Random Forest

In many cases we find that the aggregated answer is better than an expert's answer.

Similarly, if you aggregate the predictions of a group of predictors (such as classifiers or regressors), you will often get better predictions than with the best individual predictor. A group of predictors is called an ensemble; thus, this technique is called ensemble learning, and an ensemble learning algorithm is called an ensemble method.

you can train a group of decision tree classifiers, each on a different random subset of the training set. You can then obtain the predictions of all the individual trees, and the class that gets the most votes is the ensemble's prediction. Such an ensemble of decision trees is called a random forest, and despite its simplicity, this is one of the most powerful machine learning algorithms available today.

Voting Classifiers

A very simple way to create an even better classifier is to aggregate the predictions of each classifier: the class that gets the most votes is the ensemble's prediction. This majority-vote classifier is called a hard voting.

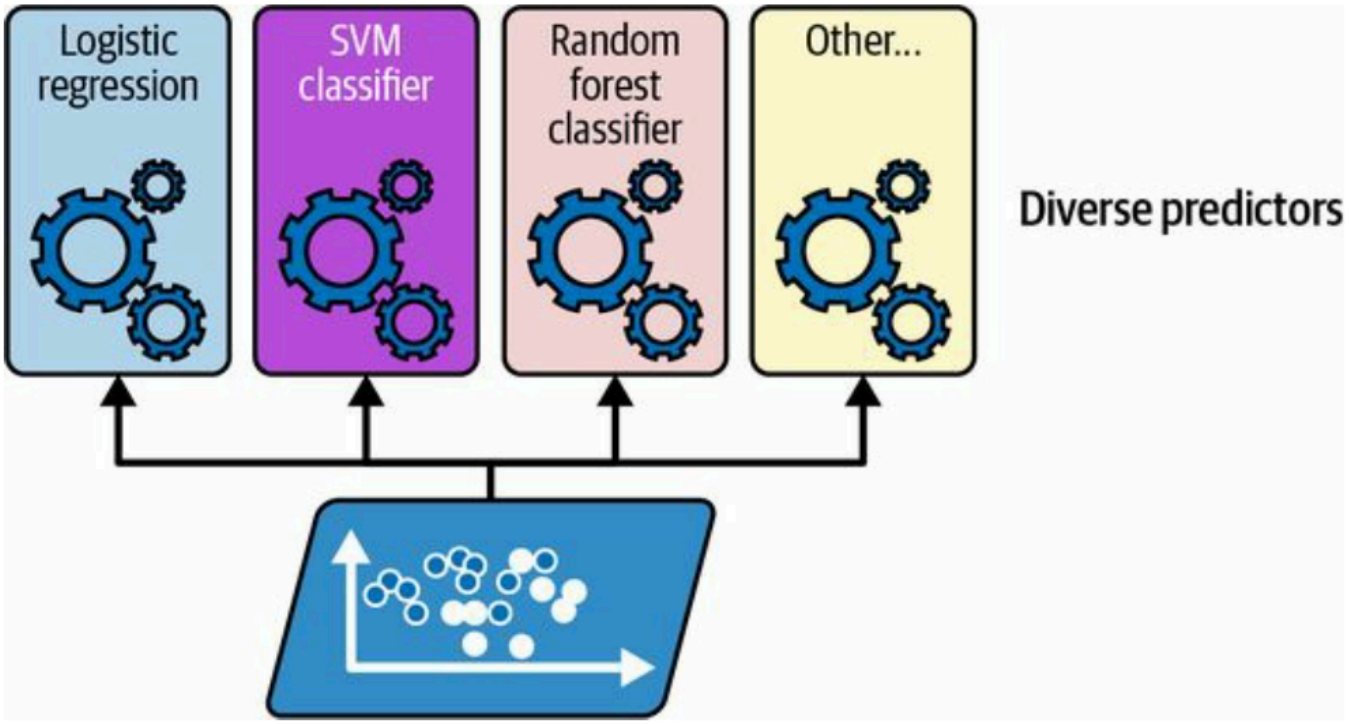

Figure 7-1. Training diverse classifiers

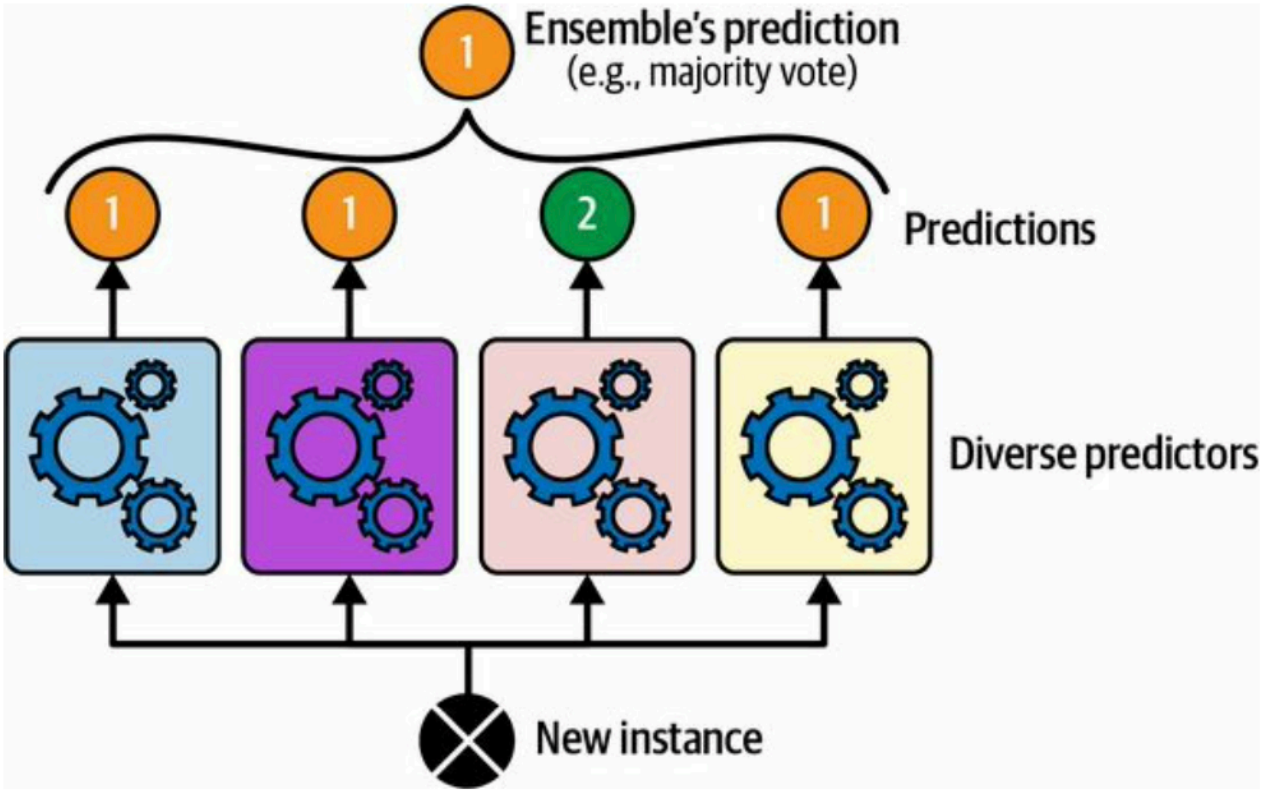

Figure 7-2. Hard voting classifier predictions

this voting classifier often achieves a higher accuracy than the best classifier in the ensemble. the ensemble can still be a strong learner (achieving high accuracy), provided there are a sufficient number of weak learners in the ensemble and they are sufficiently diverse.

Ensemble methods work best when the predictors are independent from one another.

Sklearn provides a VotingClassifier class thats easy to use , just give the names of the predictor pairs to use.

