Introduction to regression

SUPERVISED LEARNING WITH SCIKIT-LEARN

Andreas Müller
Core developer, scikit-learn

Boston housing data

```
boston = pd.read_csv('boston.csv')
print(boston.head())
```

```
      CRIM    ZN  INDUS  CHAS     NX     RM   AGE     DIS  RAD    TAX  \\
0  0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296.0
1  0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242.0
2  0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242.0
3  0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222.0
4  0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222.0
   PTRATIO       B  LSTAT  MEDV
0     15.3  396.90   4.98  24.0
1     17.8  396.90   9.14  21.6
2     17.8  392.83   4.03  34.7
3     18.7  394.63   2.94  33.4
4     18.7  396.90   5.33  36.2
```

Creating feature and target arrays

```
X = boston.drop('MEDV', axis=1).values
y = boston['MEDV'].values
```

axis = 1 #represents rows
axis = 0 #represents columns.

Predicting house value from a single feature

```
X_rooms = X[:,5]
type(X_rooms), type(y)
```

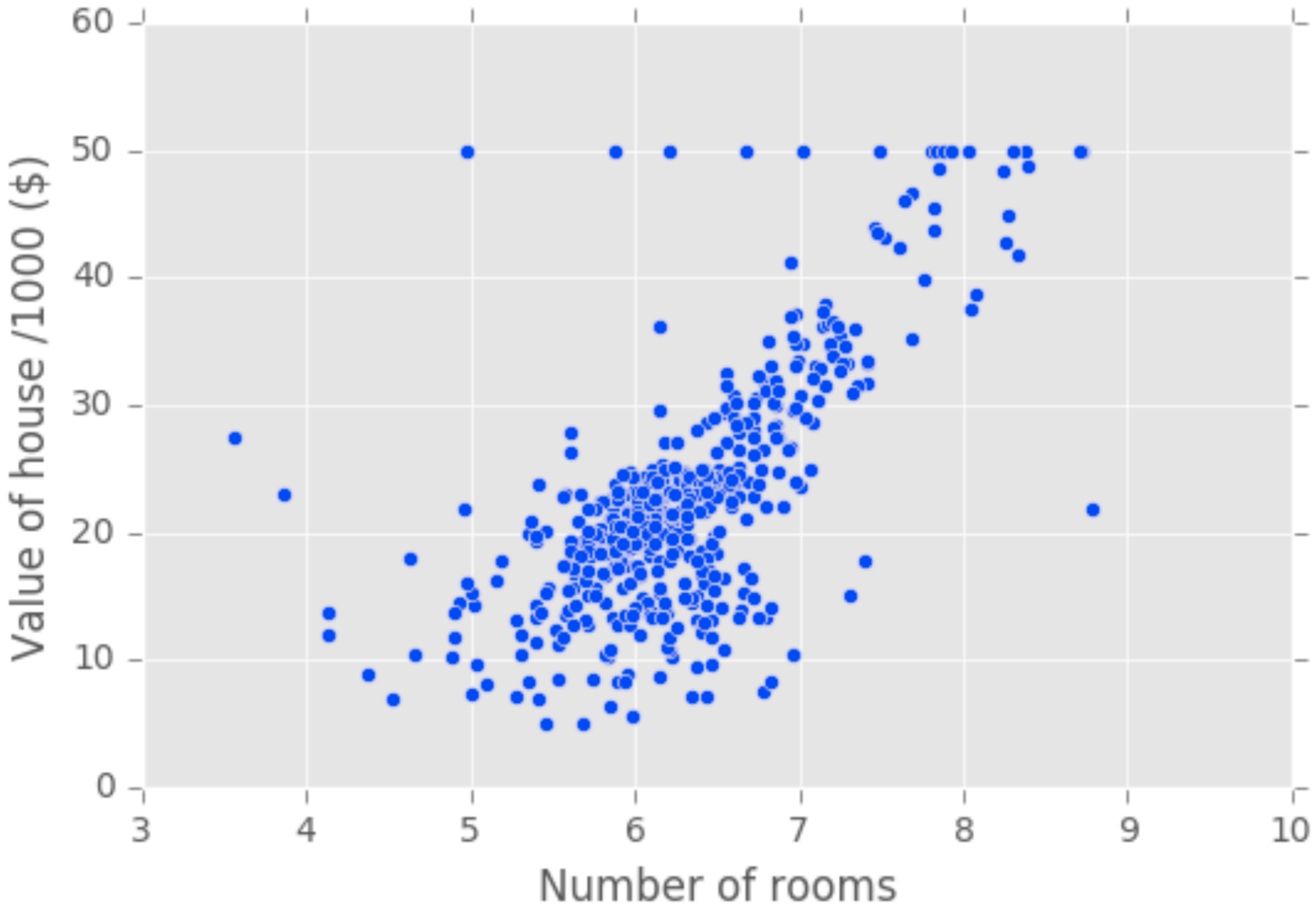
```
(numpy.ndarray, numpy.ndarray)
```

```
y = y.reshape(-1, 1)
X_rooms = X_rooms.reshape(-1, 1)
```

Plotting house value vs. number of rooms

```
plt.scatter(X_rooms, y)
plt.ylabel('Value of house /1000 ($)')
plt.xlabel('Number of rooms')
plt.show();
```

