Machine Learning

| Ensemble Learning

Prof. Dr. Mostafa Elhosseini

https://youtube.com/ElhosseiniAcademy

Agenda

- Inspiration
- Wisdom of the crowd
- Voting Classifier
- Hard Voting
- Soft Voting

Wisdom of the crowd

- **Complex question** to thousands of random people, then **aggregate** their answers. In many cases you will find that this aggregated answer is better than an expert's answer.

- This is called the **wisdom of the crowd**.

Ensemble Learning

- 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**

Ensemble Learning

- 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**

Ensemble Learning

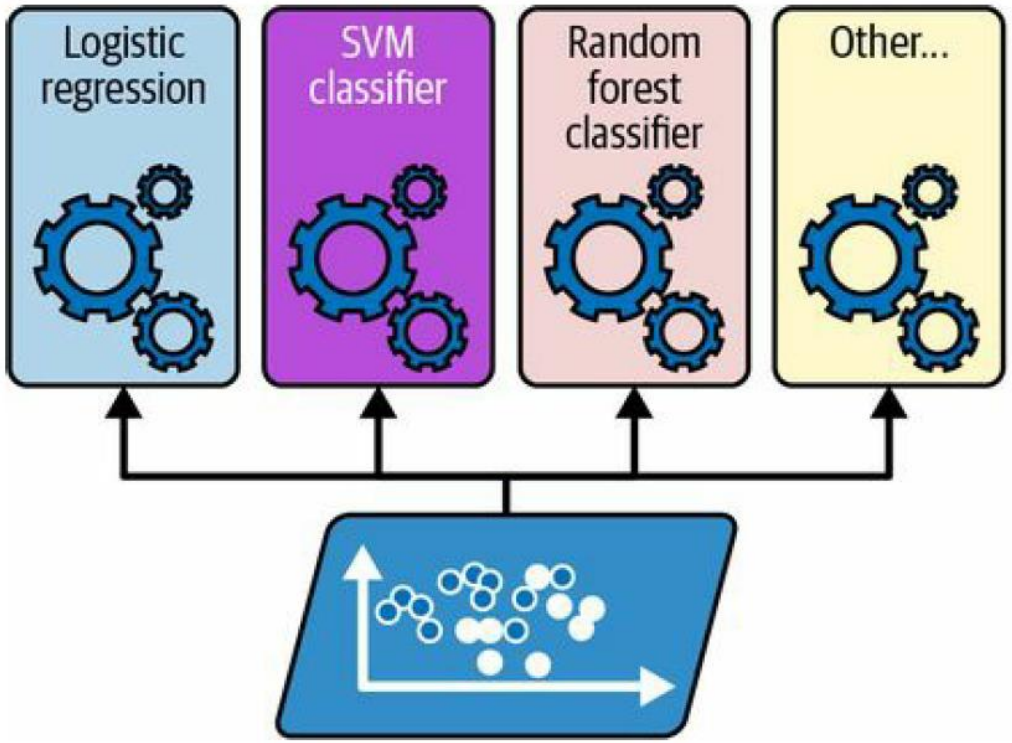
- Such an ensemble of decision trees is called a **random forest**, and despite its **simplicity**, this is one of the **most powerful** machine learning algorithms available today.

- you will often use ensemble methods **near the end of a project**, once you have already built a few good predictors, to combine them into an even better predictor. In fact, the **winning solutions** in machine learning competitions often involve several ensemble methods—most famously in the **Netflix Prize competition**.

Voting Classifiers

- The class that gets the most votes is the ensemble's prediction.

- This **majority-vote** classifier is called a **hard voting** classifier



Hard voting

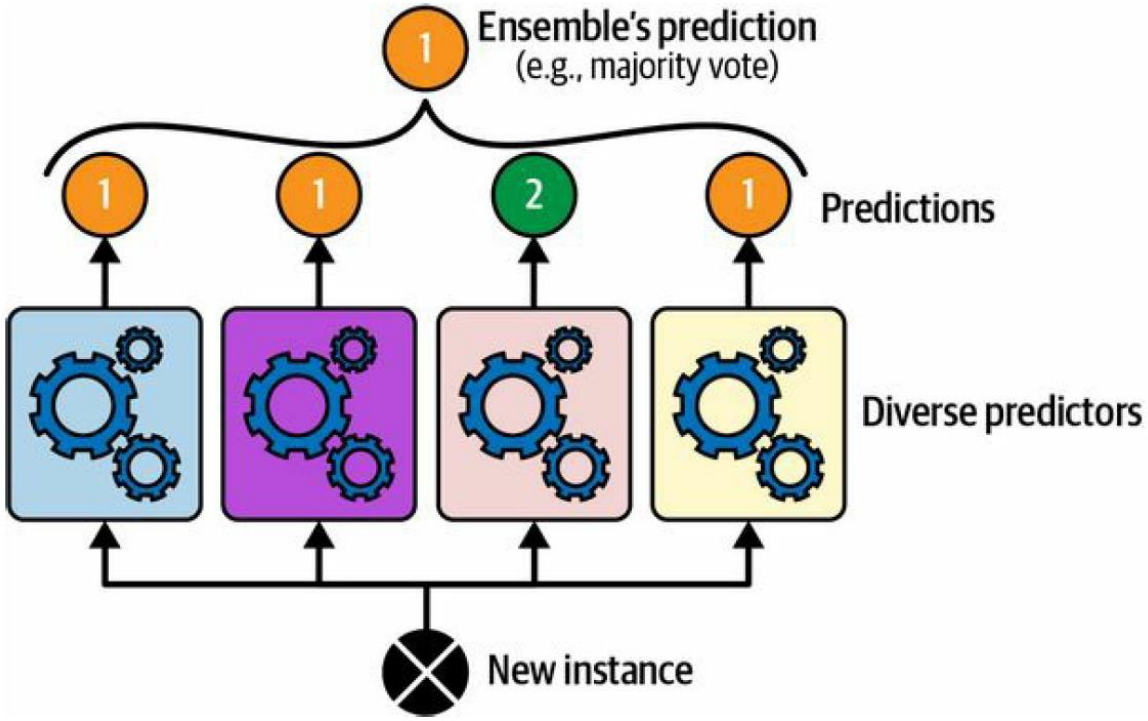
- In Hard voting, each classifier in an ensemble makes **its own class prediction** (i.e., assigns a class label), and the **final prediction** is determined by a **simple majority vote**. The class that receives the **most votes** from the individual classifiers is the one predicted by the ensemble.
- Suppose you have **three classifiers** predicting the following classes for an instance:

