Ensemble Learning

How to Improve Results?

- Choice of Model
- Feature Engineering
- Missing Values
- Feature Scaling
- Hyper-parameter tuning

Improving Results Even More

Combine the power of multiple models

This is called **Ensemble** method.

To define Ensemble methods is a machine learning technique that combines several base models in order to produce one optimal predictive model.

Ensemble Methods - Bagging

- Use subsets of training data to train multiple models
    - samples are drawn with replacement
- Subsets may have have duplicate training sample
- All models perform prediction for Test Sample
- Voting scheme is used for final classification

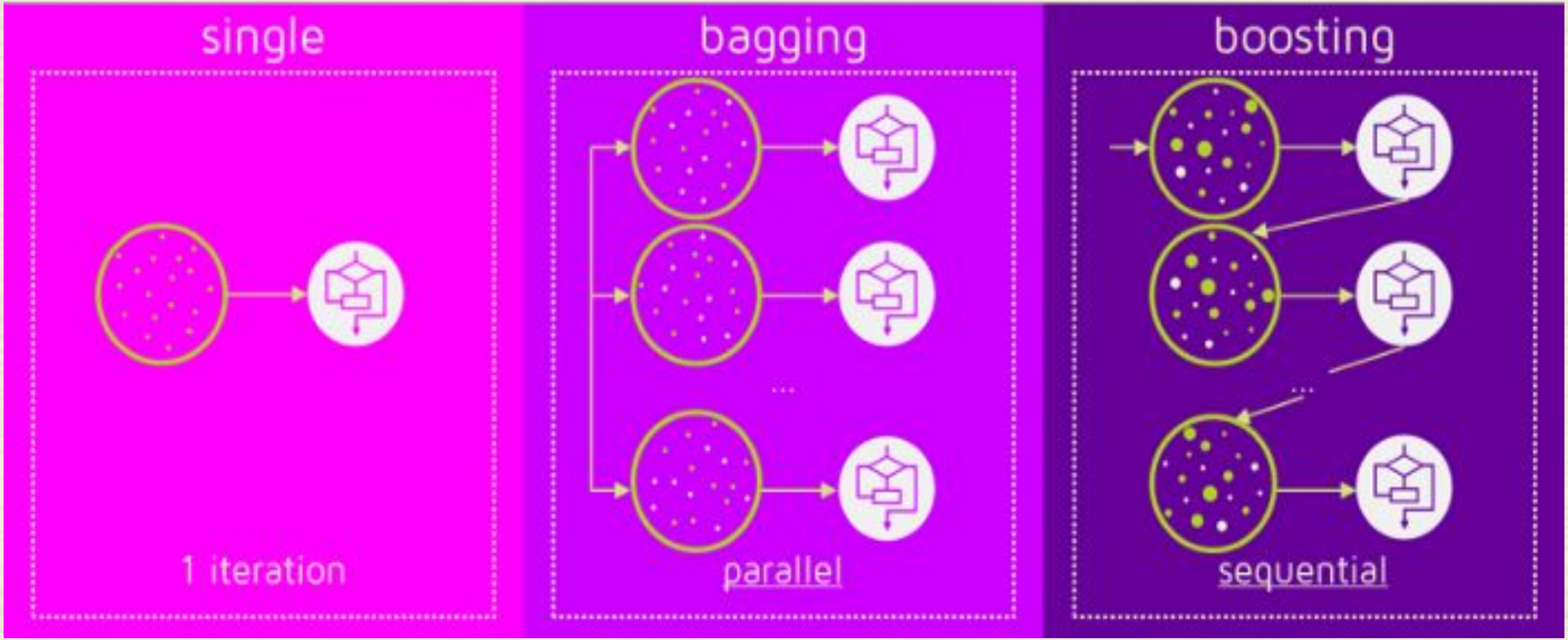
Ensemble Methods - Boosting

Boosting is also known as a scheme where weak learners are converted to strong learners.