Plotting house value vs. number of rooms



Fitting a regression model

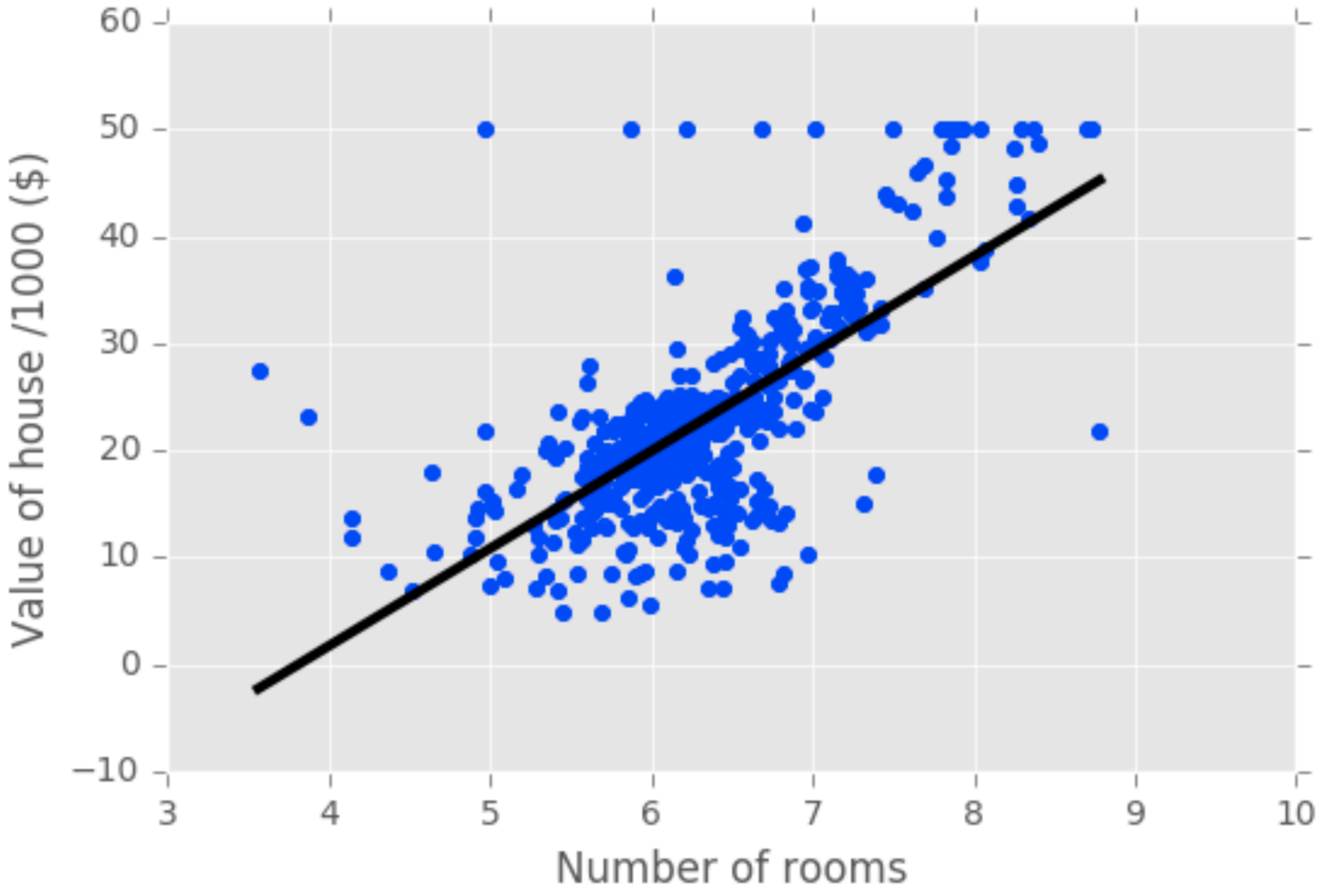
```
import numpy as np
from sklearn.linear_model import LinearRegression

reg = LinearRegression()
reg.fit(X_rooms, y)
prediction_space = np.linspace(min(X_rooms),
                               max(X_rooms)).reshape(-1, 1)
```

linspace is an in-built function in Python's NumPy library. It is used to create an evenly spaced sequence in a specified interval

```
plt.scatter(X_rooms, y, color='blue')
plt.plot(prediction_space, reg.predict(prediction_space),
         color='black', linewidth=3)
plt.show()
```

Fitting a regression model



Let's practice!

