The intuition behind stacking

ENSEMBLE METHODS IN PYTHON

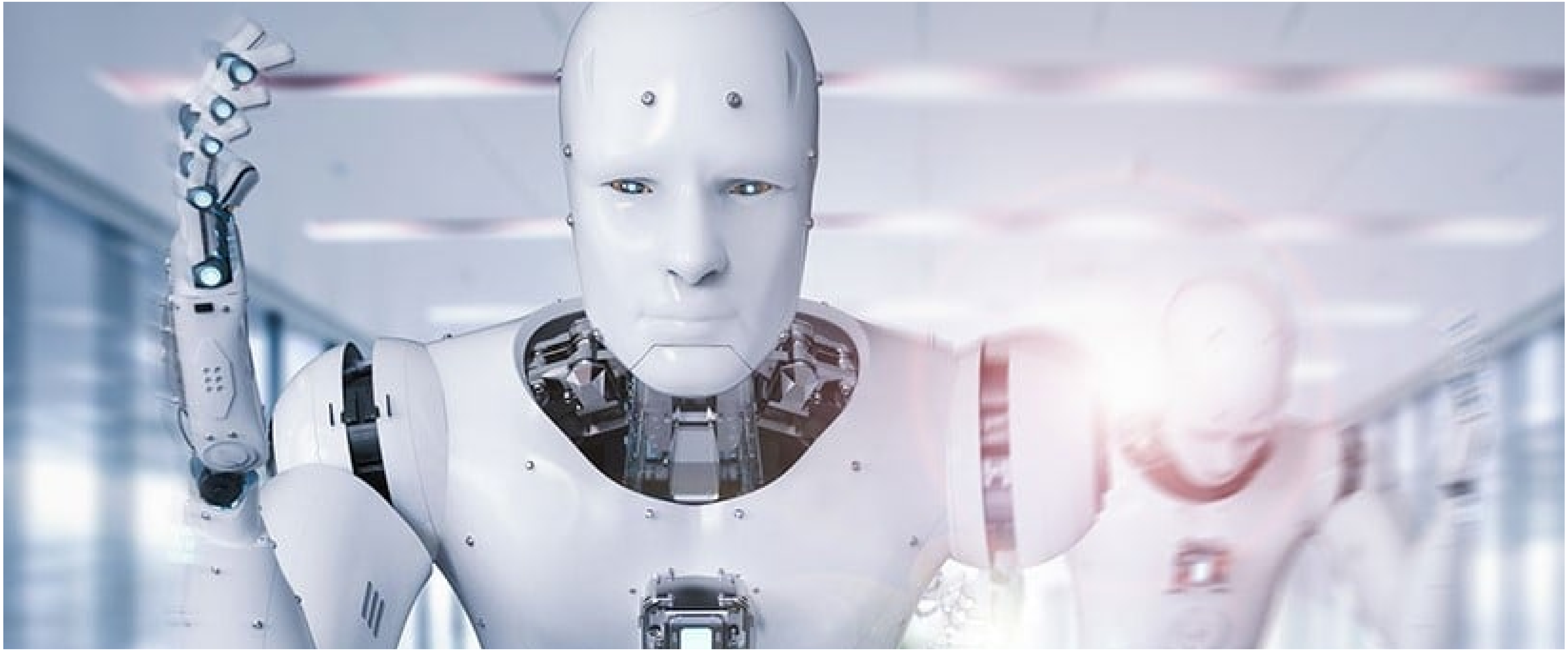
Román de las Heras
Data Scientist, Appodeal

Relay races

Effective team leader (anchor):