- Assign weights to training samples
- Train a model using a sampling scheme where weights are taken into account for selection.
- Once a classifier is done with training, weights of “incorrectly classified” samples are updated
- Model is now re-trained so it can improve its classification strategy
- Process continues until desired performance is achieved or improvement stops.

Ensemble Methods - Random Forest Approach

- Multiple Models containing random set of features.
- Often paired with bagging approach



single
1 iteration
bagging
...
parallel
boosting
...
sequential

Stacking

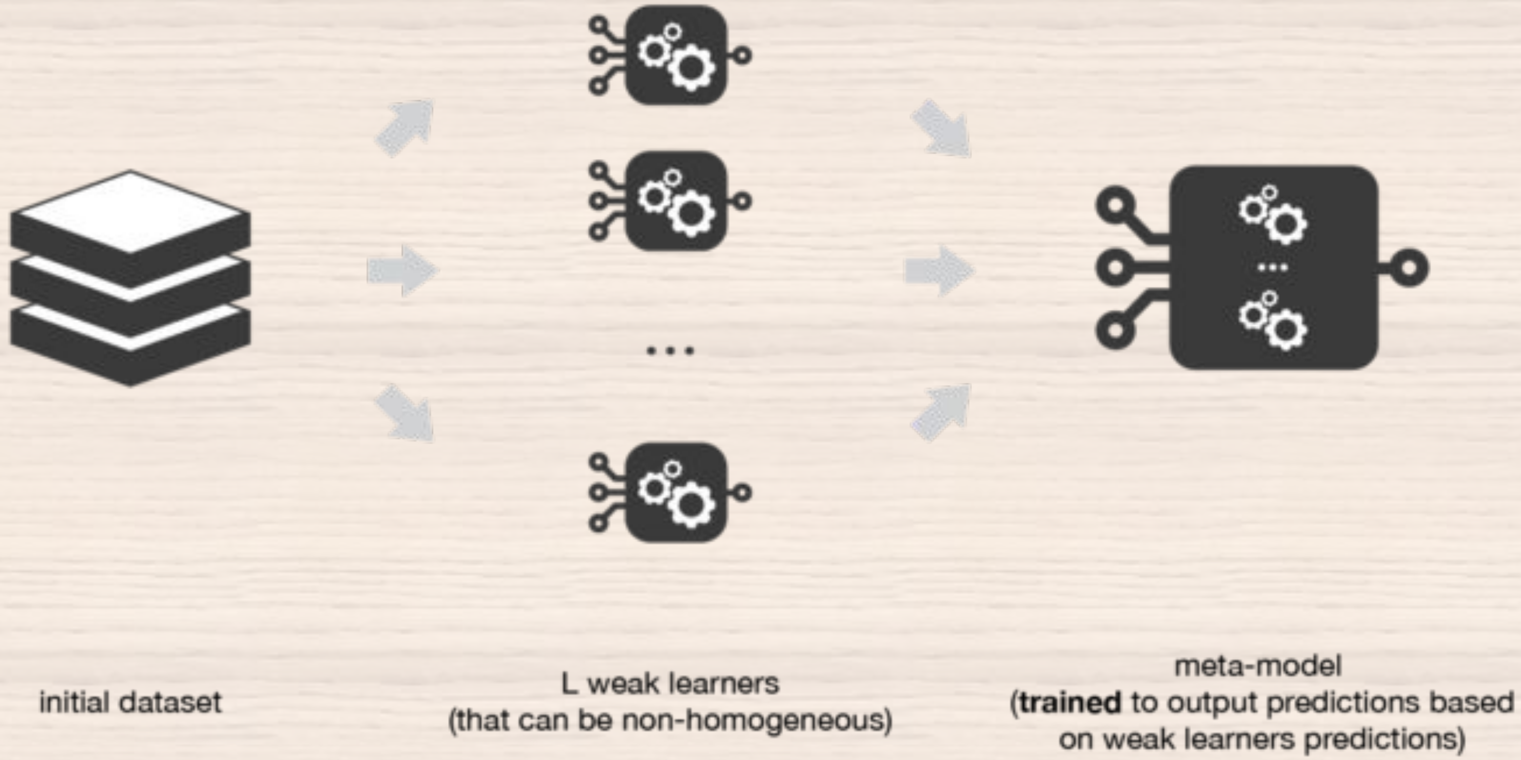
- Stacked Generalization or “Stacking” for short is an ensemble machine learning algorithm