SUPERVISED LEARNING WITH SCIKIT-LEARN

The basics of linear regression

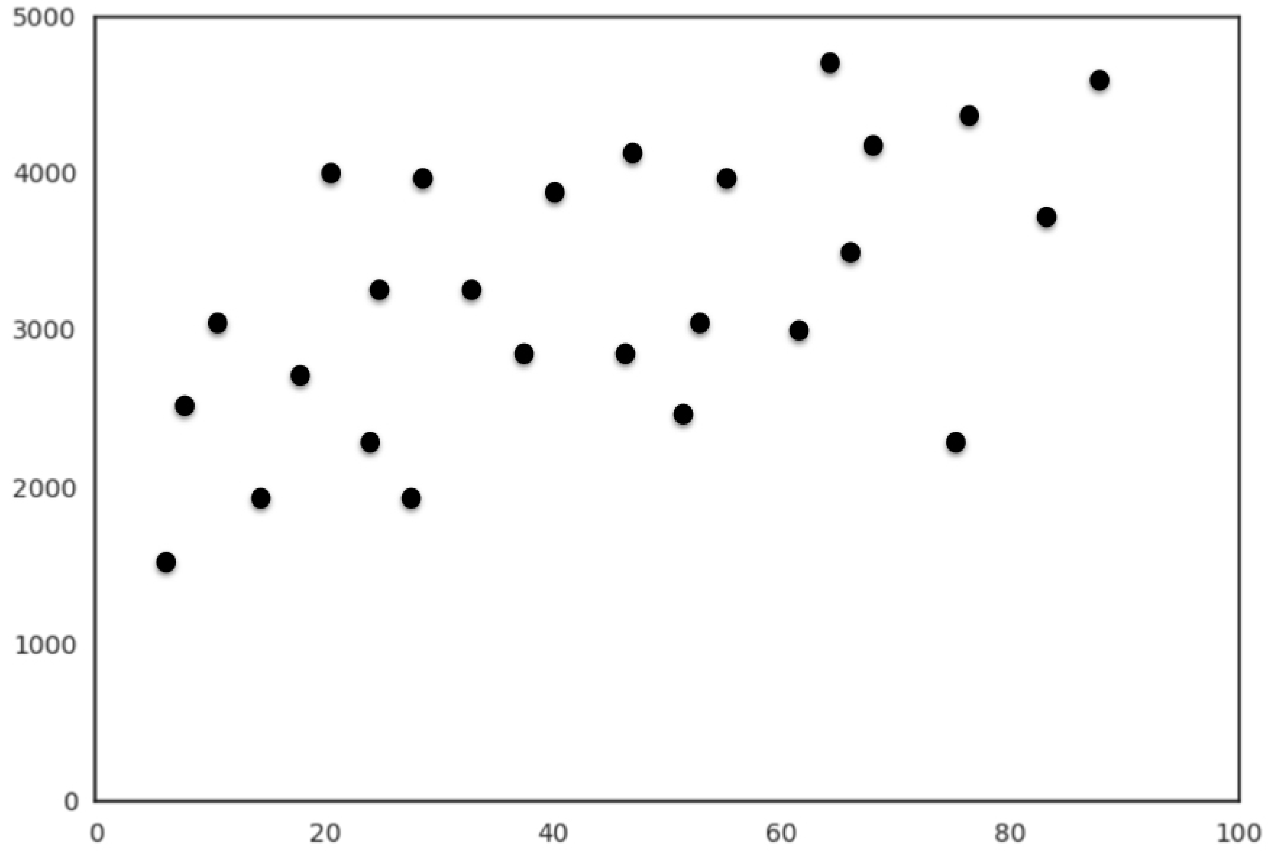
SUPERVISED LEARNING WITH SCIKIT-LEARN

Hugo Bowne-Anderson
Data Scientist, DataCamp

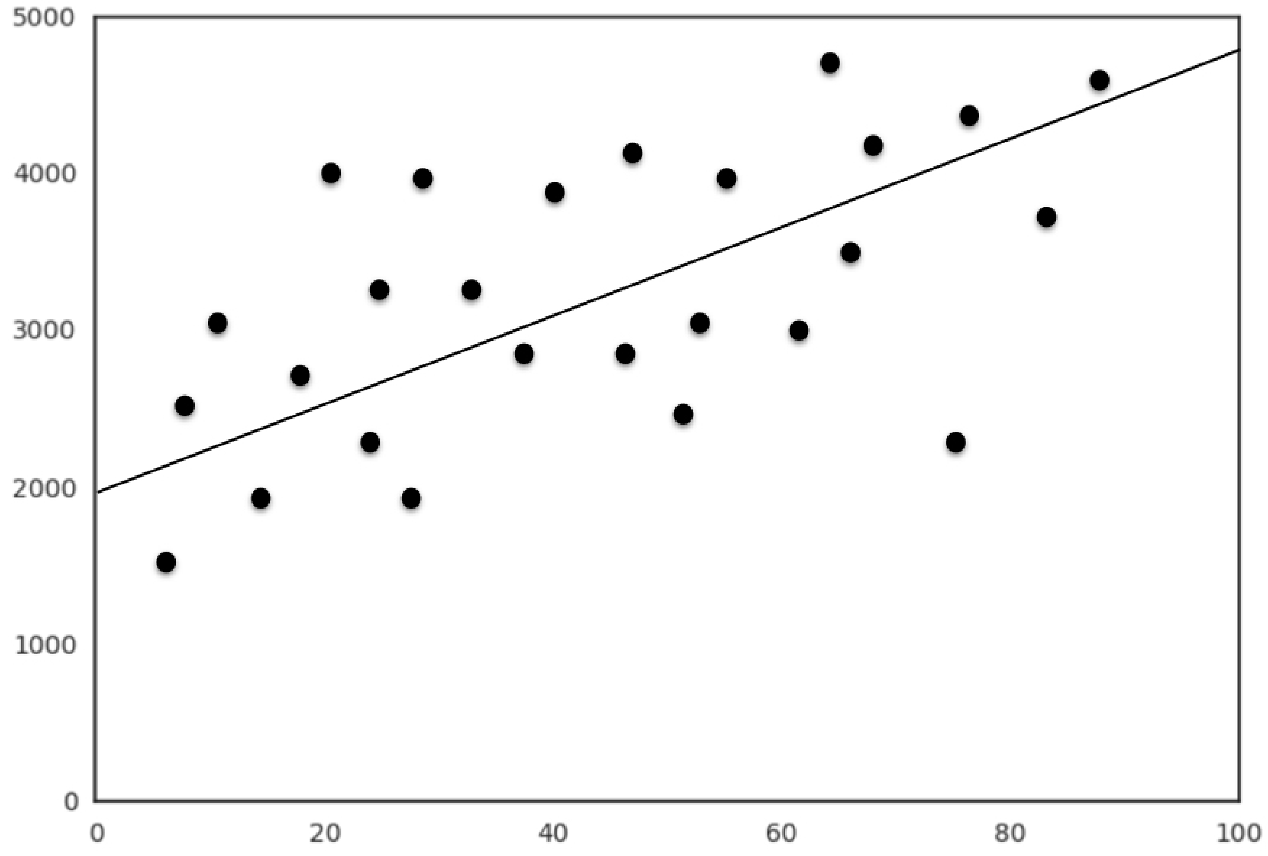
Regression mechanics

- $\boxed{y = ax + b}$
  - y = target
  - x = single feature
  - a, b = parameters of model
- How do we choose a and b?
- Define an error functions for any given line
  - Choose the line that minimizes the error function

