Improving Performance of ML Models

Ensemble learning helps improve machine learning results by combining several models. This approach allows the production of better predictive performance compared to a single model. That is why ensemble methods placed first in many prestigious machine learning competitions, such as the Netflix Competition, KDD 2009, and Kaggle.

Ensemble methods are meta-algorithms that combine several machine learning techniques into one predictive model in order to **decrease variance** (bagging), **bias** (boosting), or **improve predictions** (stacking).

Ensemble methods can be divided into two groups:

- **Sequential ensemble** methods where the base learners are generated sequentially (e.g. AdaBoost).The basic motivation of sequential methods is to **exploit the dependence between the base learners.** The overall performance can be boosted by weighing previously mislabeled examples with higher weight.
- **Parallel ensemble** methods where the base learners are generated in parallel (e.g. Random Forest).The basic motivation of parallel methods is to **exploit independence between the base learners** since the error can be reduced dramatically by averaging.

  - Most ensemble methods use a single base learning algorithm to produce homogeneous base learners, i.e. learners of the same type, leading to *homogeneous ensembles*.
  - There are also some methods that use heterogeneous learners, i.e. learners of different types, leading to *heterogeneous ensembles*. In order for ensemble methods to be more accurate than any of its individual members, the base.

Bagging

Bagging stands for bootstrap aggregation. One way to reduce the variance of an estimate is to average together multiple estimates. For example, we can train M different trees on different subsets of the data (chosen randomly with replacement) and compute the ensemble:

$$f(x) = 1/M \sum_{m=1}^{M} f_m(x)$$

Bagging uses bootstrap sampling to obtain the data subsets for training the base learners. For aggregating the outputs of base learners, bagging uses voting for classification and averaging for regression.

The following are three bagging ensemble algorithms −

- Bagged Decision Tree
- Random Forest
- Extra Trees

Boosting

Boosting refers to a family of algorithms that are able to convert weak learners to strong learners. The main principle of boosting is to fit a sequence of weak learners− models that are only slightly better than random guessing, such as small decision trees− to weighted versions of the data. More weight is given to examples that were misclassified by earlier rounds.The predictions are then combined through a weighted majority vote (classification) or a weighted sum (regression) to produce the final prediction. The principal difference between boosting and the committee methods, such as bagging, is that base learners are trained in sequence on a weighted version of the data.

The followings are the two most common boosting ensemble algorithms −

- AdaBoost
- Stochastic Gradient Boosting

Voting

In this ensemble learning model, multiple models of different types are built and some simple statistics,

like calculating mean or median etc., are used to combine the predictions. This prediction will serve as the additional input for training to make the final prediction.

Voting Ensemble Algorithms

Import Libraries:

```
from pandas import read_csv
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import VotingClassifier
```

Now, we need to load the Pima diabetes dataset as did in previous examples −

```
path = r"C:\pima-indians-diabetes.csv"
headernames = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
data = read_csv(path, names = headernames)
array = data.values
X = array[:,0:8]
Y = array[:,8]
```

Next, give the input for 10-fold cross validation as follows −Next, give the input for 10-fold cross validation as follows −

```
kfold = KFold(n_splits = 10, random_state = 7)
```

Next, we need to create sub-models as follows –

```
estimators = []
model1 = LogisticRegression()
estimators.append(('logistic', model1))
model2 = DecisionTreeClassifier()
estimators.append(('cart', model2))
model3 = SVC()
estimators.append(('svm', model3))
```

Now, create the voting ensemble model by combining the predictions of above created sub models.

```
ensemble = VotingClassifier(estimators)
results = cross_val_score(ensemble, X, Y, cv = kfold)
print(results.mean())
```

Output

```
0.7382262474367738
```

The output above shows that we got around 74% accuracy of our voting classifier ensemble model.