# Aprendizado de Máquina e Reconhecimento de Padrões 2021.2

## Scikit-Learn Design Principles

Prof. Samuel Martins (Samuka)
*samuel.martins@ifsp.edu.br*



INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
São Paulo
Campus Campinas

# Preliminary Concepts

## Model Hyperparameters

Properties that are external to the model and whose value cannot be estimated from data.

Examples:

- Imputer's strategy: 'median'
- Number of neighbors for KNN: 3

## Model Parameters

Properties that are internal to the model and whose value can be estimated from data.

Examples:

- Estimated value for missing values: 20 (median)
- Estimated coefficients of a linear regression.

**Problem**

Filling in missing values (imputer)

| | Name | Age |
|---|---|---|
| 0 | John | 19.0 |
| 1 | Maria | 22.0 |
| 2 | Alice | 18.0 |
| 3 | Margot | 20.0 |
| 4 | Pedro | NaN |
| 5 | Giovanni | 21.0 |

(training) data

**(hyperparameter)**

Imputer's strategy: 'median'

**(parameter)**

estimated value: 20

# Main Interfaces (API) of Scikit-Learn

## Estimators

- Main and core interface of Scikit-learn;
- Any object that can **estimate** some **parameters** based on a dataset;
- Estimation (learning/training) is performed by the `fit()` method;
- **Hyperparameters** needed to guide the estimation must be set as an a class attribute in the constructor.

```
# unsupervised learning
estimator.fit(X_train)


# supervised learning
estimator.fit(X_train, y_train)
```

# Main Interfaces (API) of Scikit-Learn

## Estimators

- Main and core interface of Scikit-learn;
- Any object that can **estimate** some **parameters** based on a dataset;
- Estimation (learning/training) is performed by the fit() method;
- **Hyperparameters** needed to guide the estimation must be set as an a class attribute in the constructor.

```
# unsupervised learning
estimator.fit(X_train)


# supervised learning
estimator.fit(X_train, y_train)
```

## Transformers

- **Estimators** which can also **transform** data with transform() or fit_transform();
- Preprocessing, feature selection, feature extraction and dimensionality reduction algorithms are all provided as **transformers** within the library;

```
new_data = transformer.transform(data)
new_data = transformer.fit_transform(data)
```

# Main Interfaces (API) of Scikit-Learn

## Estimators

- Main and core interface of Scikit-learn;
- Any object that can **estimate** some **parameters** based on a dataset;
- Estimation (learning/training) is performed by the fit() method;
- **Hyperparameters** needed to guide the estimation must be set as an a class attribute in the constructor.

```
# unsupervised learning
estimator.fit(X_train)


# supervised learning
estimator.fit(X_train, y_train)
```

## Transformers

- **Estimators** which can also **transform** data with transform() or fit_transform();
- Preprocessing, feature selection, feature extraction and dimensionality reduction algorithms are all provided as **transformers** within the library;

```
new_data = transformer.transform(data)
new_data = transformer.fit_transform(data)
```

## Predictors

- Estimators that can also **predict** a value.
- From a trained model, predictors return **predicted labels (numeric or categorical)** for a given **input features** by using the *predict()* method.
- Some predictors may also provide **probabilities** and prediction **scores** by using, respectively, *predict_proba()* and *score()*.

```
prediction = predictor.predict(data)
probability = predictor.predict_proba(data)
score = model.score(data)
```

# Main Interfaces (API) of Scikit-Learn

## Estimators

- Main and core interface of Scikit-learn;
- Any object that can **estimate** some **parameters** based on a dataset;
- Estimation (learning/training) is performed by the fit() method;
- **Hyperparameters** needed to guide the estimation must be set as an a class attribute in the constructor.

```
# unsupervised learning
estimator.fit(X_train)


# supervised learning
estimator.fit(X_train, y_train)
```

## Transformers

- **Estimators** which can also **transform** data with transform() or fit_transform();
- Preprocessing, feature selection, feature extraction and dimensionality reduction algorithms are all provided as **transformers** within the library;

```
new_data = transformer.transform(data)
new_data = transformer.fit_transform(data)
```

## Predictors

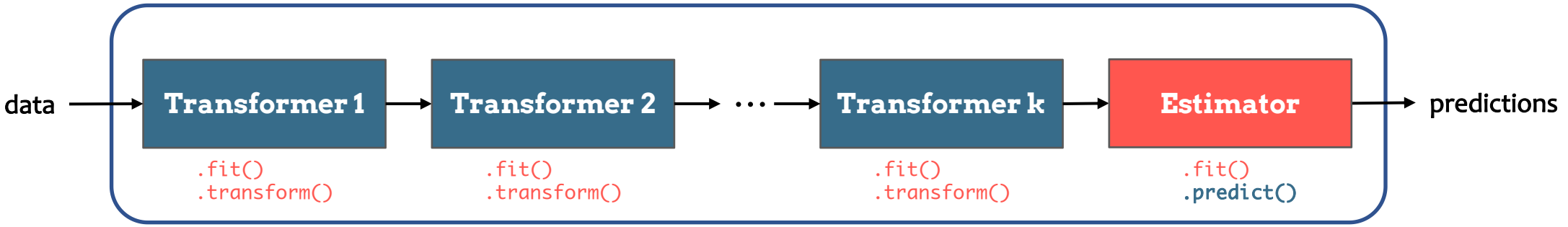
- Estimators that can also **predict** a value.
- From a trained model, predictors return **predicted labels (numeric or categorical)** for a given **input features** by using the *predict()* method.
- Some predictors may also provide **probabilities** and prediction **scores** by using, respectively, *predict_proba()* and *score()*.

```
prediction = predictor.predict(data)
probability = predictor.predict_proba(data)
score = model.score(data)
```

