cmps320_assignment5_SammanBhetwal

November 26, 2024

0.1 Homework 5 Part B

[1]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.ensemble import VotingClassifier
import numpy as np
```

[2]:
```
# Here we will import the dataset
```

[3]:
```
from sklearn.datasets import fetch_openml
mnist = fetch_openml('mnist_784', version=1)
```

[5]:
```
mnist.data.shape
```

[5]:
```
(70000, 784)
```

[6]:
```
mnist.data.head()
```

[6]:

	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	\
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

	pixel10	…	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	\
0	0.0	…	0.0	0.0	0.0	0.0	0.0	0.0	
1	0.0	…	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.0	…	0.0	0.0	0.0	0.0	0.0	0.0	
3	0.0	…	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	…	0.0	0.0	0.0	0.0	0.0	0.0	

```
   pixel781  pixel782  pixel783  pixel784
0       0.0       0.0       0.0       0.0
1       0.0       0.0       0.0       0.0
2       0.0       0.0       0.0       0.0
3       0.0       0.0       0.0       0.0
4       0.0       0.0       0.0       0.0

[5 rows x 784 columns]
```

0.1.1 1. Split the dataset into a training, validation, and test set using the ratio 5 : 1 : 1.

[10]:
```
from sklearn.model_selection import train_test_split

X = mnist['data']
y = mnist['target']

X_train, X_val_test, y_train, y_val_test = train_test_split(X, y, train_size=5/
↪7, random_state=42)

X_val, X_test, y_val, y_test = train_test_split(X_val_test,␣
↪y_val_test,train_size=1/2, random_state=42)
```

[11]:
```
print(X_train.shape, X_val.shape, X_test.shape)
```

```
(50000, 784) (10000, 784) (10000, 784)
```

[12]:
```
print(y_train.shape, y_val.shape, y_test.shape)
```

```
(50000,) (10000,) (10000,)
```

0.1.2 2. Next, train the following classifiers:

a. Random Forest Classifier

[26]:
```
random_forest_classifier = RandomForestClassifier(random_state=42)
random_forest_classifier.fit(X_train, y_train)
```

[26]:
```
RandomForestClassifier(random_state=42)
```

[27]:
```
y_val_prediction = random_forest_classifier.predict(X_val)
validation_accuracy = accuracy_score(y_val, y_val_prediction)
validation_accuracy
print("Value Accuracy for Random Forest Classifier is: ", validation_accuracy)
```

```
Value Accuracy for Random Forest Classifier is:  0.9677
```

b. Bagging Classifier

[28]:
```
bagging_classifier = BaggingClassifier(
    DecisionTreeClassifier(random_state=42),
```

```
    n_estimators=500,
    max_samples=100,
    bootstrap=True,
    random_state=42)
bagging_classifier.fit(X_train, y_train)
```

[28]:
```
BaggingClassifier(base_estimator=DecisionTreeClassifier(random_state=42),
                  max_samples=100, n_estimators=500, random_state=42)
```

[29]:
```
y_val_prediction = bagging_classifier.predict(X_val)
validation_accuracy = accuracy_score(y_val, y_val_prediction)
validation_accuracy
print("Value Accuracy for Bagging Classifier: ", validation_accuracy)
```

```
Value Accuracy for Bagging Classifier:  0.8394
```

c. Decision Tree Classifier

[30]:
```
decision_tree_classifier = DecisionTreeClassifier(random_state=42)
decision_tree_classifier.fit(X_train, y_train)
```

[30]:
```
DecisionTreeClassifier(random_state=42)
```

[31]:
```
y_val_prediction = decision_tree_classifier.predict(X_val)
validation_accuracy = accuracy_score(y_val, y_val_prediction)
validation_accuracy
print("Value Accuracy for Bagging Classifier: ", validation_accuracy)
```

```
Value Accuracy for Bagging Classifier:  0.8714
```

0.1.3 3. Obtain the top ten variable importance ranking from Decision tree and Random Forest classifier. Please interpret the results.

[32]:
```
decision_tree_importances = decision_tree_classifier.feature_importances_
random_forest_importances = random_forest_classifier.feature_importances_

decision_tree_importances = np.argsort(decision_tree_importances)[::-1][:10]
random_forest_importances = np.argsort(random_forest_importances)[::-1][:10]
```

[33]:
```
print("Decision Tree Top 10 Features:", decision_tree_importances)
```

```
Decision Tree Top 10 Features: [489 435 350 347 542 211 486 432 597 271]
```

[34]:
```
print("Random Forest Top 10 Features:", random_forest_importances)
```

```
Random Forest Top 10 Features: [378 405 461 434 401 433 406 409 542 375]
```

The Decision Tree's decision-making is significantly impacted by the pixels at indices 489, 435, 350, 347, 542, 211, 486, 432, 597, and 271. These particular pixels play a crucial role in distinguishing

between the digits in the MNIST dataset. Their importance in the model suggests that changes at these pixel positions are highly influential in identifying the digit.

The Random Forest model highlights the pixels at indices 378, 405, 461, 434, 401, 433, 406, 409, 542, and 375 as the most influential. These pixel locations on the 28x28 grid are consistently significant across various decision trees within the forest. The repeated importance of pixel 542 in both models underscores its unique significance.

0.1.4 4. Combine three classifiers in Part 2 into an ensemble on the validation set using hard voting.

[35]:
```
voting_clf = VotingClassifier(
    estimators=[
        ('rf', random_forest_classifier),
        ('bagging', bagging_classifier),
        ('dt', decision_tree_classifier)
    ],
    voting='hard'
)

voting_clf.fit(X_train, y_train)

y_val_prediction = voting_clf.predict(X_val)
validation_accuracy = accuracy_score(y_val, y_val_prediction)
```

[36]:
```
print(f"The Validation Accuracy: {validation_accuracy}")
```

```
The Validation Accuracy: 0.9334
```

0.1.5 5. Does the ensemble outperform the individual classifiers?

Random Forest Classifier: 0.9677 Bagging Classifier: 0.8394 Decision Tree Classifier: 0.8714 Ensemble Classifier: 0.9334

The ensemble classifier falls short of surpassing the Random Forest classifier, which achieves the highest accuracy among the individual models. Nevertheless, the ensemble outperforms both the Bagging and Decision Tree classifiers when assessed separately.

0.1.6 6. Next, remove the individual classifier with the smallest accuracy score.

The classifier with the lowest accuracy is the Bagging classifier (BG) has the lowest accuracy with an accuracy of 0.8394. Therefore, we can remove this.

0.1.7 7. Now combine the classifiers into an ensemble on the test data using hard voting.

[37]:
```
revised_voting_classifier = VotingClassifier(
    estimators= [
        ('rf', random_forest_classifier),
        ('dt', decision_tree_classifier)
```

```
    ],
    voting='hard'

)

revised_voting_classifier.fit(X_train, y_train)

y_val_prediction_revised = revised_voting_classifier.predict(X_val)
validation_accuracy_revised = accuracy_score(y_val, y_val_prediction_revised)
print(f"Revised Validation Accuracy: {validation_accuracy_revised}")
```

```
Revised Validation Accuracy: 0.9174
```

0.1.8 8. How much better does it perform compared to the individual classifiers? Comment on your results.

The revised ensemble does not outperform the Random Forest classifier. Random Forest Classifier remains the best performing model with an accuracy of 0.9677. Thus, we can conclude that the Random Forest classifier's strong individual performance is not significantly enhanced by combining it with the Decision Tree classifier.

[ ]:

[ ]: