The intuition behind stacking

ENSEMBLE METHODS IN PYTHON

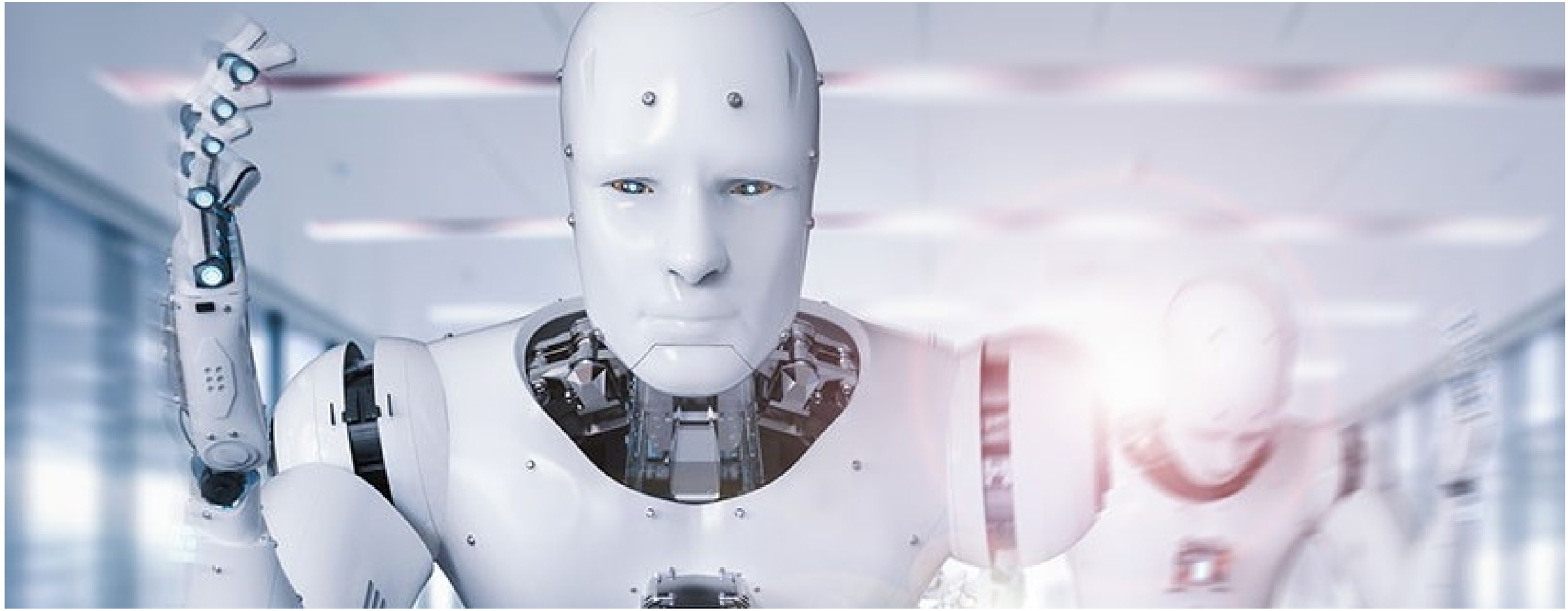
Román de las Heras
Data Scientist, SAP / Agile Solutions

Relay races

Effective team leader (anchor):

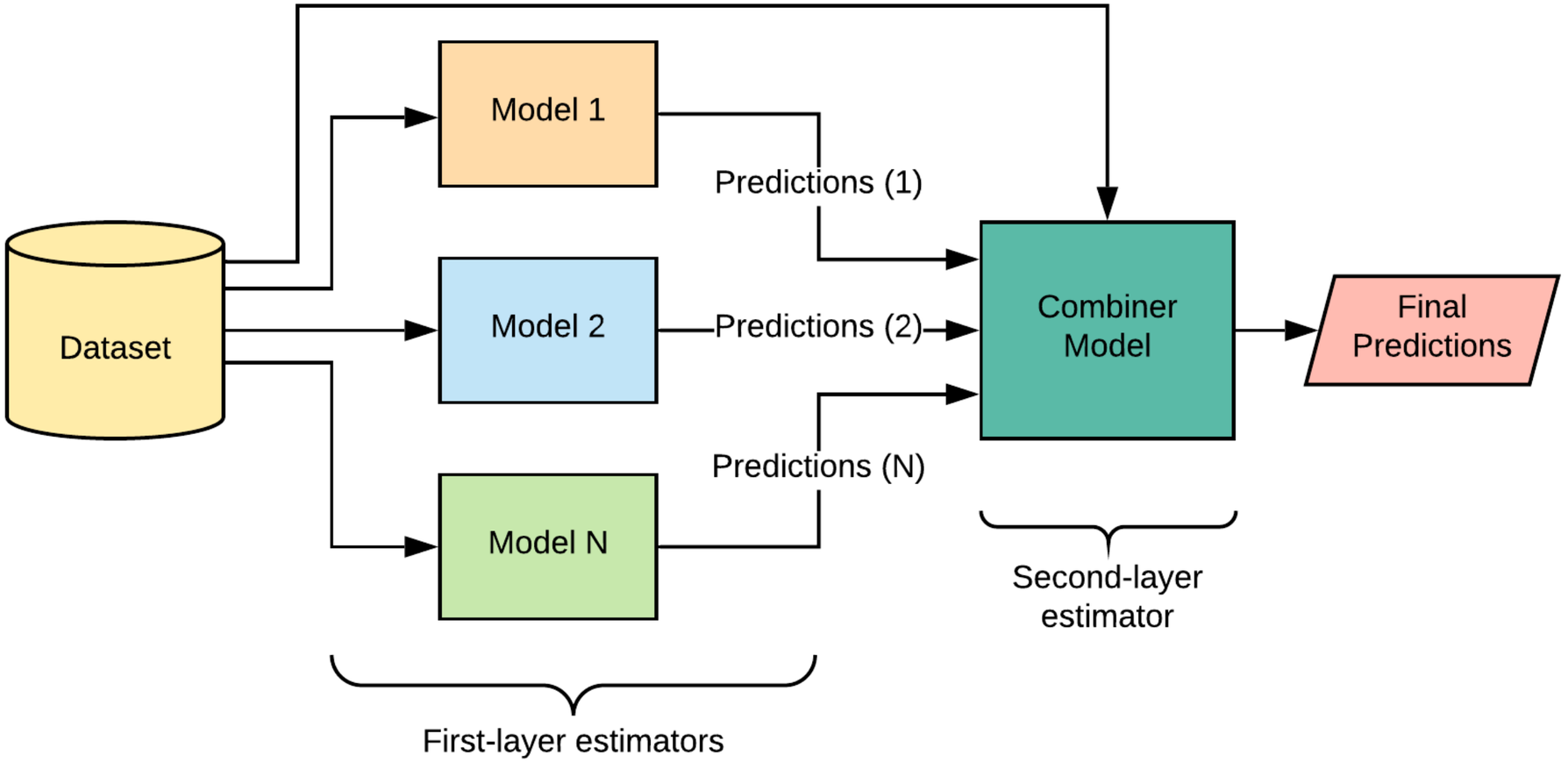
- *Know the team*: strengths and weaknesses
- *Define tasks*: responsibilities
- *Take part*: participation