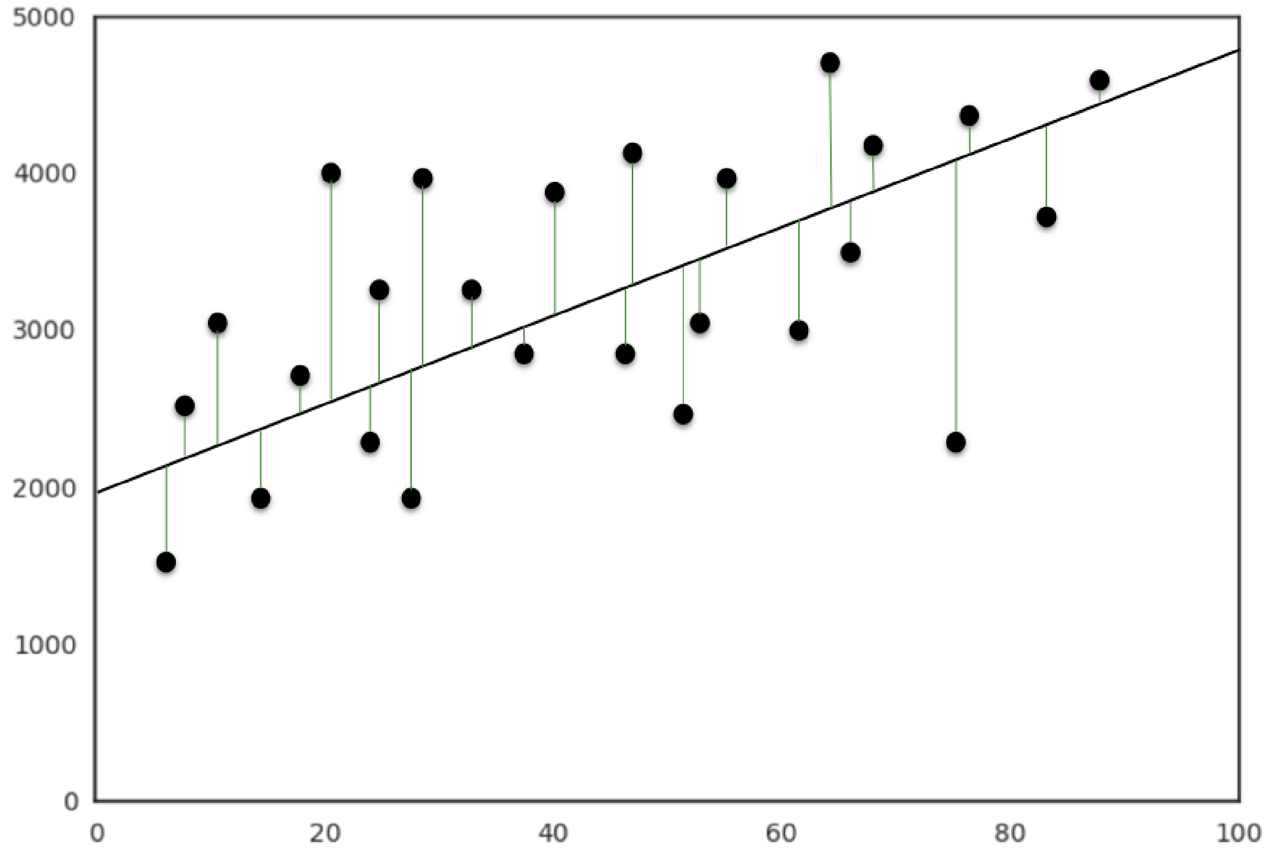
The loss function



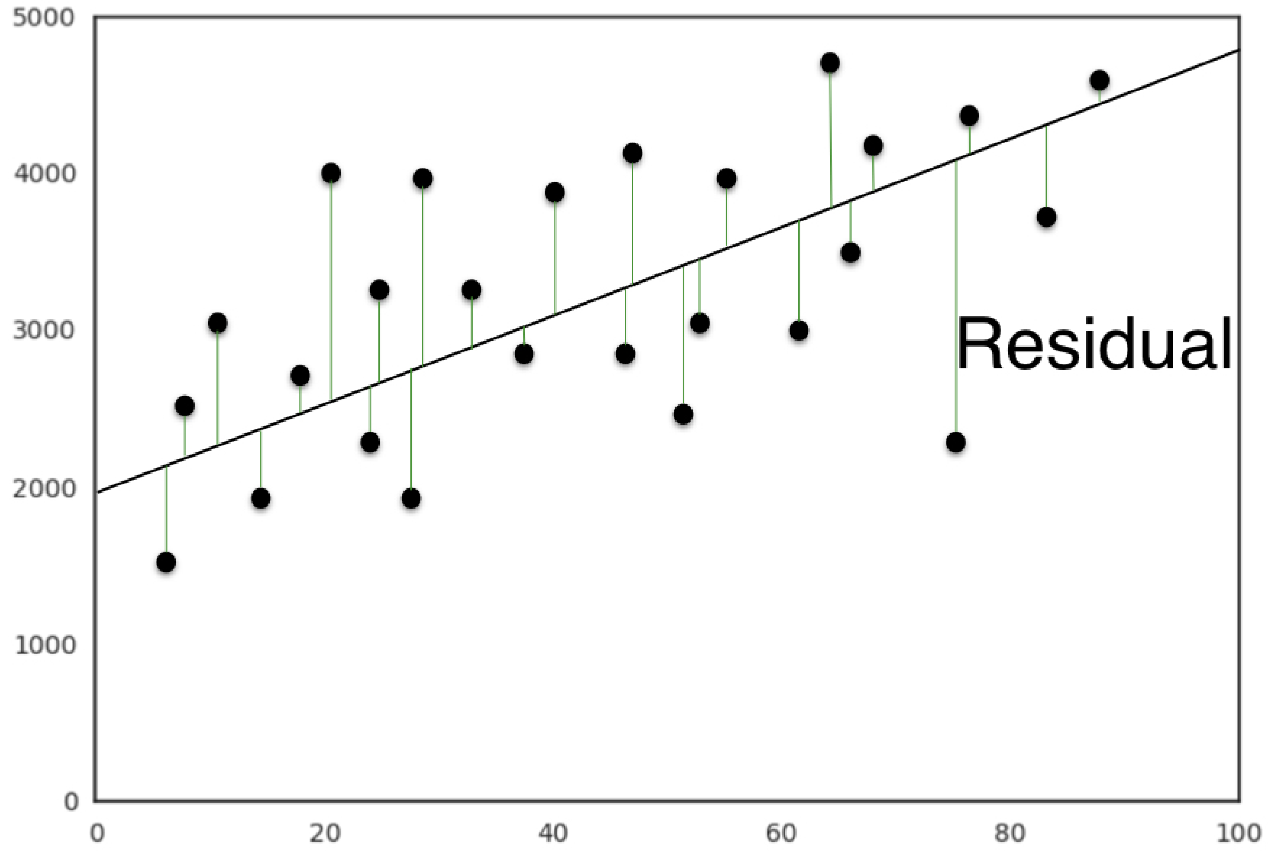
The loss function



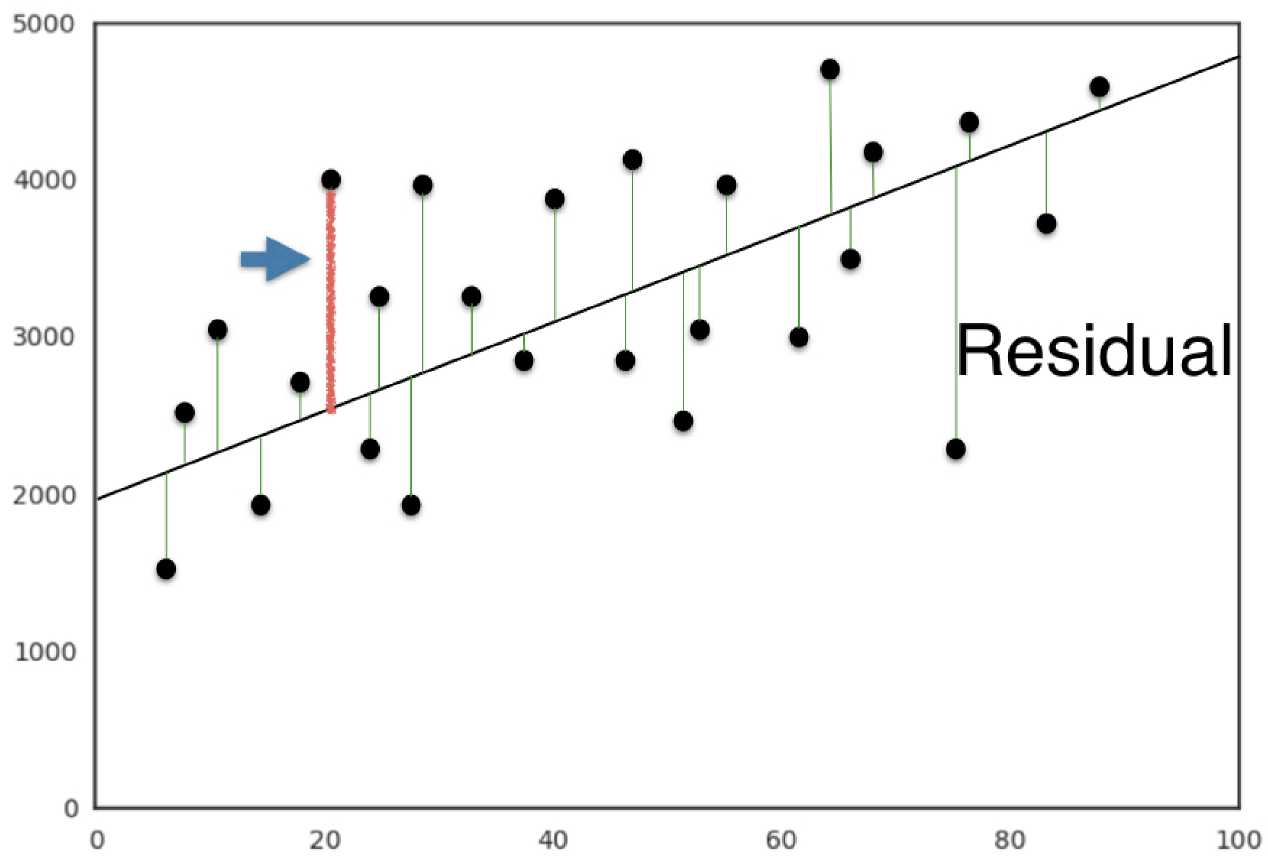
The loss function



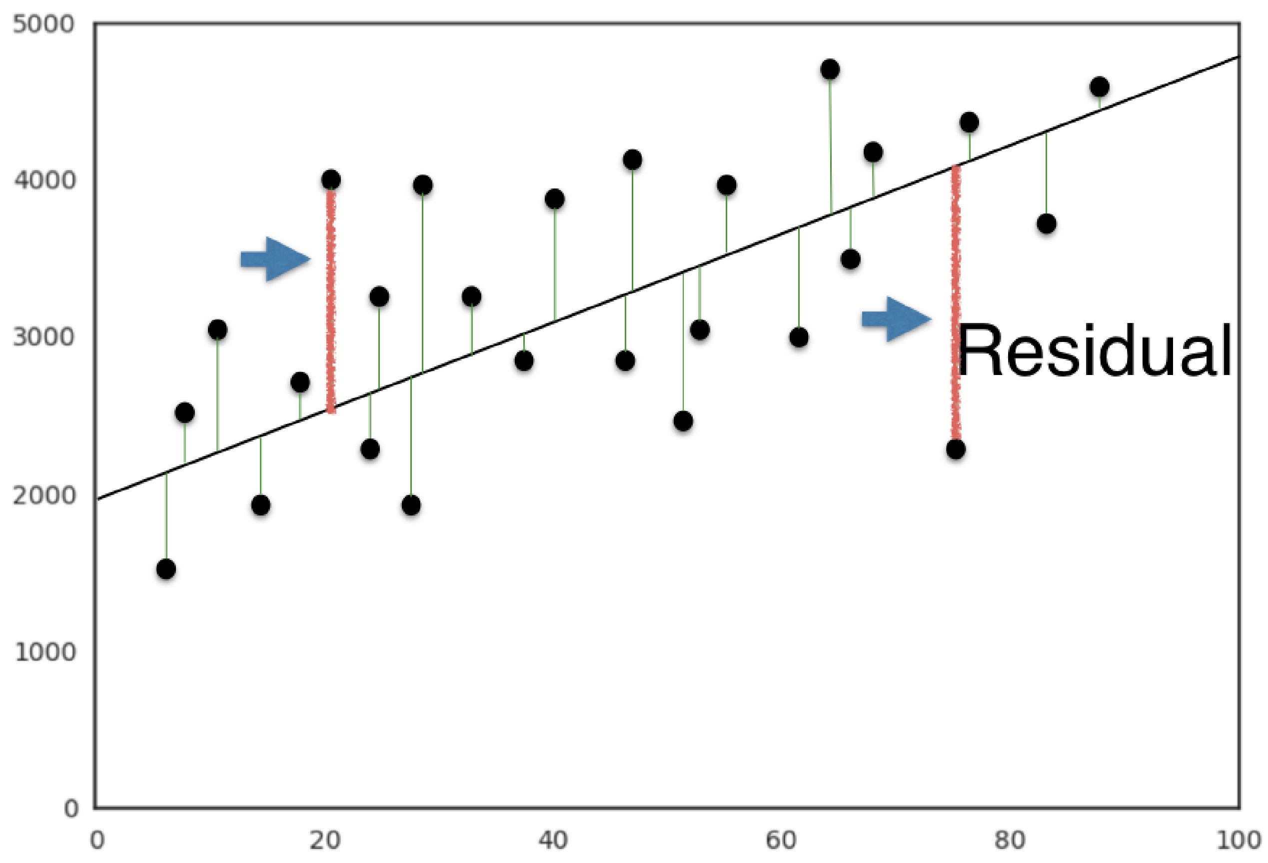
The loss function



The loss function



The loss function



Ordinary least squares(OLS): Minimize sum of squares of residuals

Linear regression in higher dimensions

$$y = a_1x_1 + a_2x_2 + b$$

- To fit a linear regression model here:
  - Need to specify 3 variables
- In higher dimensions:
  - Must specify coefficient for each feature and the variable b

$$y = a_1x_1 + a_2x_2 + a_3x_3 + ... + a_nx_n + b$$

- Scikit-learn API works exactly the same way:
  - Pass two arrays: Features, and target

Linear regression on all features

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size = 0.3, random_state=42)
reg_all = LinearRegression()
reg_all.fit(X_train, y_train)
y_pred = reg_all.predict(X_test)
reg_all.score(X_test, y_test)
```

