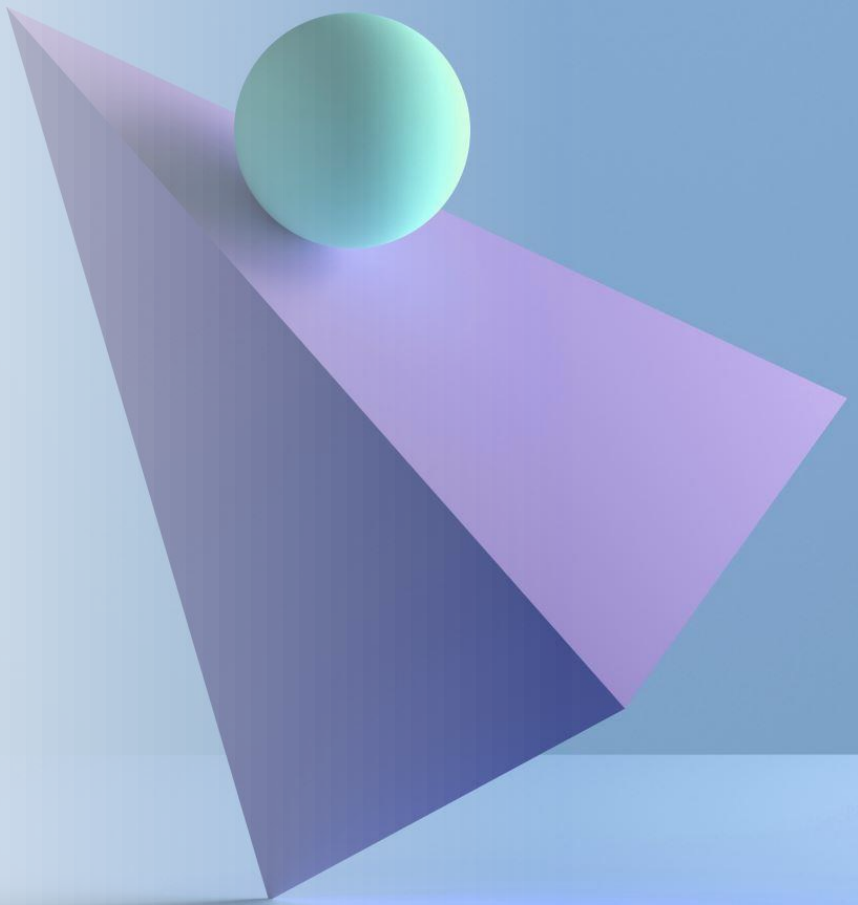
Ensemble Methods

MSc. Bui Quoc Khanh

khanhbq@hanu.edu.vn



Ensemble Classifiers

- Not yet another induction algorithm
- Ensembles combine multiple base classifiers to form a (hopefully) better classifier, thereby decreasing variability/variance and bias, and boosting performance.
- An ensemble classifier is a meta classifier: a classifier that doesn't implement a classification algorithm on its own, but uses other base classifiers to do the actual work
- Basic idea: Construct a set of base classifiers (experts) from the training data and let them vote - predict class label of previously unseen records by aggregating predictions made by multiple classifiers

