Ensemble Learning

مقدمه :

فرض کنید از هزاران فرد ، تصادفی سوالی پیچیده می‌پرسید ، سپس پاسخ آن ها را جمع می‌کنید. در بسیاری از موارد متوجه خواهید شد که این پاسخ جمع شده بهتر از پاسخ کارشناس است. به این خرد جمعی می‌گویند. به همین ترتیب اگر پیش بینی‌های گروهی از پیش بینی کننده‌ها مانند طبقه‌بندی کننده یا رگرسیون را جمع کنید ، اغلب پیش بینی‌های بهتری نسبت به بهترین پیش بینی کننده فردی خواهید داشت.

گروهی از پیش بینی کننده‌ها را جمعی (Ensemble) می‌نامند. بنابراین این تکنیک ، یادگیری جمعی (Ensemble Learning) و الگوریتم آن روش جمعی نامیده می‌شود. به عنوان مثال می‌توانید گروهی از طبقه بندی کننده‌های درخت تصمیم را در زیر مجموعه تصادفی متفاوت از مجموعه آموزش ، آموزش دهید. برای پیش بینی فقط پیش بینی تمام درختان را به دست می‌آورید ، سپس کلاسی را که بیشترین رای را کسب کرده پیش بینی می‌کنید. چنین مجموعه‌ای از درختان تصمیم جنگل تصادفی (Random Forest) نامیده می‌شود و به رغم سادگی یکی از قدرتمندترین الگوریتم‌های یادگیری ماشین است که امروز وجود دارد.

Random Forest :

همان طور که پیش تر به آن پرداختیم ، جنگل های تصادفی مجموعه ای از درختان تصمیم گیری است.

الگوریتم Random Forest حالت تصادفی اضافی را هنگام رشد درختان معرفی و به جای جستجوی بهترین ویژگی هنگام تقسیم گره آن را در میان زیر مجموعه های تصادفی ویژگی ها جستجو می کند. این امر منجر به تنوع بیشتر درخت می شود که به طور کلی باعث می شود مدل بهتری ارائه شود.

برای مثال اگر بخواهیم پروژه iris را با استفاده از جنگل تصادفی پیاده سازی کنیم ، داریم :

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
```

```
def load_data():
    DataSet = pd.read_csv("iris.csv", header = None,
                          names = ["sepal length",
                                   "sepal width",
                                   "petal length",
                                   "petal width",
                                   "label"])

    data = DataSet.iloc[:, :4]
    label = DataSet.iloc[:, 4]

    return data, label
```

```
data, label = load_data()

X_train, X_test, y_train, y_test = train_test_split(data, label,
                                                    test_size=0.2,
                                                    random_state=42)
rnd_clf = RandomForestClassifier(n_estimators=500,
                                 max_leaf_nodes=16, n_jobs=-1)
rnd_clf.fit(X_train, y_train)
```

```
y_pred = rnd_clf.predict(X_test)
print(classification_report(y_test, y_pred))
```

❖ اهمیت ویژگی :

از آن جایی که درخت تصمیم برای هر گره یک شرطی را مبتنی بر مقدار یکی از ویژگی ها در نظر می گیرد ، می توان اهمیت ویژگی ها را نسبت به هم بدست آورد. بعد از آموزش مدل ، درخت تصمیم برای هر ویژگی یک امتیازی را نسبت به اهمیت آن ویژگی در نظر می گیرد که در نهایت مجموع تمام این امتیاز ها برابر یک می شود.

برای پروژه iris داریم :

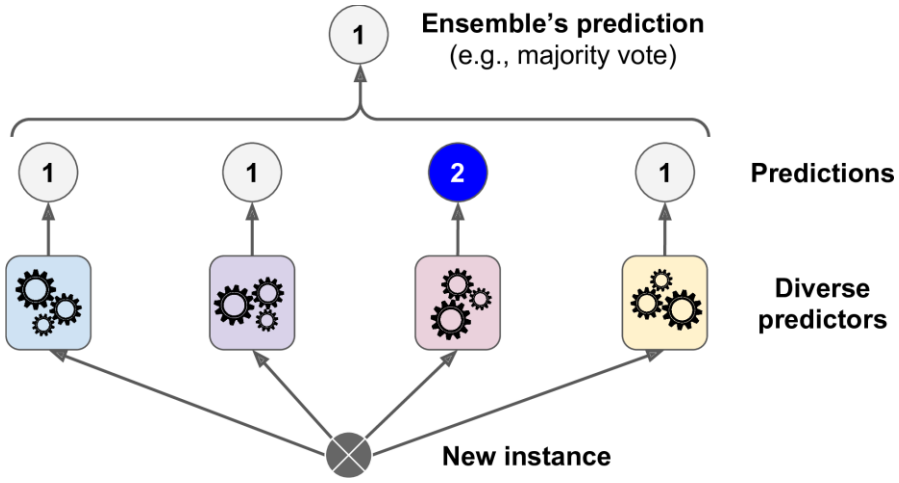
```
feature_names = ["sepal length", "sepal width",
                 "petal length", "petal width"]

for name, score in zip(feature_names, rnd_clf.feature_importances_):
    print(name, ":", score)
```

Voting Classifiers :

فرض کنید چند طبقه بندی کننده را آموزش داده اید که هر کدام از آن ها حدود ۸۰ درصد دقت دارند. ممکن است شما یک طبقه بندی کننده رگرسیون لجستیک ، طبقه بندی کننده SVM ، جنگل تصادفی ، KNN و ... داشته باشید.

روش بسیار ساده برای ایجاد طبقه بندی کننده بهتر ، جمع آوری پیش بینی های هر طبقه بندی و پیش بینی کلاسی است که بیشترین رای را دارد. به این طبقه بندی اکثریت آرا ، طبقه بندی کننده رای گیری سخت یا hard voting classifier گفته می شود.

این طبقه بندی کننده رای گیری ، اغلب از دقت بالاتری نسبت به بهترین طبقه بندی کننده گروه برخوردار است. در حقیقت حتی اگر هر طبقه بندی کننده یاد گیرنده ای ضعیف باشد ، جمعی هنوز هم می تواند آموزنده ای قوی باشد به شرط آن که تعداد کافی یاد گیرنده ضعیف وجود داشته باشد و به اندازه کافی متنوع باشند.



برای مثال اگر بخواهیم Voting Classifiers را پیاده سازی کنیم ، داریم :

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

```
data, label = make_moons(n_samples=500, noise=0.30, random_state=42)
X_train, X_test, y_train, y_test = train_test_split(data, label,
                                                    random_state=42)
```

```
log_clf = LogisticRegression()
svm_clf = SVC()
rnd_clf = RandomForestClassifier()
```

```
voting_clf = VotingClassifier(estimators=[("lr", log_clf),
                                          ("svc", svm_clf),
                                          ("rf", rnd_clf)],
                              voting="hard")
voting_clf.fit(X_train, y_train)
```

```
for clf in (log_clf, svm_clf, rnd_clf, voting_clf):
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    print(f"{clf.__class__.__name__} : {accuracy_score(y_test, y_pred)}")
```

نکته ! اگر همه طبقه بندی کننده ها بتوانند احتمالات کلاس را تخمین بزنند بنابراین می توان کلاس با بالاترین احتمال در کل طبقه بندی کننده ها را پیش بینی کرد. به این امر رای گیری نرم گفته می شود. حتما اطمینان حاصل کنید که طبقه بندی کننده ها می توانند احتمال را تخمین بزنند.