```
0.7112260057484926
```

Let's practice!

Cross-validation

SUPERVISED LEARNING WITH SCIKIT-LEARN

Hugo Bowne-Anderson
Data Scientist, DataCamp

Cross-validation motivation

- Model performance is dependent on way the data is split
- Not representative of the model's ability to generalize
- Solution: Cross-validation!

- Cross Validation is used to assess the predictive performance of the models and and to judge how they perform outside the sample to a new data set also known as test data.

- The motivation to use cross validation techniques is that when we fit a model, we are fitting it to a training dataset.

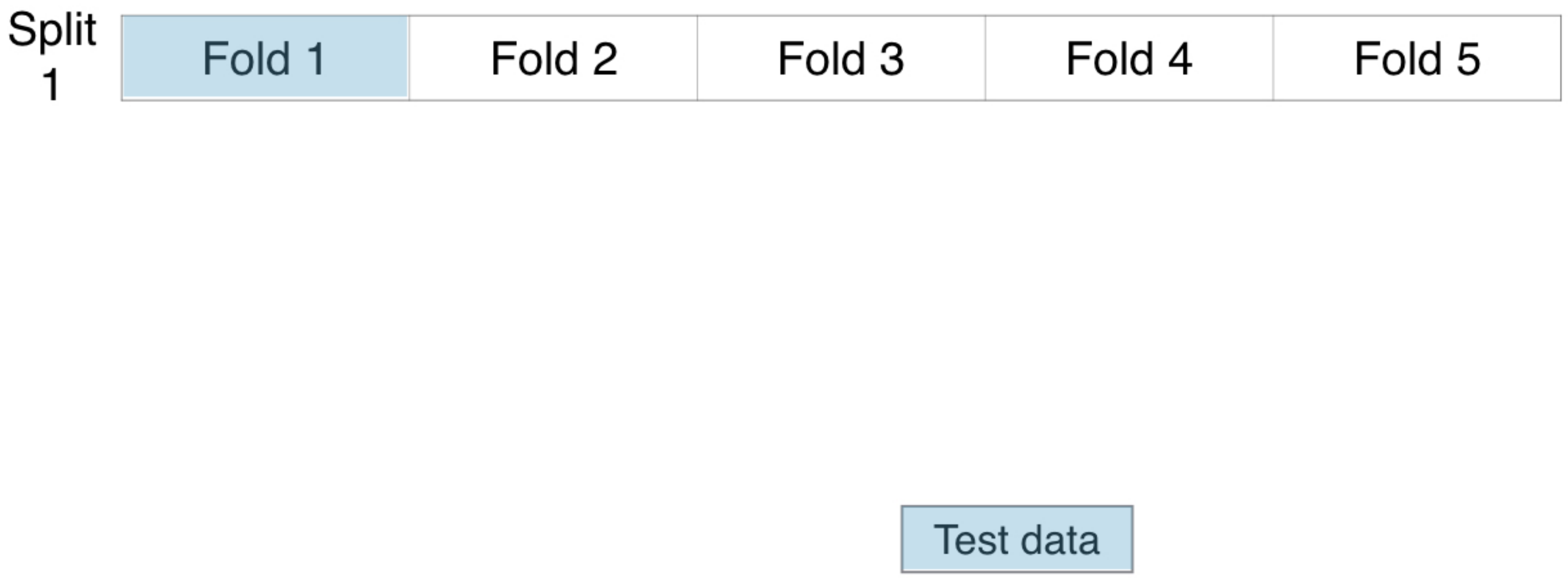
- Cross-validation is a powerful preventative measure against overfitting. The idea is clever: Use your initial training data to generate multiple mini train-test splits. Use these splits to tune your model. In standard k-fold cross-validation, we partition the data into k subsets, called folds.