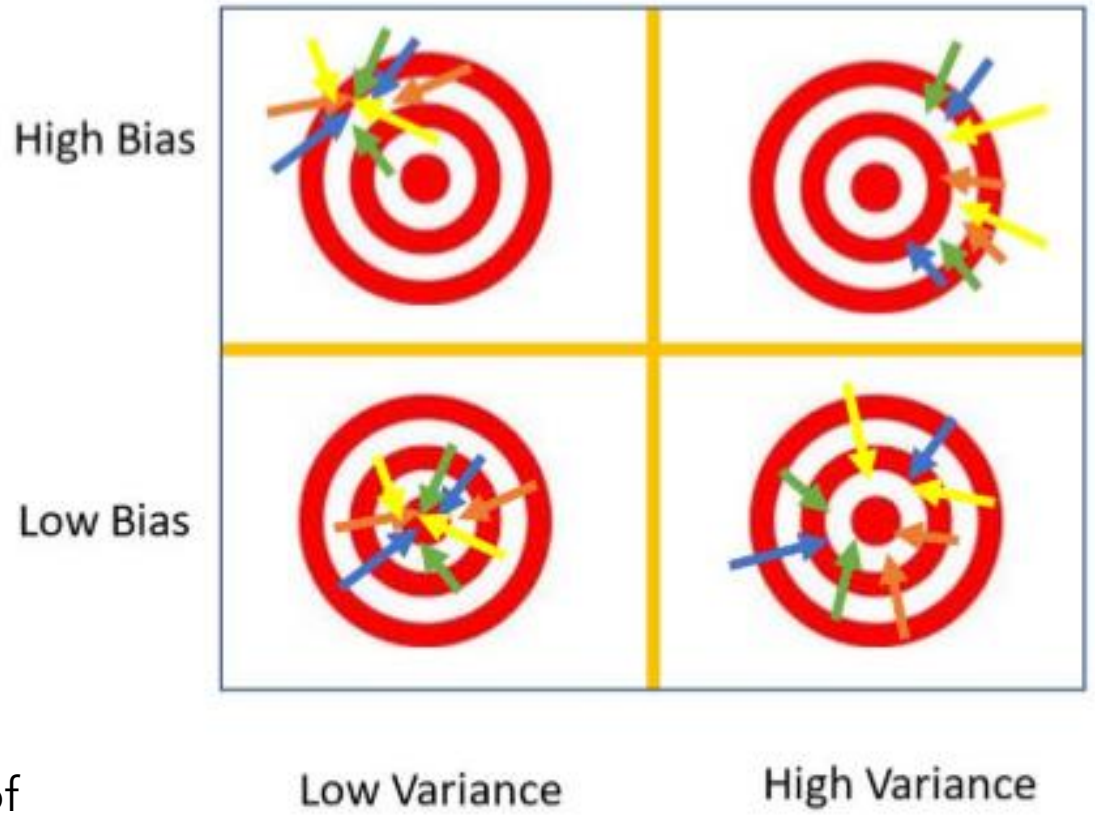
Bias and Variance

Variance

- the measure of how spread out data is

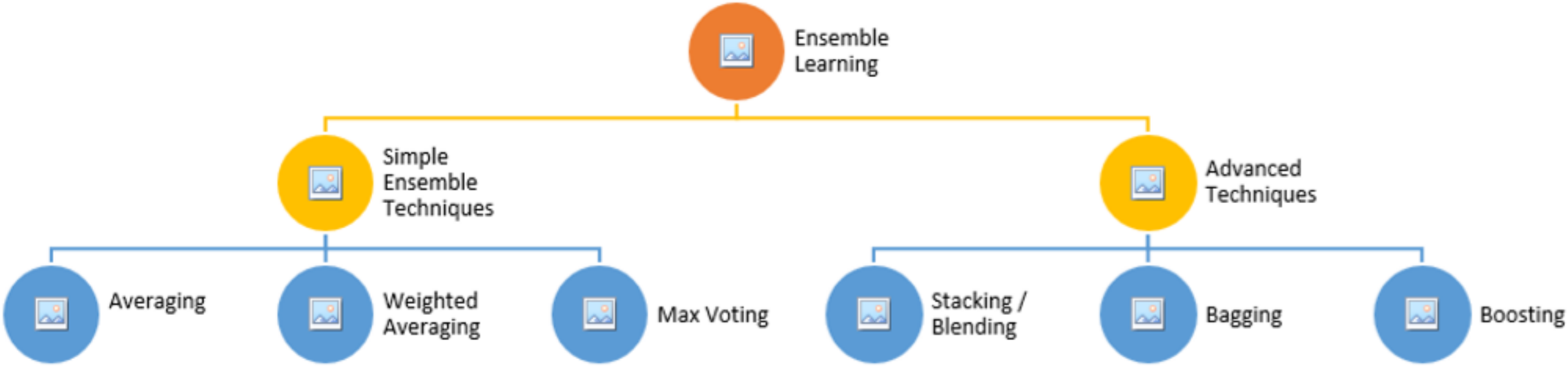
Bias

- the difference between the ground truth and the average value of our predictions



Ensemble models exemplify the adage the “wisdom of the crowds”

Different ensemble learning methods



Simple Methods for Ensemble Learning

- Averaging
- Weighted averaging
- Max voting

Averaging

- Basic idea: take the predictions of multiple individual models and then average the predictions to generate a final prediction
- Averaging the predictions => eliminate the errors made by individual learners, thereby generating a model superior to the base model
- Prerequisite: predictions of the base models must be uncorrelated

Averaging - Example

```
model1 = LogisticRegression(random_state=123)
model2 = KNeighborsClassifier(n_neighbors=5)
model3 = RandomForestClassifier(n_estimators=500)
```

```
# Fitting all three models on the training data
model1.fit(X_train,y_train)
model2.fit(X_train,y_train)
model3.fit(X_train,y_train)
```