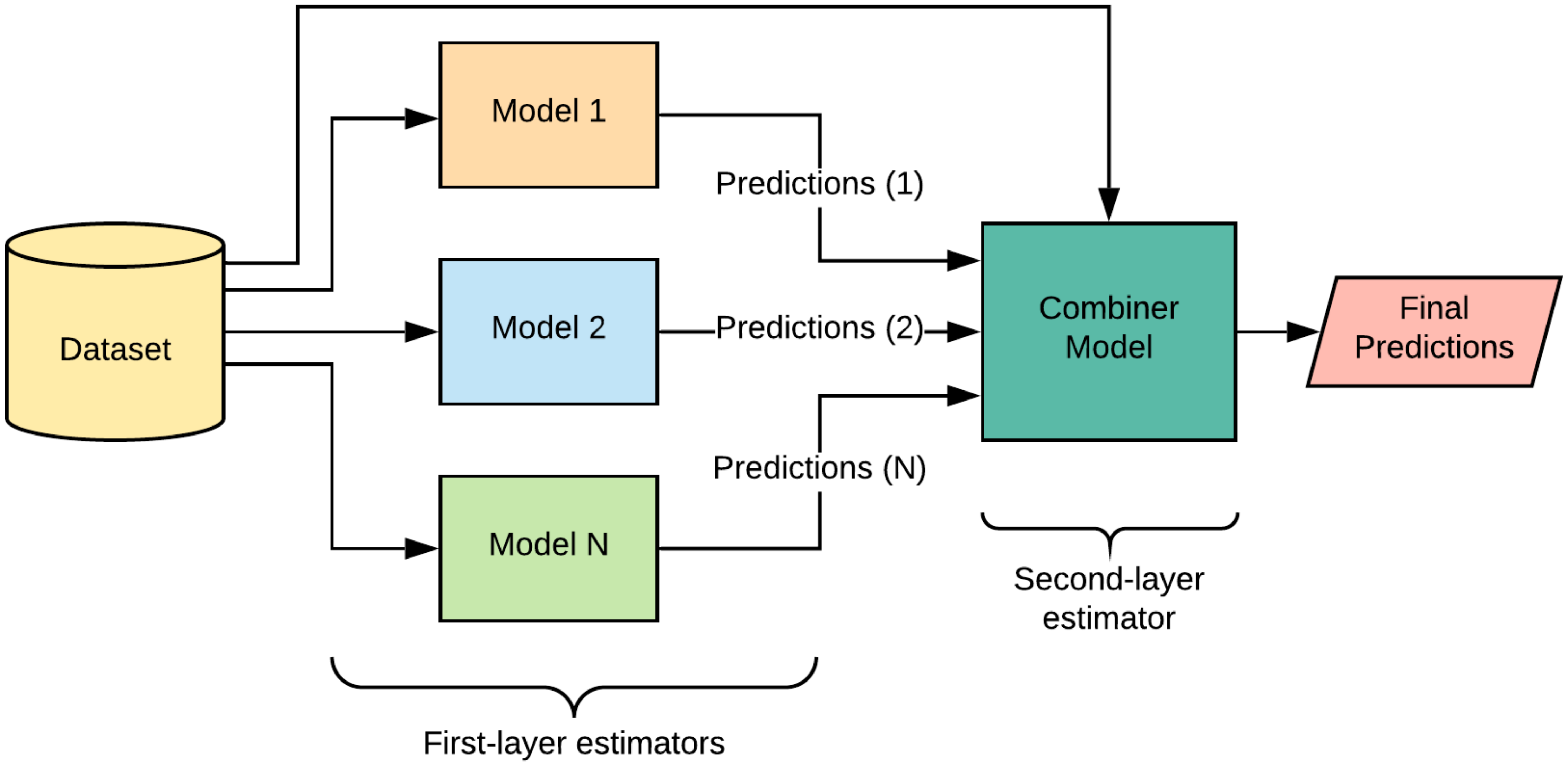
- *Know the team*: strengths and weaknesses
- *Define tasks*: responsibilities
- *Take part*: participation