If all classifiers are able to estimate class probabilities (i.e., if they all have a predict _proba() method), then you can tell Scikit-Learn to predict the class with the highest class probability, averaged over all the individual classifiers. This is called soft voting. It often achieves higher performance than hard voting because it gives more weight to highly confident votes.

Bagging and pasting

One way to get a diverse set of classifiers is to use very different training algorithms, as just discussed. Another approach is to use the same training algorithm for every predictor but train them on different random subsets of the training set.
When sampling is performed with replacement, 1 this method is called bagging When sampling is performed without replacement, it is called pasting.

Both aim to reduce variance

both bagging and pasting allow training instances to be sampled several times across multiple predictors, but only bagging allows training instances to be sampled several times for the same predictor.

Bagging introduces a bit more diversity in the subsets that each predictor is trained on, so bagging ends up with a slightly higher bias than pasting; but the extra diversity also means that the predictors end up being less correlated, so the ensemble's variance is reduced. Overall, bagging often results in better models.

Out of bag evaluation

Out-of-bag (OOB) evaluation is a technique used primarily in ensemble methods like random forests to estimate the performance of a model without needing a separate validation set. Here's how it works:

1. **Bootstrap Sampling**: When creating each tree in a random forest, the algorithm samples the training data with replacement. This means that for each tree, some data points are left out of the sample.
2. **OOB Data**: The data points that are not included in a particular tree's bootstrap sample are referred to as out-of-bag samples. On average, about one-third of the data points will be OOB for any given tree.