Relay race for models

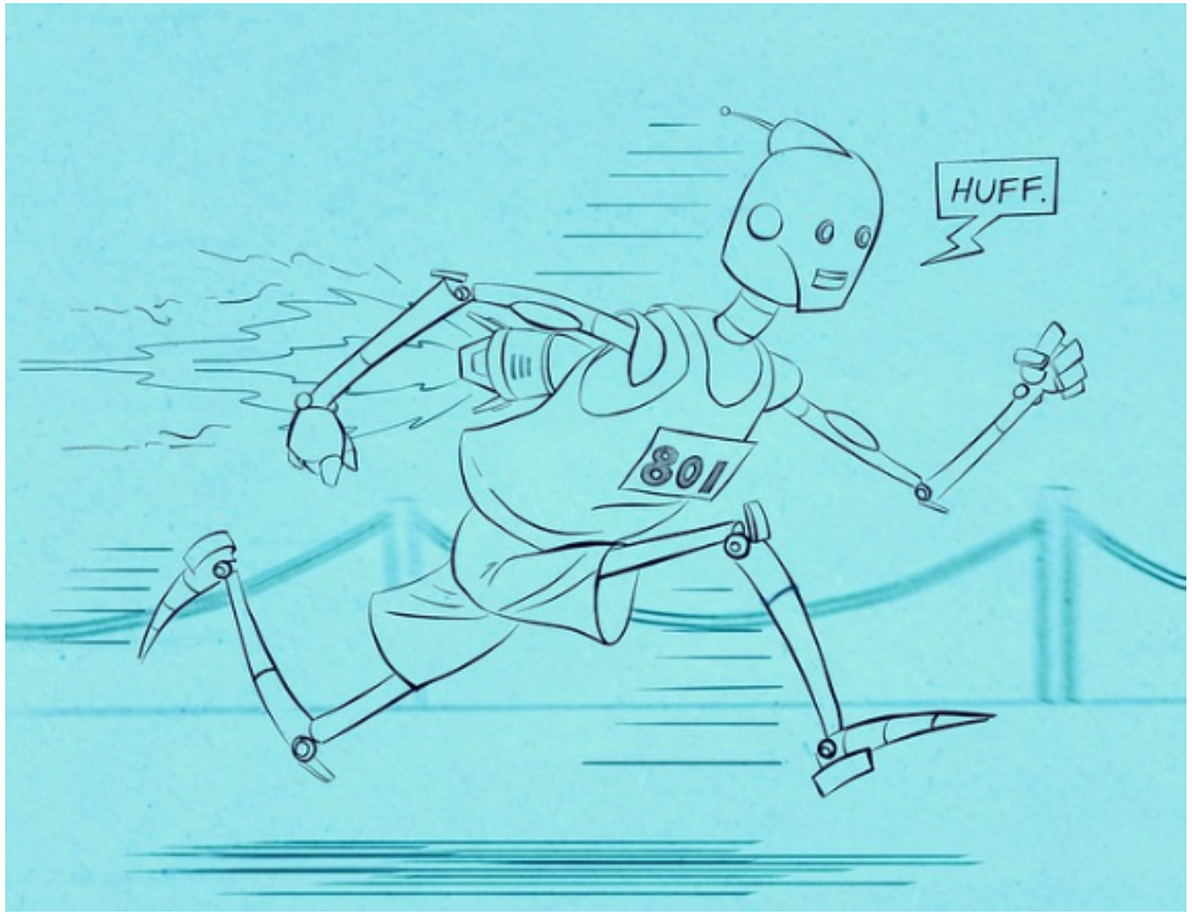


Passing the baton <--> Passing predictions

Stacking architecture



Combiner model as anchor



Effective combiner model (anchor):

- *Know the team*: strengths and weaknesses
- *Define tasks*: responsibilities
- *Take part*: participation

Time to practice!

