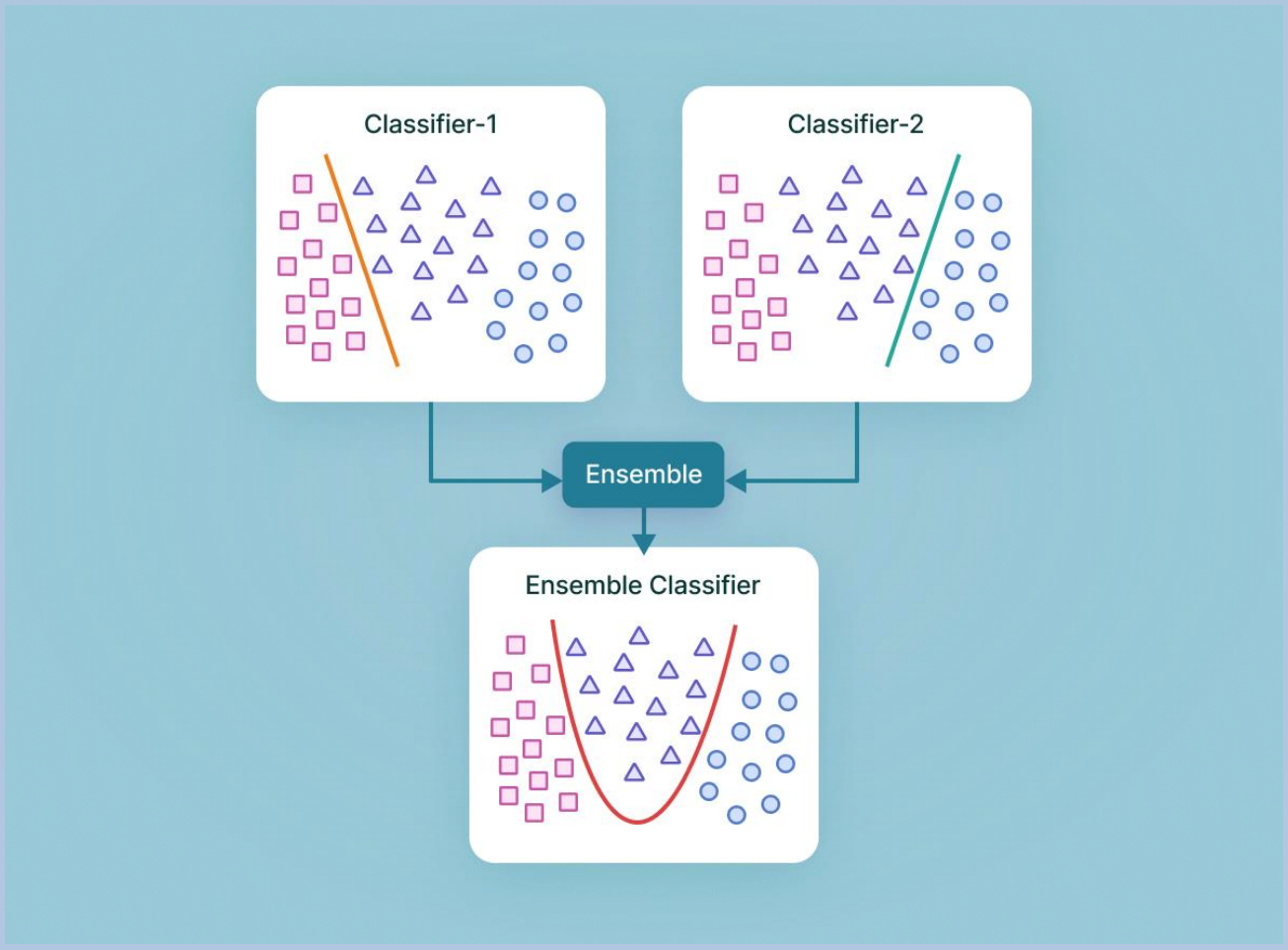
GJU
الجامعة الألمانية الأردنية
German Jordanian University

SEEIT

Ensemble Learning



Rana Husni

GJU
الجامعة الألمانية الأردنية
German Jordanian University

SEEIT

Suppose you wanted to purchase a car.Now by just visiting the first car company and based on the dealer's advise will we straight away make a buy on a car? Answer is defenitely a big NO right?

So what we do is first decide whether which car to buy ,whether it is a new or used car ,type of car,model and year of manufacture, look for list of dealers ,look for discounts/offers ,customer reviews,opinion from friends and family, performance ,fuel efficiency and obvious any car buyer will for the best price range etc.

In short, you wouldn’t directly reach a conclusion, but will instead make a decision considering all the above mentioned factors before we decide on the best choice.