The training instance not sampled are called out of bag evaluation.

Sampling is controlled by 2 hyperparameter max_features and bootstrap_features, these are used when dealing with High Dimensional Data to speed up the process.

Random Forest

A random forest is an ensemble learning method used for classification and regression tasks. It operates by constructing multiple decision trees during training and outputting the mode of the classes (for classification) or the mean prediction (for regression) of the individual trees. Here are the key features of random forests:

Key Concepts

1. **Ensemble Learning**: Combines multiple models to improve performance and robustness compared to a single model.
2. **Decision Trees**: The base learners in a random forest are decision trees, which make predictions based on features by splitting the data at various thresholds.
3. **Bootstrap Sampling**: Each tree is trained on a random subset of the data, created by sampling with replacement. This means some data points may be used multiple times in one tree while others may not be used at all.
4. **Random Feature Selection**: When splitting a node in a tree, a random subset of features is chosen instead of considering all features. This helps in reducing correlation among trees and improves the model's diversity.
5. **Voting/Averaging**: For classification, each tree votes for a class, and the class with the majority votes is chosen. For regression, the predictions from all trees are averaged.

Advantages

- **Robustness**: Less prone to overfitting compared to individual decision trees.
- **Handles High Dimensionality**: Effective even with a large number of features.
- **Feature Importance**: Can provide insights into feature importance, helping in model interpretation.

Limitations

- **Complexity**: More complex and less interpretable than single decision trees.
- **Computationally Intensive**: Can require more memory and processing power, especially with a large number of trees.

A forest of extremely random trees is called randomized tree.

Yet another great quality of random forests is that they make it easy to measure the relative importance of each feature.

Boosting

Boosting refers to any ensemble method that can combine several learners into a strong learner.

Boosting is a machine learning ensemble technique that combines multiple weak learners (often decision trees) to create a strong predictive model. The main idea is to sequentially train models, where each new model focuses on correcting the errors made by the previous ones. Here's a closer look at two popular boosting algorithms: AdaBoost and Gradient Boosting.

Boosting Overview

- **Weak Learners**: Typically, simple models (e.g., shallow decision trees) that perform slightly better than random guessing.
- **Sequential Learning**: Each model is trained based on the performance of the previous ones, focusing more on misclassified instances.
- **Weighting**: Boosting algorithms assign weights to training samples, increasing the importance of harder-to-predict samples.

1. AdaBoost (Adaptive Boosting)

- **Basic Idea**: Combines multiple weak classifiers to form a strong classifier. It adapts by focusing more on misclassified samples in subsequent iterations.
- **Process**:
    1. Initialize weights for each training sample.
    2. Train a weak learner (e.g., decision stump) and evaluate its performance.

3. Update the weights: Increase weights for misclassified samples and decrease weights for correctly classified samples.
4. Repeat the process for a specified number of iterations or until a certain error rate is achieved.
5. The final model is a weighted sum of all weak learners.

- **Advantages**:
  - Effective and simple to implement.
  - Works well with various types of data.
- **Limitations**:
  - Sensitive to noisy data and outliers.

2. Gradient Boosting

- **Basic Idea**: Builds models sequentially, with each new model attempting to correct the residual errors of the previous model. It uses a gradient descent approach to minimize a loss function.
- **Process**:
  1. Initialize the model with a constant value (often the mean of the target).
  2. Calculate the residuals (errors) of the current model.
  3. Fit a weak learner (e.g., a decision tree) to these residuals.
  4. Update the model by adding the new learner, scaled by a learning rate.
  5. Repeat the process until a specified number of models is reached or performance improves.
- **Advantages**:
  - Highly flexible and can optimize various loss functions.
  - Often provides better accuracy than AdaBoost.
- **Limitations**:
  - More complex and computationally intensive.
  - Requires careful tuning of hyperparameters (e.g., learning rate, tree depth).

Stacking

Stacking, or stacked generalization, is an ensemble learning technique that combines multiple models to improve predictive performance. The key idea is to use a "meta-model" to learn how to best combine the predictions from several base models. Here’s how it works:

Key Concepts of Stacking

1. **Base Models**: These are the initial models trained on the dataset. They can be of different types (e.g., decision trees, logistic regression, support vector machines) to leverage their diverse strengths.
2. **Meta-Model**: This is a second-level model that takes the predictions of the base models as input and learns to combine them optimally. The meta-model can also be a simple model, like linear regression, or more complex.

3. **Training Process**:
    - **First Layer**: Train the base models on the original training data.
    - **Second Layer**: To prevent overfitting, a common approach is to use cross-validation. For each base model, predictions are made on validation folds, creating a new dataset of predictions. This new dataset serves as the input for the meta-model.
    - **Final Prediction**: Once the meta-model is trained, it is used to make final predictions based on the outputs of the base models.

