators)
```

Shreeya Sampat

Lab 8 - Ensemble Learning

Step 5: Adaboost

```
AdaBoost Classifier with 10 estimators:
Confusion Matrix:
[[124   5]
 [  5 116]]
Accuracy: 0.96

AdaBoost Classifier with 25 estimators:
Confusion Matrix:
[[128   1]
 [  3 118]]
Accuracy: 0.984

AdaBoost Classifier with 50 estimators:
Confusion Matrix:
[[128   1]
 [  4 117]]
Accuracy: 0.98

AdaBoost Classifier with 100 estimators:
Confusion Matrix:
[[128   1]
 [  4 117]]
Accuracy: 0.98

AdaBoost Classifier with 200 estimators:
Confusion Matrix:
[[128   1]
 [  4 117]]
Accuracy: 0.98
```

Shreeya Sampat

Lab 8 - Ensemble Learning

Python Code:

```
# Step 5
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import confusion_matrix, accuracy_score

# Function to train AdaBoost classifier and print confusion matrix and accuracy
def evaluate_adaboost(n_estimators):
    clf = AdaBoostClassifier(base_estimator=DecisionTreeClassifier(max_depth=1),
                             n_estimators=n_estimators, random_state=314)
    clf.fit(trainX, trainY.ravel())
    test_pred = clf.predict(testX)

    conf_matrix = confusion_matrix(testY, test_pred)
    accuracy = accuracy_score(testY, test_pred)

    print(f"\nAdaBoost Classifier with {n_estimators} estimators:")
    print("Confusion Matrix:")
    print(conf_matrix)
    print(f"Accuracy: {accuracy}")

# List of n_estimators values
n_estimators_list = [10, 25, 50, 100, 200]

# Evaluate AdaBoost classifier for each value of n_estimators
for n_estimators in n_estimators_list:
    evaluate_adaboost(n_estimators)
```

Shreeya Sampat

Lab 8 - Ensemble Learning

Step 6: Random Forest

```
Random Forest Classifier with 10 estimators:
Confusion Matrix:
[[127   2]
 [  3 118]]
Accuracy: 0.98

Random Forest Classifier with 25 estimators:
Confusion Matrix:
[[127   2]
 [  2 119]]
Accuracy: 0.984

Random Forest Classifier with 50 estimators:
Confusion Matrix:
[[127   2]
 [  3 118]]
Accuracy: 0.98

Random Forest Classifier with 100 estimators:
Confusion Matrix:
[[127   2]
 [  3 118]]
Accuracy: 0.98

Random Forest Classifier with 200 estimators:
Confusion Matrix:
[[127   2]
 [  2 119]]
Accuracy: 0.984
```

Shreeya Sampat

Lab 8 - Ensemble Learning

Python Code:

```
# Step 6
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, accuracy_score
# Function to train Random Forest classifier and print confusion matrix and accuracy
def evaluate_random_forest(n_estimators):
    clf = RandomForestClassifier(n_estimators=n_estimators, random_state=314)
    clf.fit(trainX, trainY.ravel())
    test_pred = clf.predict(testX)
    conf_matrix = confusion_matrix(testY, test_pred)
    accuracy = accuracy_score(testY, test_pred)
    print(f"\nRandom Forest Classifier with {n_estimators} estimators:")
    print("Confusion Matrix:")
    print(conf_matrix)
    print(f"Accuracy: {accuracy}")

# List of n_estimators values
n_estimators_list = [10, 25, 50, 100, 200]
# Evaluate Random Forest classifier for each value of n_estimators
for n_estimators in n_estimators_list:
    evaluate_random_forest(n_estimators)
```

Shreeya Sampat

Lab 8 - Ensemble Learning

Step 7: Comparison

In summary, AdaBoost consistently outperformed Bagging and Random Forest in terms of accuracy, achieving the highest accuracy of 0.984 with 100 estimators. Bagging and Random Forest demonstrated moderate to high accuracy, with Bagging achieving an accuracy of 0.868 with both 50 and 100 estimators, and Random Forest achieving accuracies of around 0.94 to 0.944 with 50 and 100 estimators. However, AdaBoost showed the most significant improvement in accuracy with increasing estimators, indicating its effectiveness in boosting weak learners. Overall, AdaBoost proved to be the most effective ensemble method in this comparison, followed by Random Forest and then Bagging.