ENSEMBLE METHODS IN PYTHON

Build your first stacked ensemble

ENSEMBLE METHODS IN PYTHON

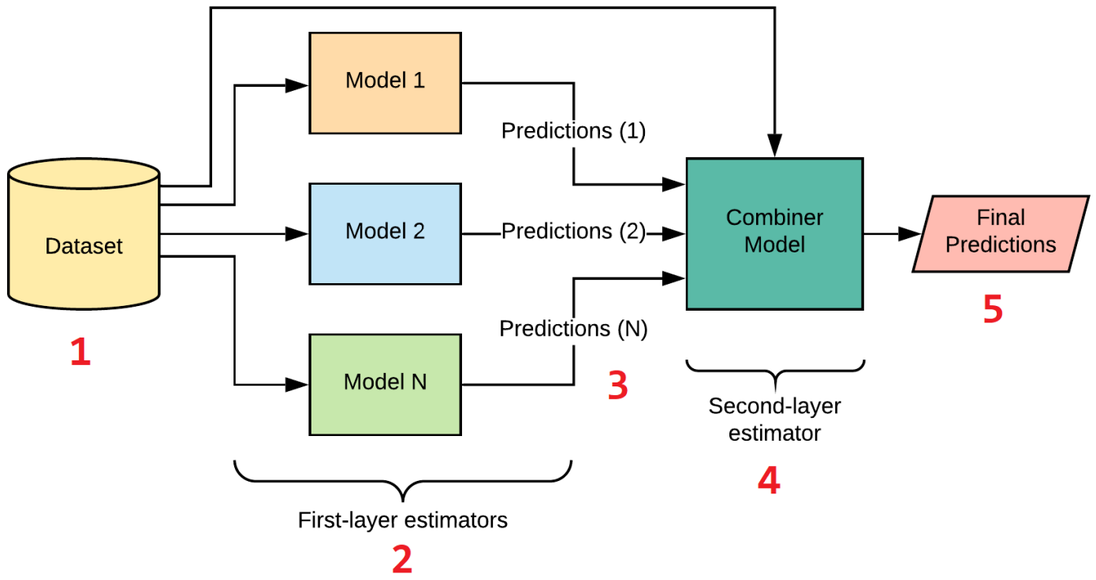
Román de las Heras
Data Scientist, SAP / Agile Solutions

Stacking models with scikit-learn

Some reasons to build from scratch:

1. `scikit-learn` has no stacking implementation
2. We will build stacking models from scratch
3. `scikit-learn` estimators can be used as a base

General Steps



General steps for the implementation:

1. Prepare the dataset
2. Build the first-layer estimators
3. Append the predictions to the dataset
4. Build the second-layer meta estimator
5. Use the stacked ensemble for predictions

1. Prepare the dataset

```
# Select input features and target
selected_feats = ['feat1', 'feat2', ...,
                  'featN']
features = dataset.iloc[:,selected_feats]
target = dataset['target_feature']
```

```
# Data cleaning
# Example: Fill NA values with zero
features.fillna(value=0, inplace=True)
```

```
# Apply any required transformations
# Example: categorical to 'dummies'
features = pd.get_dummies(features)
```

```
# Split into train (60%) and test(40%)
X_train, X_test, y_train, y_test =
train_test_split(
    features, target, test_size=0.4,
    random_state=42)
```

2. Build the first-layer estimators

Build and fit the first-layer estimators

```
# 1. A Gaussian Naive Bayes classifier
clf_nb = GaussianNB()
clf_nb.fit(X_train, y_train)
```

```
# 2. A 5-nearest neighbors classifier using the 'Ball-Tree' algorithm
clf_knn = KNeighborsClassifier(n_neighbors=5, algorithm='ball_tree')
clf_knn.fit(X_train, y_train)
```

3. Append the predictions to the dataset

Calculate predictions and add them to the training set:

```
# Predict with the first-layer estimators on X_train
pred_nb = clf_nb.predict(X_train)
pred_knn = clf_knn.predict(X_train)
```

```
# Create a Pandas DataFrame with the predictions
pred_df = pd.DataFrame({
    'pred_nb': pred_nb,
    'pred_knn': pred_knn
})
```

```
# Concatenate X_train with the predictions DataFrame
X_train_2nd = pd.concat([X_train, pred_df], axis=1)
```

4. Build the second-layer meta estimator

```
# Instantiate the second-layer estimator
# Example: a Logistic Regression classifier
clf_stack = LogisticRegression()
```

```
# Train the model using the second training set
clf_stack.fit(X_train_2nd, y_train)
```

5. Use the stacked ensemble for predictions

```
# Predict with the first-layer estimators on X_train
pred_nb = clf_nb.predict(X_test)
pred_knn = clf_knn.predict(X_test)
pred_df = pd.DataFrame({
    'pred_nb': pred_nb,
    'pred_knn': pred_knn
})
```

```
# Concatenate X_test with the predictions DataFrame
X_test_2nd = pd.concat([X_test, pred_df], axis=1)
```

```
# Obtain the final predictions from the second-layer estimator
pred_stack = clf_stack.predict(X_test_2nd)
```

It's your turn!