Relay race for models

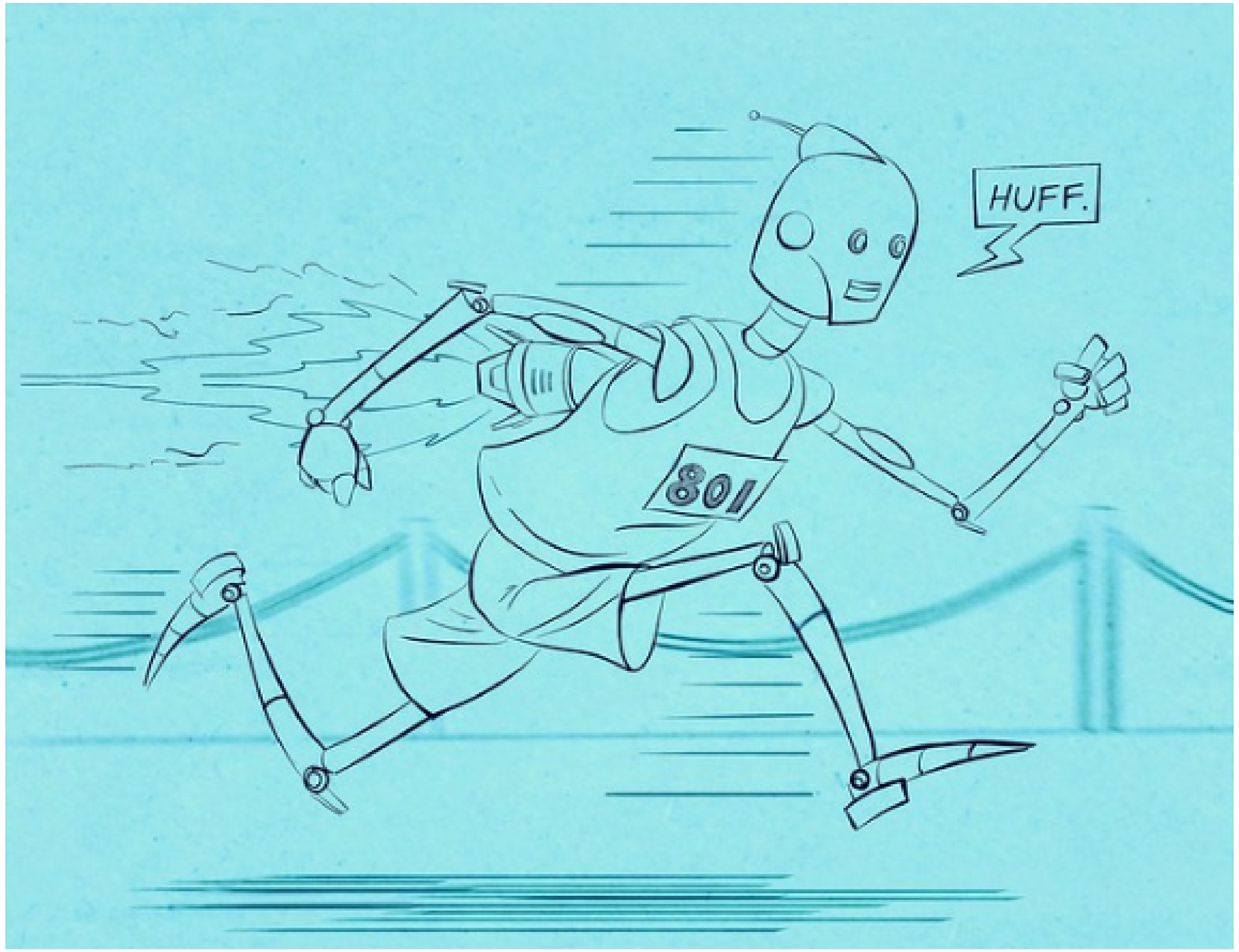


Passing the baton <--> Passing predictions

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

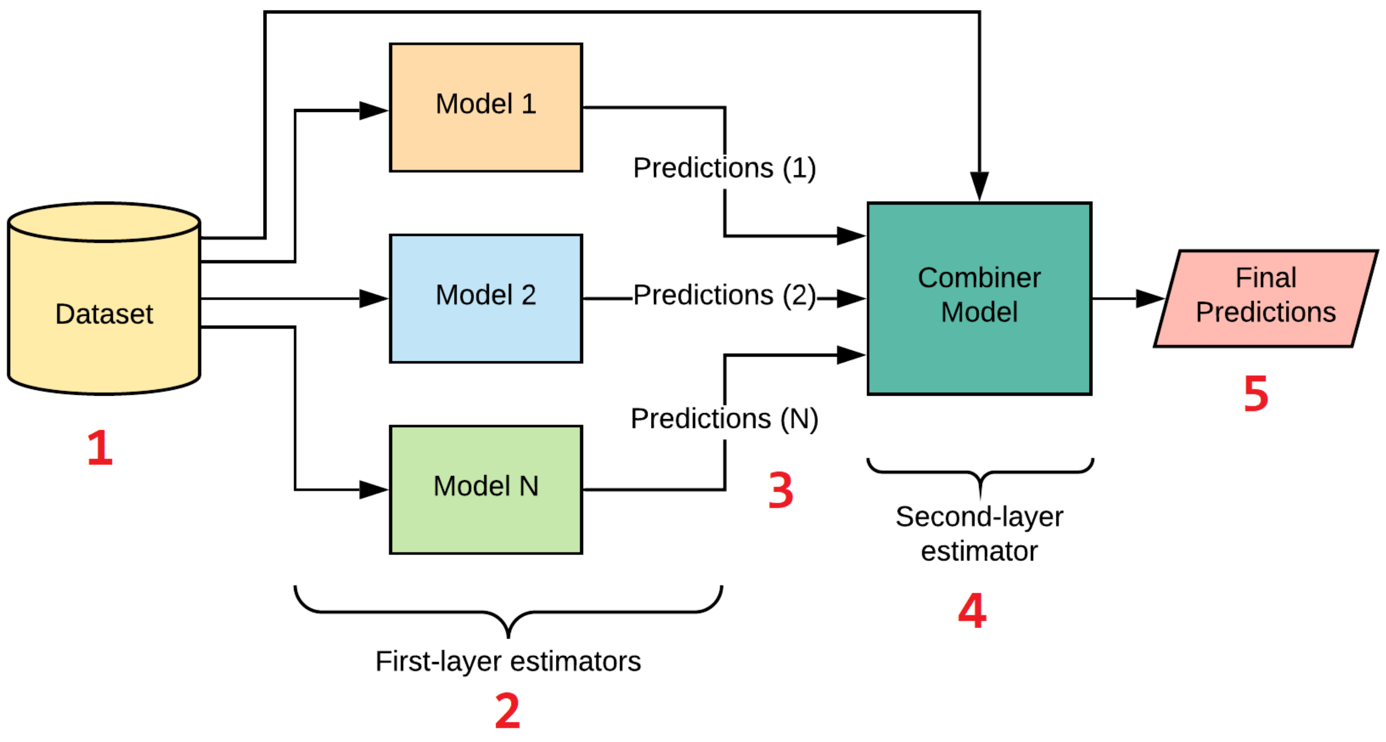
Román de las Heras
Data Scientist, Appodeal

Stacking models with scikit-learn

Some features of stacking implementation from scikit-learn:

1. `scikit-learn` provides stacking estimators (since version 0.22)
2. Compatible with other `scikit-learn` estimators
3. The final estimator is trained through cross-validation

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

Stacking classifier

```python
from sklearn.ensemble import StackingClassifier
```

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

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

```python
# Build the Stacking classifier
clf_stack = StackingClassifier(
   estimators=classifiers,
   final_estimator=clf_meta,
   cv=5,
   stack_method='predict_proba',
   passthrough=False)
```

```python
# Use the fit and predict methods
clf_stack.fit(X_train, y_train)
pred = clf_stack.predict(X_test)
```

Stacking regressor

```python
from sklearn.ensemble import StackingRegressor
```

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

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

```python
# Build the Stacking regressor
reg_stack = StackingRegressor(
   estimators=regressors,
   final_estimator=reg_meta,
   cv=5,
   passthrough=False)
```

```python
# Use the fit and predict methods
reg_stack.fit(X_train, y_train)
pred = reg_stack.predict(X_test)
```

It's your turn!

ENSEMBLE METHODS IN PYTHON

Let's mlxtend it!