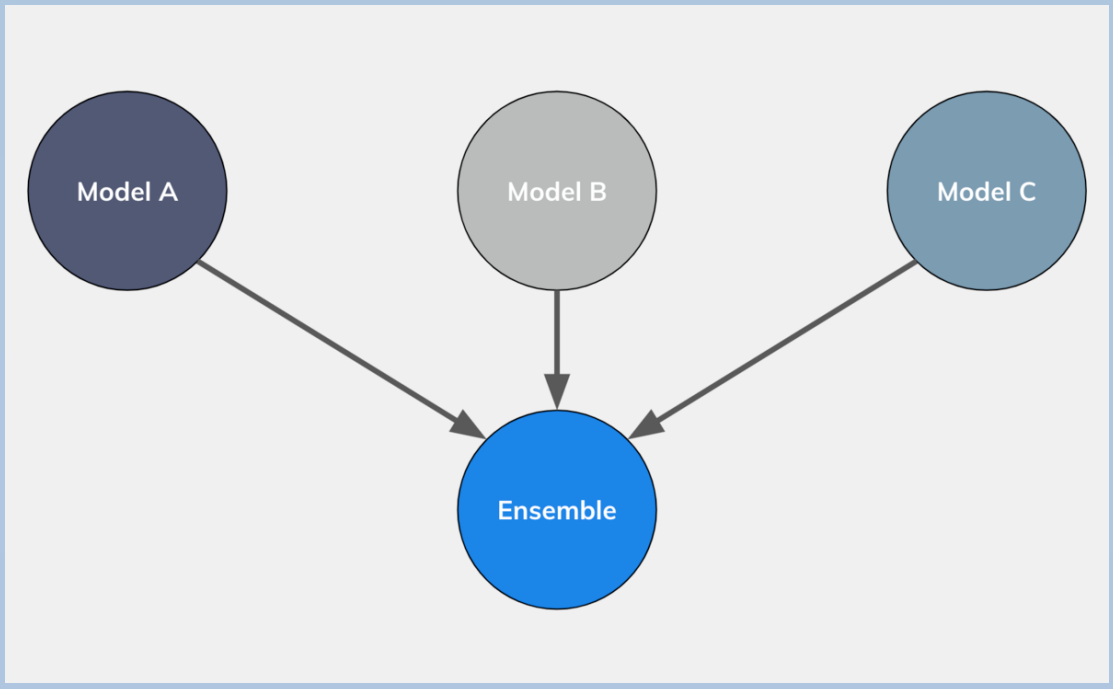
Rana Husni

GJU
الجامعة الألمانية الأردنية
German Jordanian University

SEEIT

Ensemble models

Ensemble Learning helps improve machine learning results by combining several models to improve predictive performance compared to a single model.



Simple Ensemble Techniques

- Max Voting
- Averaging
- Weighted Averaging

Rana Husni

GJU
الجامعة الألمانية الأردنية
German Jordanian University

SEEIT

Max Voting

The max voting method is generally used for classification problems. In this technique, multiple models are used to make predictions for each data point. The predictions by each model are considered as a ‘vote’. The predictions which we get from the majority of the models are used as the final prediction.

For example, when you asked 5 of your colleagues to rate your movie (out of 5); we’ll assume three of them rated it as 4 while two of them gave it a 5. Since the majority gave a rating of 4, the final rating will be taken as 4. You can consider this as taking the mode of all the predictions.

The result of max voting would be something like this:

Colleague 1	Colleague 2	Colleague 3	Colleague 4	Colleague 5	Final rating
5	4	5	4	4	4

Rana Husni

GJU
الجامعة الألمانية الأردنية
German Jordanian University

SEEIT

Heart Dataset

This database contains 76 attributes, but all published experiments refer to using a subset of 14 of them. In particular, the Cleveland database is the only one that has been used by ML researchers to this date. The "goal" field refers to the presence of heart disease in the patient. It is integer valued from 0 (no presence) to 4.

The following link includes dataset's details
https://colab.research.google.com/drive/16iFRPq0vx_CZypo4ZyJ_qTLrvds3FlDb?usp=sharing

Rana Husni

Voting In Python

Here x_train consists of independent variables in training data, y_train is the target variable for training data. The validation set is x_test (independent variables) and y_test (target variable) .

```
# IMPORTS
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
import statistics as st
from sklearn.metrics import
confusion_matrix,accuracy_score,precision_score,recall_score,f1_score

# SPLITTING THE DATASET
df = pd.read_csv('heart.csv')
x = df.iloc[:,0:13]
y = df['target']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
# MODELS CREATION
model1 = DecisionTreeClassifier()
model2 = KNeighborsClassifier()
model3= RandomForestClassifier()

model1.fit(x_train,y_train)
model2.fit(x_train,y_train)
model3.fit(x_train,y_train)
```

Rana Husni


```
# PREDICTION
pred1=model1.predict(x_test)
pred2=model2.predict(x_test)
pred3=model3.predict(x_test)
ac=accuracy_score(pred1,y_test)
print("Model 1 Accuracy=",ac)
ac=accuracy_score(pred2,y_test)
print("Model 2 Accuracy=",ac)
ac=accuracy_score(pred3,y_test)
print("Model 3 Accuracy=",ac)

# FINAL_PREDICTION
final_pred = np.array([])
for i in range(0,len(x_test)):
    final_pred = np.append(final_pred, st.mode([pred1[i], pred2[i],
pred3[i]]))
print(final_pred)


ac=accuracy_score(final_pred,y_test)
print("final_pred Max_Voting Accuracy=",ac)
```