ENSEMBLE METHODS IN PYTHON

Let's mlxtend it!

ENSEMBLE METHODS IN PYTHON

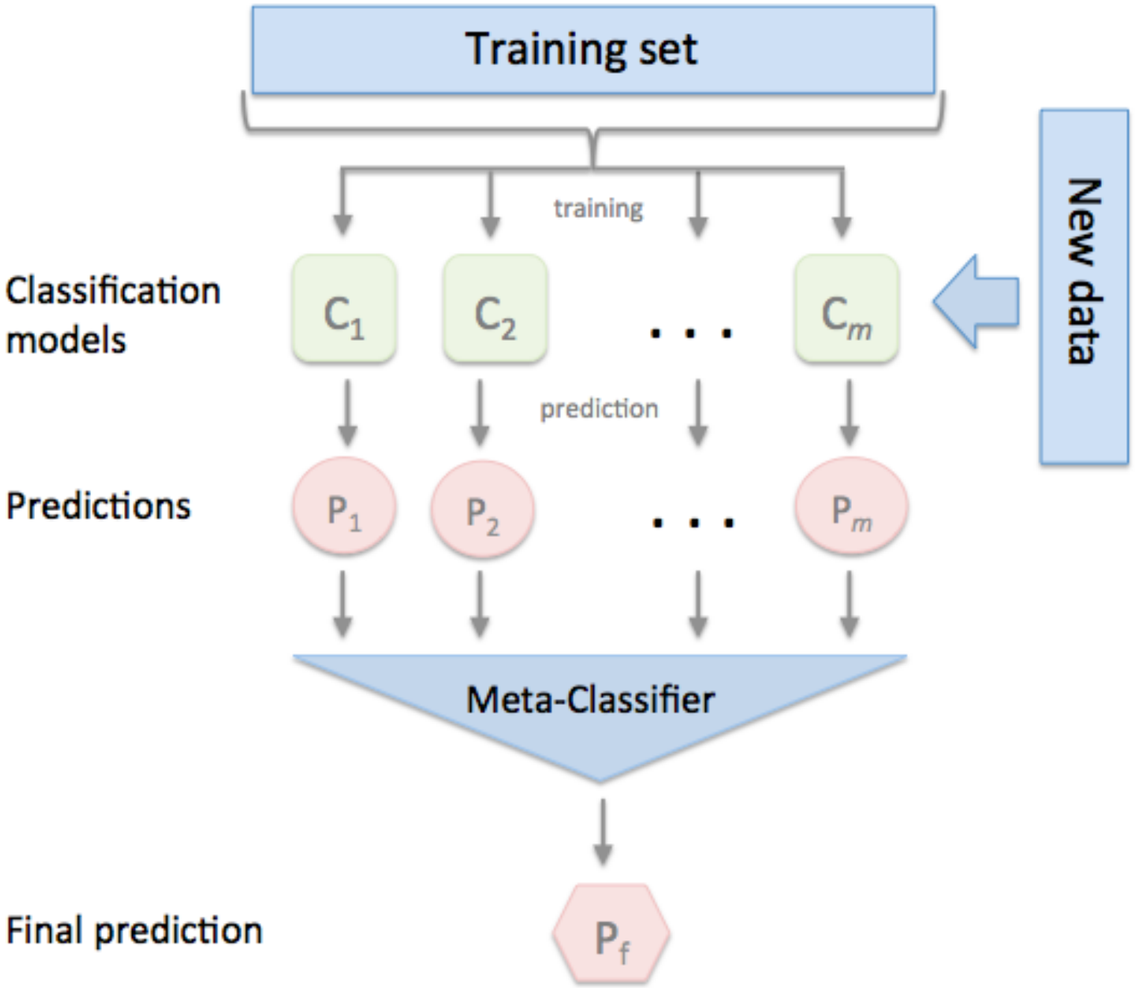
Román de las Heras
Data Scientist, SAP / Agile Solutions