Example	Class: '1'	Class: '0'
Test set example 1	0.902	0.098
Test set example 2	0.401	0.559
Test set example 3	0.732	0.268

```
# Predicting probabilities of each model on the test set
pred1=model1.predict_proba(X_test)
pred2=model2.predict_proba(X_test)
pred3=model3.predict_proba(X_test)
```

```
# Calculating the ensemble prediction by averaging three base model predictions
ensemblepred=(pred1+pred2+pred3)/3
```

```
import numpy as np
pred = np.argmax(ensemblepred,axis = 1)
```

Weighted Averaging

- an extension of the averaging method
- assign weights to each model's predictions and then generate the combined predictions
- weights are assigned based on our judgment of which model would be the most influential in the ensemble
- These weights, which are initially assigned arbitrarily, have to be evolved after a lot of experimentation

Weighted Averaging - Example

```
# Calculating the ensemble prediction by applying weights for each prediction
ensemblepred=(pred1 *0.60 + pred2 * 0.20 + pred3 * 0.20)
```

	precision	recall	f1-score	support
0	0.92	0.88	0.90	107
1	0.86	0.91	0.89	89
accuracy			0.89	196
macro avg	0.89	0.89	0.89	196
weighted avg	0.89	0.89	0.89	196

```
# Calculating the ensemble prediction by applying weights for each prediction
ensemblepred=(pred1 *0.70+pred2 * 0.15+pred3 * 0.15)
```

	precision	recall	f1-score	support
0	0.93	0.88	0.90	107
1	0.86	0.92	0.89	89
accuracy			0.90	196
macro avg	0.90	0.90	0.90	196
weighted avg	0.90	0.90	0.90	196

Max Voting

- Works on the principle of majority rule.
- Each individual learner's prediction is considered to be a vote.
- On the test set, whichever class gets the maximum vote is the ultimate winner

Max Voting - Example

Test Example	Prediction of learner 1	Prediction of learner 2	Prediction of learner 3	Final Prediction
Example 1	1	1	1	1
Example 2	1	0	0	0
Example 3	1	1	0	1
Example 4	0	1	0	0

```
# Defining the ensemble model using VotingClassifier
model = VotingClassifier(estimators=[('lr', model1),('knn', model2),('rf', model3)],
voting= 'hard')
```

```
              precision    recall  f1-score   support

           0       0.93      0.90      0.91       107
           1       0.88      0.92      0.90        89

    accuracy                           0.91       196
   macro avg       0.91      0.91      0.91       196
weighted avg       0.91      0.91      0.91       196
```

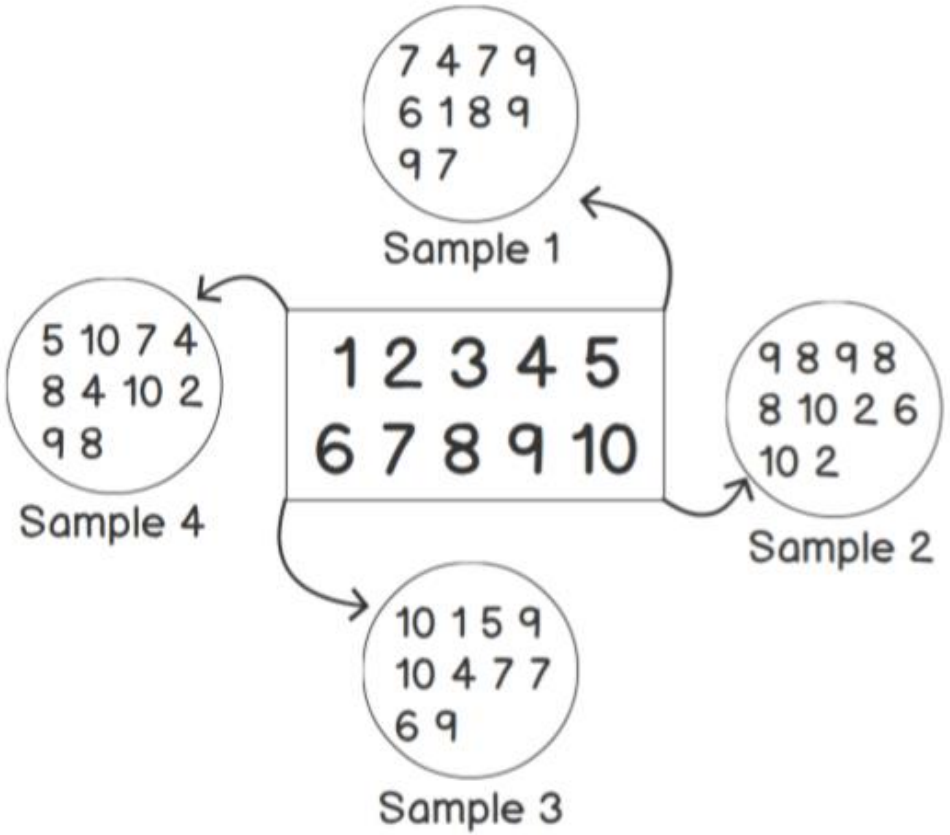
Advanced Techniques for Ensemble Learning