Advantages of Stacking

- **Improved Accuracy**: By combining multiple models, stacking often achieves better performance than any individual model.
- **Flexibility**: Different types of models can be combined, allowing for a more comprehensive approach to capturing patterns in the data.
- **Robustness**: Reduces the risk of overfitting by leveraging the strengths of different models.

Limitations

- **Complexity**: Stacking can be more complex to implement and tune compared to simpler ensemble methods like bagging or boosting.
- **Computational Cost**: Training multiple models and a meta-model can be computationally intensive, especially with large datasets.

Train a model to perform the aggregation.

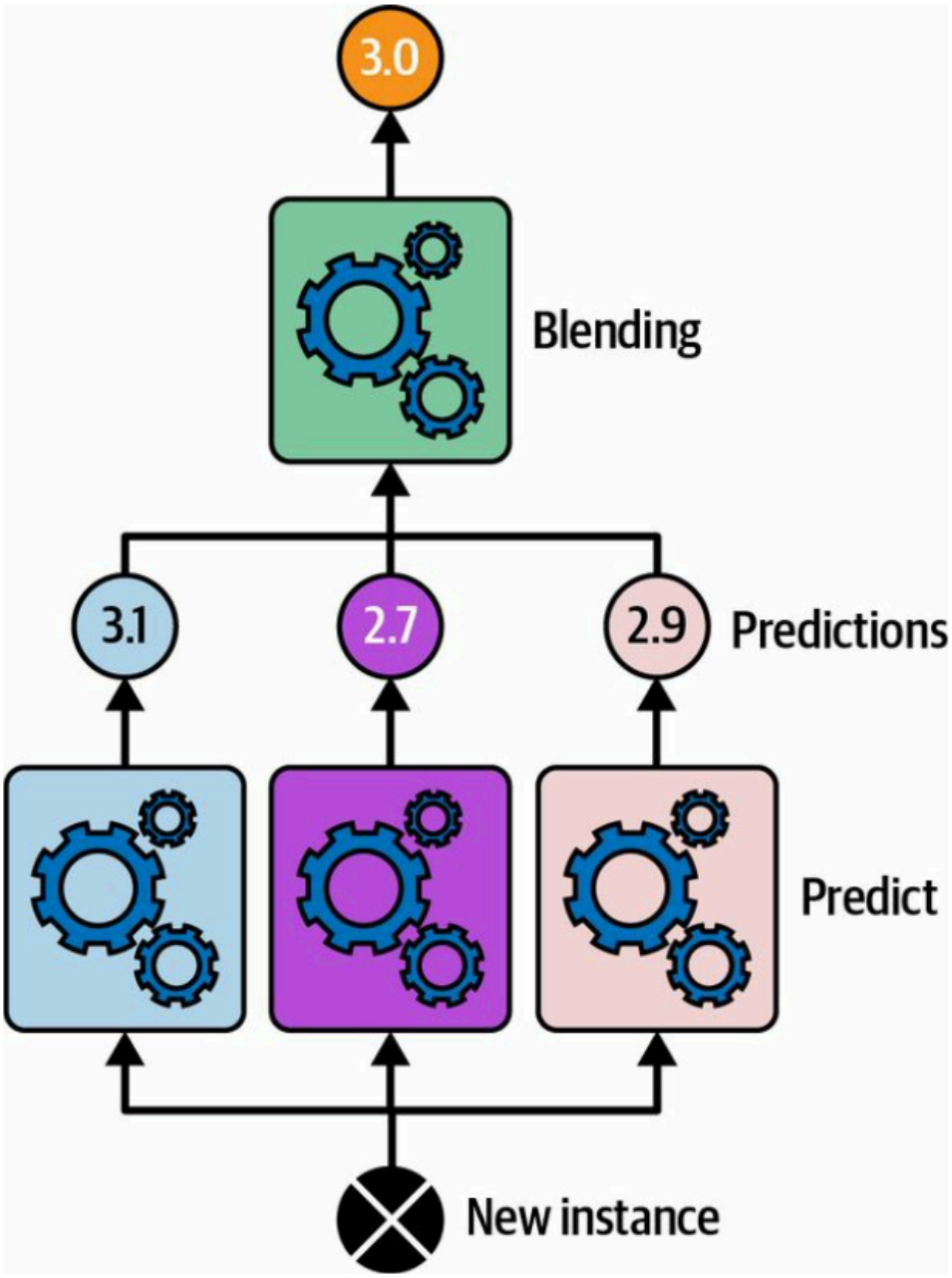

Figure 7-11. Aggregating predictions using a blending predictor