20-Voting_Classifier

October 20, 2024

1 Voting Classifier

A **Voting Classifier** is an ensemble learning technique that combines multiple machine learning models (often referred to as "base learners" or "weak learners") to make a single, more robust prediction. The idea is to leverage the strengths of different algorithms and reduce the risk of overfitting while improving the overall accuracy of predictions.

The Voting Classifier is a powerful ensemble method that improves prediction accuracy and robustness by aggregating multiple machine learning models. Its flexibility and effectiveness make it a valuable tool in many predictive modeling tasks.

There are two main types of voting:

- **Hard Voting**: The predicted class is the one that receives the majority of votes from the base models.
- **Soft Voting**: The predicted class is based on the average predicted probabilities of each class, and the class with the highest average probability is chosen.

1.0.1 When to Use Voting Classifier?

Voting Classifier is particularly useful when:

- You want to improve prediction accuracy by combining the strengths of different models.
- You have multiple algorithms that perform well individually, and you want to harness their collective performance.
- You want to reduce variance and increase robustness, especially in scenarios where different models may have different error patterns.

1.0.2 How Does Voting Classifier Work?

The Voting Classifier operates as follows:

1. Model Selection:

- Choose a diverse set of models (e.g., Logistic Regression, Decision Trees, Support Vector Machines, etc.) that will serve as base learners.

2. Training:

- Each model is trained independently on the same training dataset.

3. Prediction:

- For a new data point:
  - Hard Voting: Each model predicts a class label, and the class with the most votes is selected as the final prediction.
  - Soft Voting: Each model outputs the predicted probabilities for each class. The probabilities are averaged across all models, and the class with the highest average probability is selected.

4. Final Output:

- The Voting Classifier provides the final class label for the data point based on the voting mechanism.

1.0.3 Who Should Use Voting Classifier?

- **Data scientists and machine learning practitioners**: Who aim to build robust models by combining the strengths of multiple algorithms.
- **Kaggle competitors**: The Voting Classifier is a common strategy in competitions to boost accuracy by leveraging multiple models.
- **Industries requiring high accuracy**: Useful in fields like finance, healthcare, and marketing, where precise predictions are critical.

Advantages of Voting Classifier:

- **Improved accuracy**: Combining models often leads to better performance than any individual model.
- **Robustness**: Reduces the risk of overfitting and is less sensitive to noise in the data.
- **Flexibility**: Can incorporate different types of models, allowing the use of various algorithms tailored to specific aspects of the data.
- **Easy implementation**: Simple to implement using libraries like scikit-learn.

Disadvantages of Voting Classifier:

- **Complexity**: More complex than using a single model, as it requires managing multiple algorithms.
- **Computationally expensive**: Training multiple models can be time-consuming and resource-intensive, especially with large datasets.
- **Interpretability**: The ensemble nature makes it harder to interpret the model compared to a single algorithm.

1.0.4 Real-World Applications of Voting Classifier:

1. **Sentiment analysis**: Combining different models to classify sentiments from text data.
2. **Fraud detection**: In finance, where multiple algorithms can identify fraudulent patterns.
3. **Medical diagnosis**: Using various models to improve the accuracy of predicting patient outcomes.

4. **Customer churn prediction**: In marketing, to predict which customers are likely to stop using a service.

[34]:
```
import pandas as pd

from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split, cross_val_score,␣
 ↪GridSearchCV
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, VotingClassifier

X, y = make_classification(n_samples=2500, n_features=15, n_informative=8,␣
 ↪n_redundant=2, random_state=11)
X_train, X_test, Y_train, Y_test = train_test_split(X,y, test_size=0.2,␣
 ↪random_state=11)
```

[35]:
```
gnb = GaussianNB()
gnb.fit(X_train, Y_train)
```

[35]:
```
GaussianNB()
```

[36]:
```
cross_val_score(gnb,X_train, Y_train, cv=3).mean()
```

[36]:
```
0.7884931408169789
```

[37]:
```
lr = LogisticRegression()
lr.fit(X_train, Y_train)
```

[37]:
```
LogisticRegression()
```

[38]:
```
cross_val_score(lr,X_train, Y_train, cv=3).mean()
```

[38]:
```
0.7455041248144697
```

[39]:
```
rfc = RandomForestClassifier()
rfc.fit(X_train, Y_train)
```

[39]:
```
RandomForestClassifier()
```

[40]:
```
cross_val_score(rfc,X_train, Y_train, cv=3).mean()
```

[40]:
```
0.9044929487208347
```

[41]:
```
vc = VotingClassifier([
    ('NaiveBayes', gnb),
    ('LogisticRegression', lr),
    ('RandomForestClassifier', rfc)
```

```
])
cross_val_score(vc,X_train, Y_train, cv=3).mean()
```

[41]:
```
0.8315016665841254
```

[42]:
```
param_grid = {
    'voting': ['hard', 'soft'],
    'weights': [(1,1,1), (2,1,1), (1,2,1), (1,1,2)]
}

vc2 = GridSearchCV(vc, param_grid, cv=5, n_jobs=-1)
vc2.fit(X_train, Y_train)
```

[42]:
```
GridSearchCV(cv=5,
             estimator=VotingClassifier(estimators=[('NaiveBayes',
                                                     GaussianNB()),
                                                    ('LogisticRegression',
                                                     LogisticRegression()),
                                                    ('RandomForestClassifier',
RandomForestClassifier())]),
             n_jobs=-1,
             param_grid={'voting': ['hard', 'soft'],
                         'weights': [(1, 1, 1), (2, 1, 1), (1, 2, 1),
                                     (1, 1, 2)]})
```

[43]:
```
vc2.best_params_
```

[43]:
```
{'voting': 'hard', 'weights': (1, 1, 2)}
```

[44]:
```
vc2.best_score_
```

[44]:
```
0.875
```