- Bagging
- Boosting
- Stacking/blending

Bagging

- A pseudonym for **B**ootstrap **Agg**regat**ing**
- Bootstrapping: to make the best use of the available resources
- In statistical context, bootstrapping entails taking samples from the available dataset by replacement

Bagging – Example



Bagging - Example

```
# Defining the base learner
from sklearn.ensemble import RandomForestClassifier
bl1 = RandomForestClassifier(random_state=123)
```

```
# Creating the bagging meta learner
from sklearn.ensemble import BaggingClassifier
baggingLearner = BaggingClassifier(base_estimator=bl1, n_estimators=10, max_
samples=0.8, max_features=0.7)
```

	precision	recall	f1-score	support
0	0.89	0.93	0.91	107
1	0.91	0.87	0.89	89
accuracy			0.90	196
macro avg	0.90	0.90	0.90	196
weighted avg	0.90	0.90	0.90	196

slightly lower result (0.90) than what we got from the averaging and max voting methods (0.91)

Boosting

- Bagging technique is parallel learning technique: fit all at once
- Boosting technique works in a sequential manner: fit one after the other
- A base learner tries to correct the error generated by the previous learner and this process continues until a superior meta learner is created

Boosting steps

1. A base learner is fit on a subset of the dataset.
2. Once the model is fit, predictions are made on the entire dataset.
3. The errors in the predictions are identified by comparing them with the actual labels.
4. Those examples that generated the wrong predictions are given larger weights.
5. Another base learner is fit on the dataset where the weights of the wrongly predicted examples in the previous step are altered.
6. This base learner tries to correct the errors of the earlier model and gives their predictions.
7. Steps 4, 5, and 6 are repeated until a strong meta learner is generated

Boosting - Example

```
# Defining the base learner
from sklearn.linear_model import LogisticRegression
bl1 = LogisticRegression(random_state=123)
```

```
# Define the boosting meta learner
from sklearn.ensemble import AdaBoostClassifier
boosting = AdaBoostClassifier(base_estimator=bl1, n_estimators=200)
```