- Stacking addresses the question that:
    - Given multiple machine learning models that are skillful on a problem, but in different ways, how do you choose which model to use ?

Stacking - Example

- The architecture of a stacking model involves two or more base models, and a meta-model that combines the predictions of the base models
  - **Base-Models:** Models fit on the training data and whose predictions are compiled.
  - **Meta-Model:** Model that learns how to best combine the predictions of the base models.
- For example, for a classification problem, we can choose as
  - Base Models: a KNN classifier, a logistic regression and a SVM,
  - Meta Model: a neural network.
    - the neural network will take as inputs the outputs of our three weak learners and will learn to return final predictions based on it.

Stacking


initial dataset
L weak learners
(that can be non-homogeneous)
meta-model
(**trained** to output predictions based
on weak learners predictions)

How is Stacking Different?

- Unlike bagging, in stacking, the classifiers are typically different

- In stacking classifiers are trained on the same dataset (e.g. instead of samples of the training dataset).

- Unlike boosting, in stacking, a single model is used to learn how to best combine the predictions from the contributing models (e.g. instead of a sequence of models that correct the predictions of prior models).

Stacking - continued

- The outputs from the base models used as input to the meta-model may be real value in the case of regression, and probability values, probability like values, or class labels in the case of classification.

- The meta-model is trained on the predictions made by base models on out-of-sample data.
  - data not used to train the base models is fed to the base models,
  - predictions are made, and
  - these predictions, along with the expected outputs, provide the input and output pairs of the training dataset used to fit the meta-model.

Stacking – SciKit-Learn

Stacking is provided via the StackingRegressor and StackingClassifier modules in sk-learn

```
from sklearn.datasets import load_iris
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import LinearSVC
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.ensemble import StackingClassifier
X, y = load_iris(return_X_y=True)
estimators = [
    ('rf', RandomForestClassifier(n_estimators=10, random_state=42)),
    ('svr', make_pipeline(StandardScaler(),
                          LinearSVC(random_state=42)))
]
clf = StackingClassifier(
    estimators=estimators, final_estimator=LogisticRegression()
)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, stratify=y, random_state=42
)
clf.fit(X_train, y_train).score(X_test, y_test)
```

References & Learning Resources

https://towardsdatascience.com/ensemble-methods-in-machine-learning-what-are-they-and-why-use-them-68ec3f9fef5f

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html

https://machinelearningmastery.com/stacking-ensemble-machine-learning-with-python/

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.StackingRegressor.html

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.StackingClassifier.html

Additional Slides

Exporting Trained Models for Use

```
import pickle
# Fit the model on training set
model = LogisticRegression()
model.fit(X_train, Y_train)
# save the model to disk
filename = 'finalized_model.sav'
pickle.dump(model, open(filename, 'wb'))

# some time later...

# load the model from disk
loaded_model = pickle.load(open(filename, 'rb'))
result = loaded_model.score(X_test, Y_test)
print(result)
```

Exporting Trained Models for Use

```
import joblib
# Fit the model on training set
model = LogisticRegression()
model.fit(X_train, Y_train)
# save the model to disk
filename = 'finalized_model.sav'
joblib.dump(model, filename)

# some time later...

# load the model from disk
loaded_model = joblib.load(filename)
result = loaded_model.score(X_test, Y_test)
print(result)
```