ENSEMBLE METHODS IN PYTHON

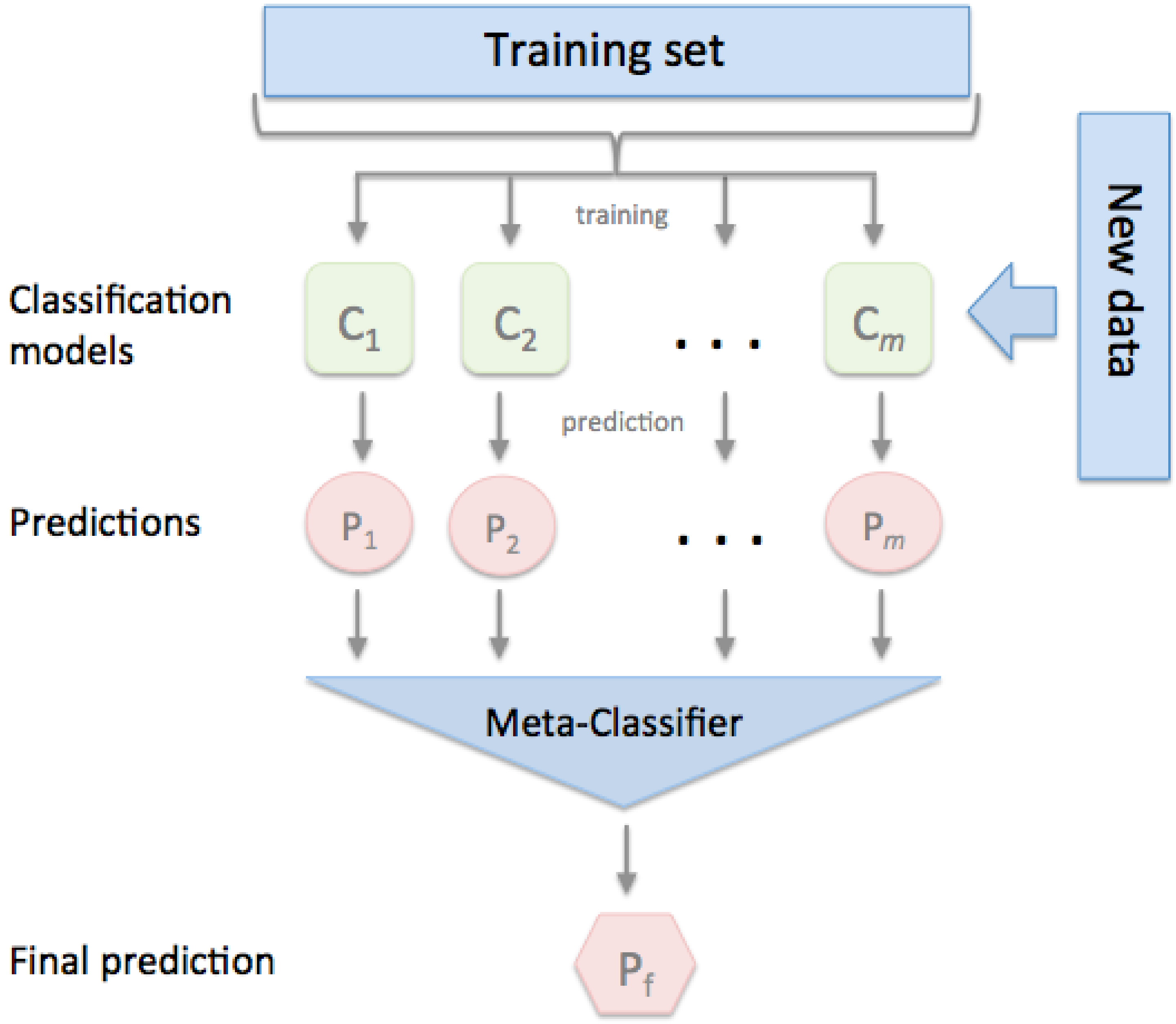
Román de las Heras
Data Scientist, Appodeal

Mlxtend

- **M**achine **L**earning **Ext**ensions
- 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: https://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 StackingClassifier
```

```
# 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

```python
from mlxtend.regressor
    import StackingRegressor
```

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

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

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

```python
# 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, Appodeal

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

- Homogeneous 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