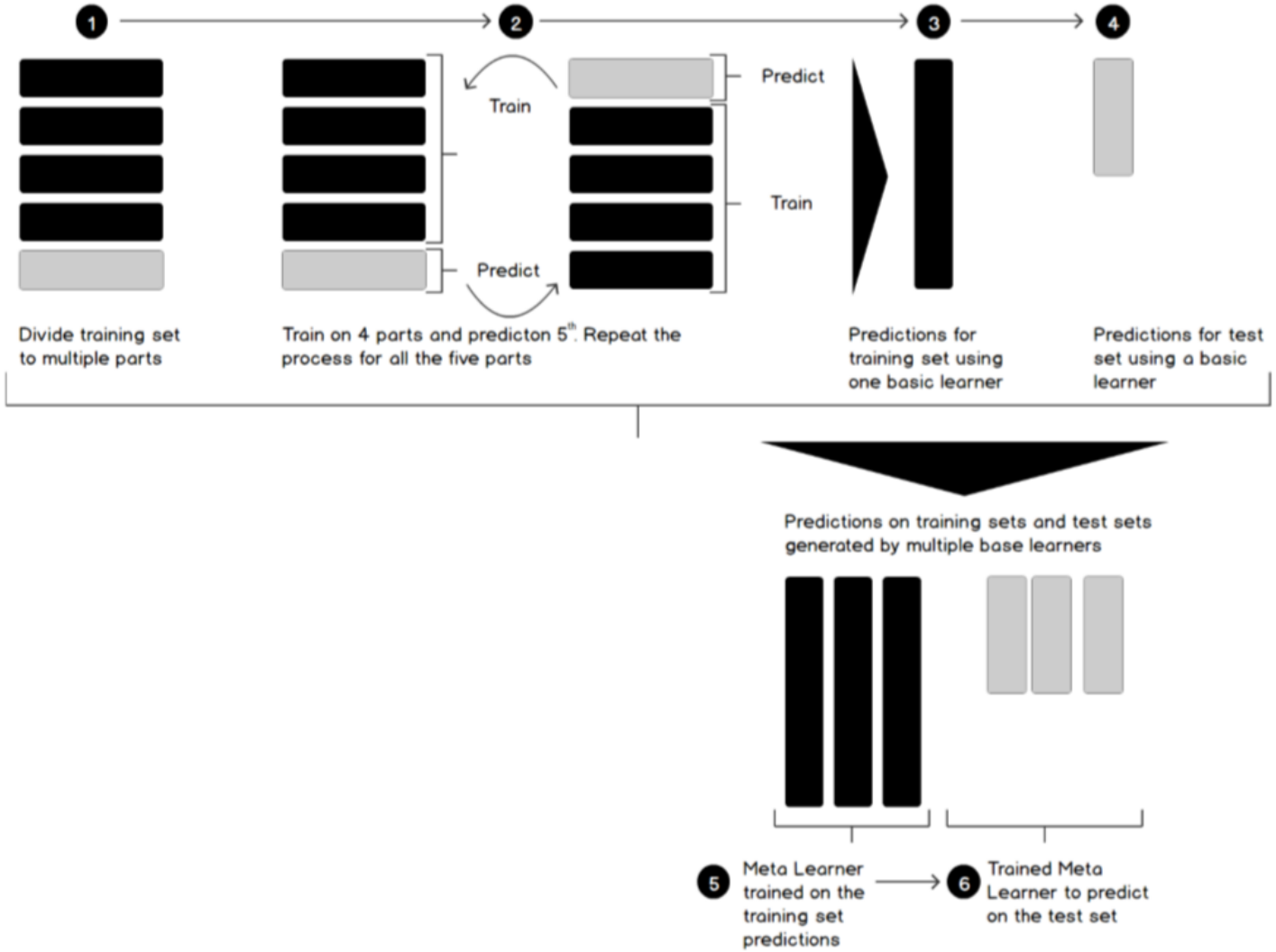
	precision	recall	f1-score	support
0	0.92	0.90	0.91	107
1	0.88	0.91	0.90	89
accuracy			0.90	196
macro avg	0.90	0.90	0.90	196
weighted avg	0.90	0.90	0.90	196

More balanced compared to the results that we got for the bagging method (the recall values of 0.93 and 0.87

Stacking

1. The training set is split into multiple parts, say, five parts.
2. A base learner (say, KNN) is fitted on four parts of the training set and then predicted on the fifth set. This process continues until the base learner predicts on each of the five parts of the training set. All the predictions, which are so generated, are collated to get the predictions for the complete training set.
3. The same base learner is then used to generate predictions on the test set as well.
4. Steps 2 and 3 are then repeated with a different base learner (e.g. random forest).
5. Next enters a new model, which acts as the meta learner (e.g. logistic regression).
6. The meta learner is fit on the predictions generated on the training set by the base learners.
7. Once the meta learner is fit on the training set, the same model is used to predict on the predictions generated on the test set by the base learners

Stacking


1
2
3
4
Predict
Train
Train
Predict
Divide training set to multiple parts
Train on 4 parts and predicton 5th. Repeat the process for all the five parts
Predictions for training set using one basic learner
Predictions for test set using a basic learner
Predictions on training sets and test sets generated by multiple base learners
5
Meta Learner trained on the training set predictions
6
Trained Meta Learner to predict on the test set

Stacking - Example

```
bl1 = KNeighborsClassifier(n_neighbors=5)
bl2 = RandomForestClassifier(random_state=123)
ml = LogisticRegression(random_state=123)
```

```
# Creating the stacking classifier
from mlxtend.classifier import StackingClassifier
stackclf = StackingClassifier(classifiers=[bl1, bl2],
                              meta_classifier=ml)
```

	precision	recall	f1-score	support
0	0.88	0.92	0.89	107
1	0.89	0.84	0.87	89
accuracy			0.88	196
macro avg	0.88	0.88	0.88	196
weighted avg	0.88	0.88	0.88	196

No improvements over the benchmark model using the stacking classifier
Re-choose base learners and meta learners

Summary

- Ensemble learning is a technique that aims to combine individual models to create a superior model , thereby reducing variance and bias and improving performance.
- Six different techniques of ensemble learning;
- Not all techniques will work in all scenarios.
- The rigor of experimentation with different techniques, use cases, and datasets makes a good data scientists!

Thank you