  Classifier 1: Class A **Classifier 2: Class B** **Classifier 3: Class A**

- Result: **Class A receives 2 votes;** Class B receives 1 vote. The ensemble predicts Class A.

Hard voting

- **Voting classifier** often achieves a higher accuracy than the best classifier in the ensemble.
- In fact, even if each classifier is a **weak learner** (meaning it does only slightly better than **random guessing**), 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.



Hard voting

- The following **analogy** can help shed some light on this mystery
- Slightly biased coin that has a 51% chance of coming up heads
- Toss it 1,000 times, you will generally get more or less 510 heads and 490 tails, and hence a majority of heads
- The probability of obtaining a majority of heads after 1,000 tosses is close to 75%
- Law of large numbers

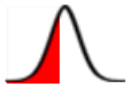
Hard voting

- Law of large numbers
- **Binomial Distribution**
- $n = 1000$
- $P = 0.51$
- $P(x >= 501) = 0.7261$

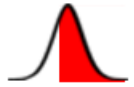
- **Normal Distribution**
- Mean = $np = 510$
- $\sigma^2 = np(1-p) = 249.9$
- $\sigma = 15.81$

Z-score = -0.56926

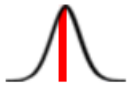
Probability of x<501: 0.28459



Probability of x>501: 0.71541



Probability of 501<x<510: 0.21541



Hard voting

- Similarly, suppose you build an ensemble containing **1,000 classifiers** that are individually correct only **51%** of the time (barely better than **random guessing**). If you predict the majority voted class, you can hope for up to **75% accuracy**! However, this is only true if all classifiers are **perfectly independent**, making **uncorrelated errors**, which is clearly not the case because they are **trained on the same data**.
- They are likely to make the **same types of errors**, so there will be many majority votes for the wrong class, reducing the ensemble's accuracy.

Hard voting

- Ensemble methods work best when the predictors are as **independent** from one another as possible.

- One way to get diverse classifiers is to train them using very **different algorithms**. This increases the chance that they will make very different types of errors, improving the ensemble's accuracy.

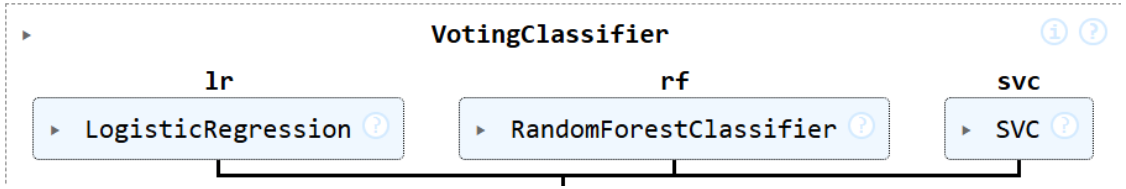
Hard voting

- VotingClassifier class – Moons dataset

```
from sklearn.datasets import make_moons
from sklearn.ensemble import RandomForestClassifier, VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC

X, y = make_moons(n_samples=500, noise=0.30, random_state=42)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)

voting_clf = VotingClassifier(
    estimators=[
        ('lr', LogisticRegression(random_state=42)),
        ('rf', RandomForestClassifier(random_state=42)),
        ('svc', SVC(random_state=42))
    ]
)
voting_clf.fit(X_train, y_train)
```



Hard voting

- VotingClassifier class – Moons dataset
  - Fitted classifier's accuracy

```
for name, clf in voting_clf.named_estimators_.items():
    print(name, "=", clf.score(X_test, y_test))
```

```
lr = 0.864
rf = 0.896
svc = 0.896
```

  - When you call the **voting classifier's** predict() method, it performs **hard voting**

```
voting_clf.predict(X_test[:1])
```

```
array([1])
```

Hard voting

- VotingClassifier class – Moons dataset
  - two out of three classifiers predict that class

```
[clf.predict(X_test[:1]) for clf in voting_clf.estimators_]
```

```
[array([1]), array([1]), array([0])]
```

  - Performance of the voting classifier on the test set

```
voting_clf.score(X_test, y_test)
```

```
0.912
```

Soft Voting

- In soft voting, each classifier **provides a probability** (or confidence score) for each class rather than just a class label. These probabilities are averaged (or summed) across all classifiers, and the final prediction is made by selecting the class with the highest average probability.
- It often achieves **higher performance** than hard voting because it gives **more weight to highly confident votes**. All you need to do is set the voting classifier's voting hyperparameter to "**soft**", and ensure that all classifiers can estimate class probabilities

Soft Voting

- We reach 92% accuracy simply by using soft voting

```
voting_clf.voting = "soft"
voting_clf.named_estimators["svc"].probability = True
voting_clf.fit(X_train, y_train)
voting_clf.score(X_test, y_test)
```

```
0.92
```

Next Lecture

- Most popular ensemble methods, including voting classifiers, bagging and pasting ensembles, random forests, and boosting, and stacking ensembles