Voting Classifier

SEEIT

```
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.ensemble import VotingClassifier

# SPLITTING THE DATASET
df = pd.read_csv('heart.csv')
x = df.iloc[:,0:13]
y = df['target']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
# MODELS CREATION
model1 = DecisionTreeClassifier()
model2 = KNeighborsClassifier()
model3= RandomForestClassifier()
models = [('Decision Tree', DecisionTreeClassifier()),
        ('KNN', KNeighborsClassifier()),
        ('Random Forest Classifier', RandomForestClassifier())]
for name, model in models:
    model.fit(x_train, y_train)
    prediction = model.predict(x_test)
    score = accuracy_score(y_test, prediction)
    print('{} Model Accuracy: {}'.format(name,score))
    ensemble = VotingClassifier(estimators=models)
ensemble.fit(x_train, y_train)
prediction = ensemble.predict(x_test)
score = accuracy_score(y_test, prediction)
print('Ensemble Model Accuracy: {}'.format(score))
```

Averaging

Similar to the max voting technique, multiple predictions are made for each data point in averaging. In this method, we take an average of predictions from all the models and use it to make the final prediction. Averaging can be used for making predictions in regression problems or while calculating probabilities for classification problems.

For example, in the below case, the averaging method would take the average of all the values.

i.e. (5+4+5+4+4)/5 = 4.4

Colleague 1	Colleague 2	Colleague 3	Colleague 4	Colleague 5	Final rating
5	4	5	4	4	4.4


```
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

# SPLITTING THE DATASET
df = pd.read_csv('heart.csv')
x = df.iloc[:,0:13]
y = df['target']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
# MODELS CREATION
model1 = DecisionTreeClassifier()
model2 = KNeighborsClassifier()
model3= RandomForestClassifier()

model1.fit(x_train,y_train)
model2.fit(x_train,y_train)
model3.fit(x_train,y_train)
# PREDICTION
pred1=model1.predict_proba(x_test)
pred2=model2.predict_proba(x_test)
pred3=model3.predict_proba(x_test)
finalpred=(pred1+pred2+pred3)/3
```

GJU
الجامعة الألمانية الأردنية
German Jordanian University

SEEIT

Weighted Average

This is an extension of the averaging method. All models are assigned different weights defining the importance of each model for prediction. For instance, if two of your colleagues are critics, while others have no prior experience in this field, then the answers by these two friends are given more importance as compared to the other people.

The result is calculated as [(5*0.23) + (4*0.23) + (5*0.18) + (4*0.18) + (4*0.18)] = 4.41.

	Colleague 1	Colleague 2	Colleague 3	Colleague 4	Colleague 5	Final rating
weight	0.23	0.23	0.18	0.18	0.18	
rating	5	4	5	4	4	4.41

Rana Husni


```
# IMPORTS
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

# SPLITTING THE DATASET
df = pd.read_csv('heart.csv')
x = df.iloc[:,0:13]
y = df['target']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
# MODELS CREATION
model1 = DecisionTreeClassifier()
model2 = KNeighborsClassifier()
model3= RandomForestClassifier()

model1.fit(x_train,y_train)
model2.fit(x_train,y_train)
model3.fit(x_train,y_train)
# PREDICTION
pred1=model1.predict_proba(x_test)
pred2=model2.predict_proba(x_test)
pred3=model3.predict_proba(x_test)


finalpred=(pred1*0.3+pred2*0.3+pred3*0.4)
```

GJU
الجامعة الألمانية الأردنية
German Jordanian University

SEEIT

Bagging Algorithm

Bagging is a supervised learning technique that can be used for both regression and classification tasks. Here is an overview of the steps including Bagging classifier algorithm:

•**Bootstrap Sampling:** Divides the original training data into 'N' subsets and randomly selects a subset with replacement in some rows from other subsets. This step ensures that the base models are trained on diverse subsets of the data and there is no class imbalance.

•**Base Model Training**: For each bootstrapped sample, train a base model independently on that subset of data. These weak models are trained in parallel to increase computational efficiency and reduce time consumption.

•**Prediction Aggregation:** To make a prediction on testing data combine the predictions of all base models. For classification tasks, it can include majority voting or weighted majority while for regression, it involves averaging the predictions.

•**Out-of-Bag (OOB) Evaluation**: Some samples are excluded from the training subset of particular base models during the bootstrapping method. These "out-of-bag" samples can be used to estimate the model's performance without the need for cross-validation.

•**Final Prediction:** After aggregating the predictions from all the base models, Bagging produces a final prediction for each instance.


```
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import numpy as np
import pandas as pd

# Load the Iris dataset
df = pd.read_csv('heart.csv')
x = df.iloc[:,0:13]
y = df['target']
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2,
random_state=42)
# Create a base classifier (e.g., Decision Tree)
base_classifier = DecisionTreeClassifier()
bagging_classifier = BaggingClassifier(base_classifier, n_estimators=10,
random_state=42)
bagging_classifier.fit(X_train, y_train)
y_pred = bagging_classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```