**Predictors** are commonly referred to as **estimators**.

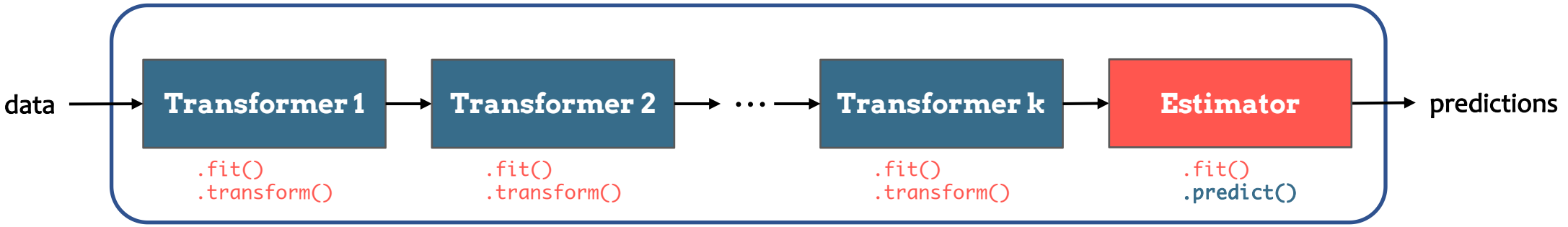
# Scikit-Learn Pipelines

- A scikit-learn Pipeline is an object that sequently applies **a list of transforms** and a final estimator (predictor);



# Scikit-Learn Pipelines

- A scikit-learn Pipeline is an object that sequently applies **a list of transforms** and a final estimator (predictor);




- The Pipeline is build using a list of **(key, value)** pairs, where:
  - the **key** a string containing the **name** you want to give this step, and the **value** is an **estimator object;**

```
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression

from sklearn.pipeline import Pipeline
```

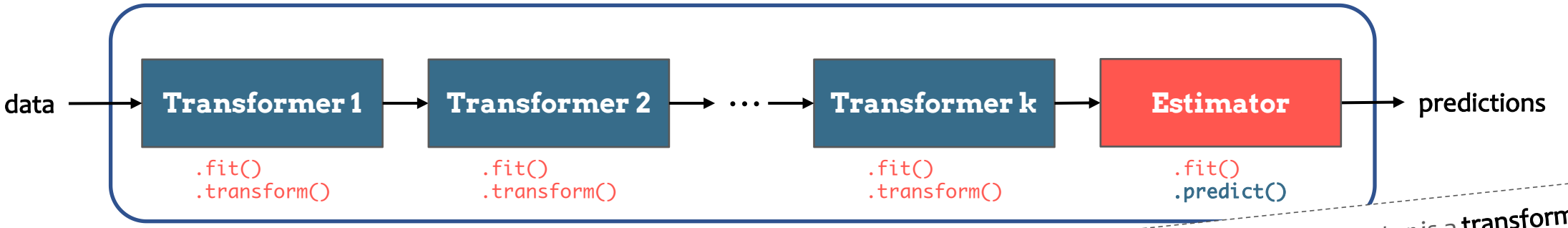
```
pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy="median")),
    ('scaler', StandardScaler()),
    ('linear_regression', LinearRegression())
])
```

```
pipeline.fit(X_train, y_train)
```

```
y_test_pred = pipeline.predict(X_test)
```

# Scikit-Learn Pipelines

- A scikit-learn Pipeline is an object that sequently applies **a list of transforms** and a final estimator (predictor);




- The Pipeline is build using a list of **(key, value)** pairs, where:
  - the **key** a string containing the **name** you want to give this step, and the **value** is an **estimator ob**

If the last estimator is a **transformer**, its `fit_transform()` method will be executed so that the **output** will be the **final transformed data**.

```
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression

from sklearn.pipeline import Pipeline

pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy="median")),
    ('scaler', StandardScaler()),
    ('linear_regression', LinearRegression())
])

pipeline.fit(X_train, y_train)

y_test_pred = pipeline.predict(X_test)
```

# Aprendizado de Máquina e Reconhecimento de Padrões 2021.2

## Scikit-Learn Design Principles

Prof. Samuel Martins (Samuka)
*samuel.martins@ifsp.edu.br*



INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
São Paulo
Campus Campinas