Mlxtend

- Machine **L**earning E**xten**sions
- Utilities and tools for Data Science tasks:
  - Feature selection
  - Ensemble methods
  - Visualization
  - Model evaluation
- Intuitive and friendly API
- Compatible with `scikit-learn` estimators

[1] Raschka, Sebastian (2018) MLxtend: Providing machine learning and data science utilities and extensions to Python's scientific computing stack: http://rasbt.github.io/mlxtend/

Stacking implementation from mlxtend



Characteristics:

- Individual estimators are trained on the complete features
- The meta-estimator is trained **using the predictions as the only meta-features**
- The meta-estimator can be trained with labels or probabilities as target

StackingClassifier with mlxtend

```
from mlxtend.classifier import StackingCla
```

```
# Instantiate the 1st-layer classifiers
clf1 = Classifier1(params1)
clf2 = Classifier2(params2)
...
clfN = ClassifierN(paramsN)
```

```
# Instantiate the 2nd-layer classifier
clf_meta = ClassifierMeta(paramsMeta)
```

```
# Build the Stacking classifier
clf_stack = StackingClassifier(
    classifiers=[clf1, clf2, ... clfN],
    meta_classifier=clf_meta,
    use_probas=False,
    use_features_in_secondary=False)
```

```
# Use the fit and predict methods
# like with scikit-learn estimators
clf_stack.fit(X_train, y_train)
pred = clf_stack.predict(X_test)
```

StackingRegressor with mlxtend

```
from mlxtend.regressor import StackingRegr
```

```
# Instantiate the 1st-layer regressors
reg1 = Regressor1(params1)
reg2 = Regressor2(params2)
...
regN = RegressorN(paramsN)
```

```
# Instantiate the 2nd-layer regressor
reg_meta = RegressorMeta(paramsMeta)
```

```
# Build the Stacking regressor
reg_stack = StackingRegressor(
    regressors=[reg1, reg2, ... regN],
    meta_regressor=reg_meta,
    use_features_in_secondary=False)
```

```
# Use the fit and predict methods
# like with scikit-learn estimators
reg_stack.fit(X_train, y_train)
pred = reg_stack.predict(X_test)
```

Let's mlxtend it!

ENSEMBLE METHODS IN PYTHON

Ensembling it all together

ENSEMBLE METHODS IN PYTHON

Román de las Heras
Data Scientist, SAP / Agile Solutions

Chapter 1: Voting and Averaging

Voting

- **Combination:** mode (majority)
- Classification
- Heterogeneous ensemble method

Averaging

- **Combination:** mean (average)
- Classification and Regression
- Heterogeneous ensemble method

Good choices when you:

- Have built multiple different models
- Are not sure which is the best
- Want to improve the overall performance

Chapter 2: Bagging

Weak estimator

- Performs just better than random guessing
- Light model and fast model
- Base for homogeneous ensemble methods

Bagging (Bootstrap Aggregating)

- Random subsamples with replacement
- Large amount of "weak" estimators
- Aggregated by Voting or Averaging
- Homogeneous ensemble method

Good choice when you:

- Want to reduce variance
- Need to avoid overfitting
- Need more stability and robustness

** Observation:*

- Bagging is computationally expensive

Chapter 3: Boosting

Gradual learning

- Heterogeneous ensemble method type
- Based on iterative learning
- Sequential model building

Boosting algorithms

- AdaBoost
- Gradient Boosting:
  - XGBoost
  - LightGBM
  - CatBoost

Good choice when you:

- Have complex problems
- Need to apply parallel processing or distributed computing
- Have big datasets or high-dimensional categorical features

Chapter 4: Stacking

Stacking

- **Combination:** meta-estimator (model)
- Classification and Regression
- Heterogeneous ensemble method

Implementation

- From scratch using pandas and sklearn
- Using the existing MLxtend library

Good choice when you:

- Have tried Voting / Averaging but results are not as expected
- Have built models which perform well in different cases

Thank you and well ensembled!

ENSEMBLE METHODS IN PYTHON