- k-fold cross classification is about estimating the accuracy, not improving the accuracy. Most implementations of k-fold cross validation give you an estimate of how accurately they are measuring your accuracy:

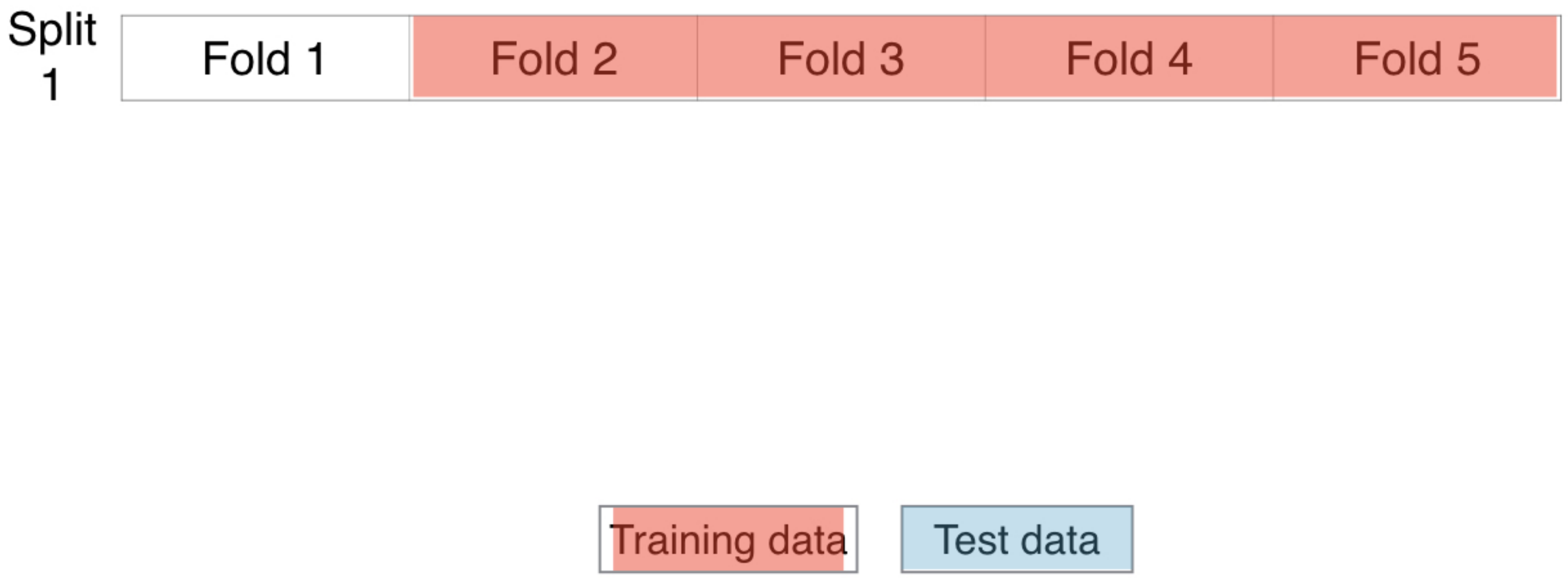
Cross-validation basics

Split 1	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5

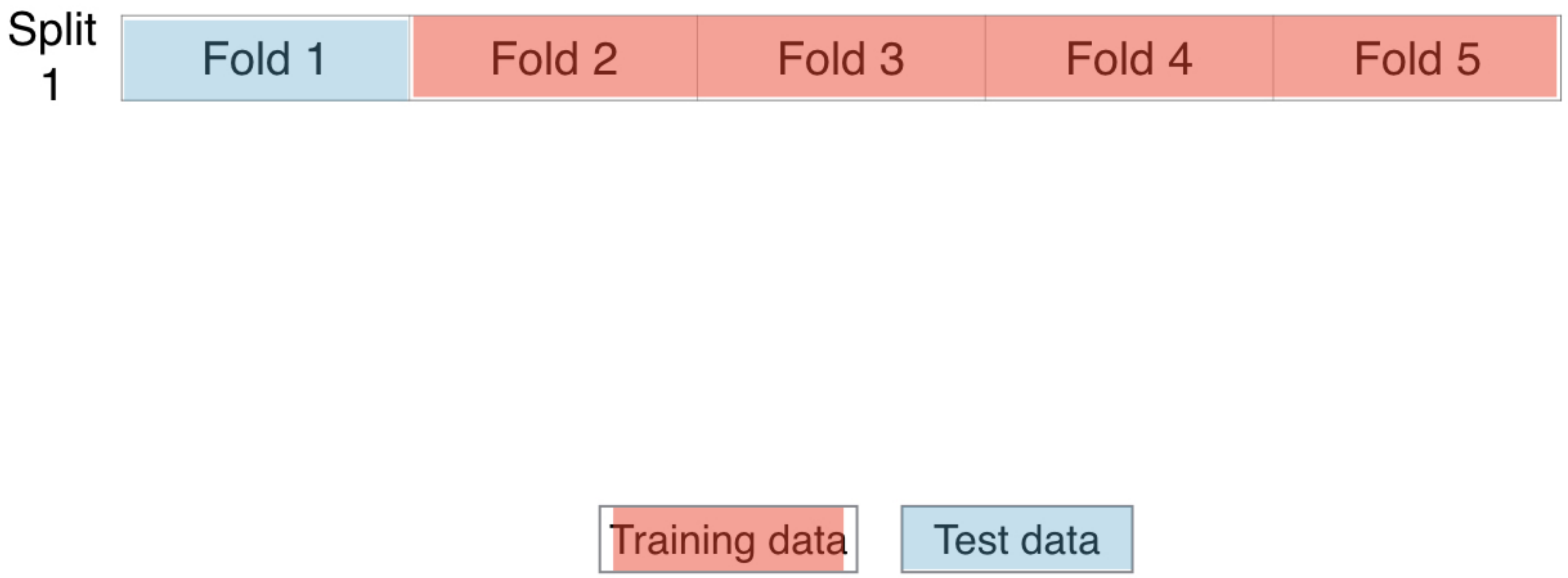
Cross-validation basics



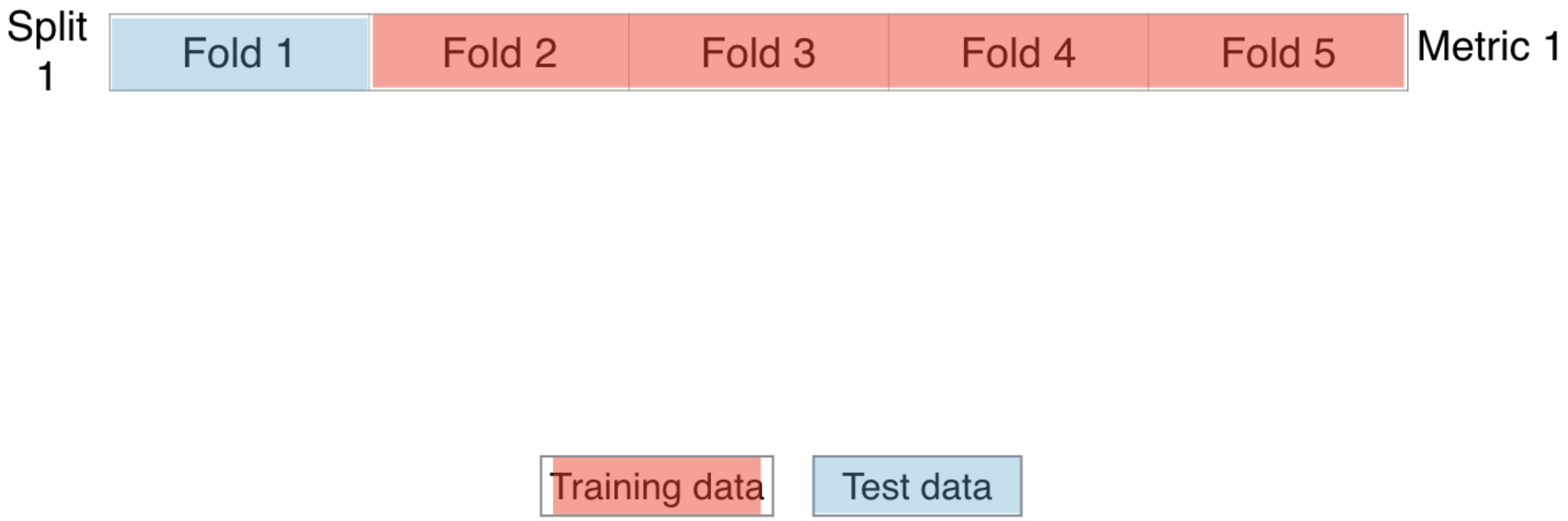
Cross-validation basics



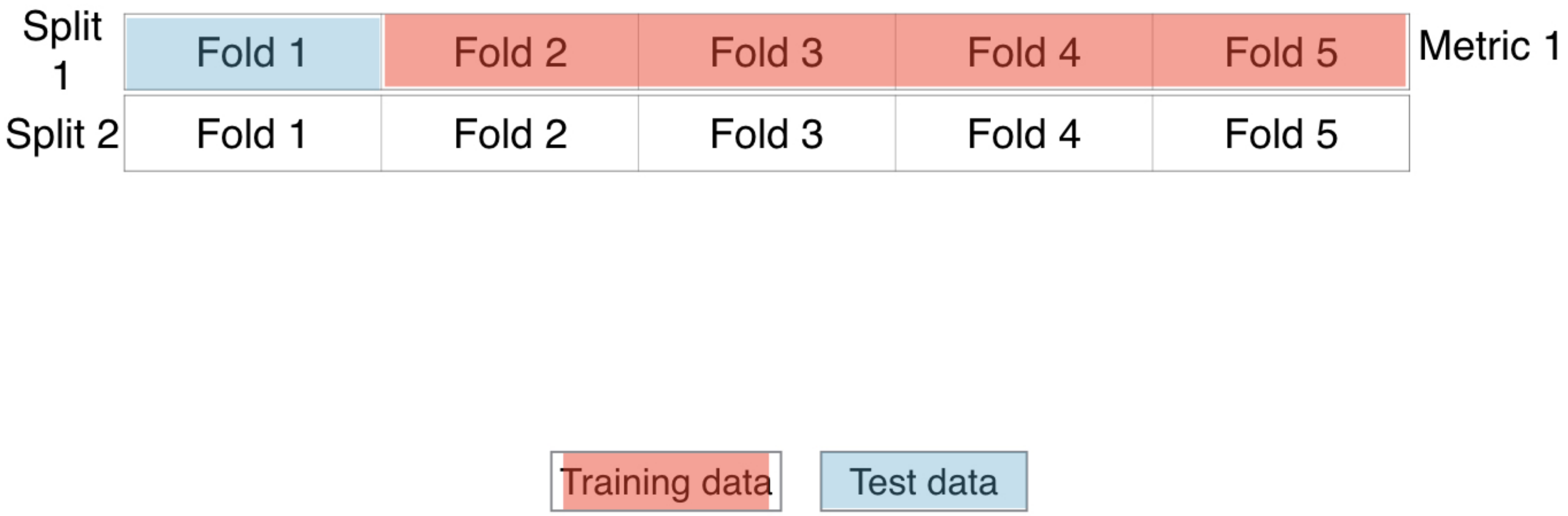
Cross-validation basics



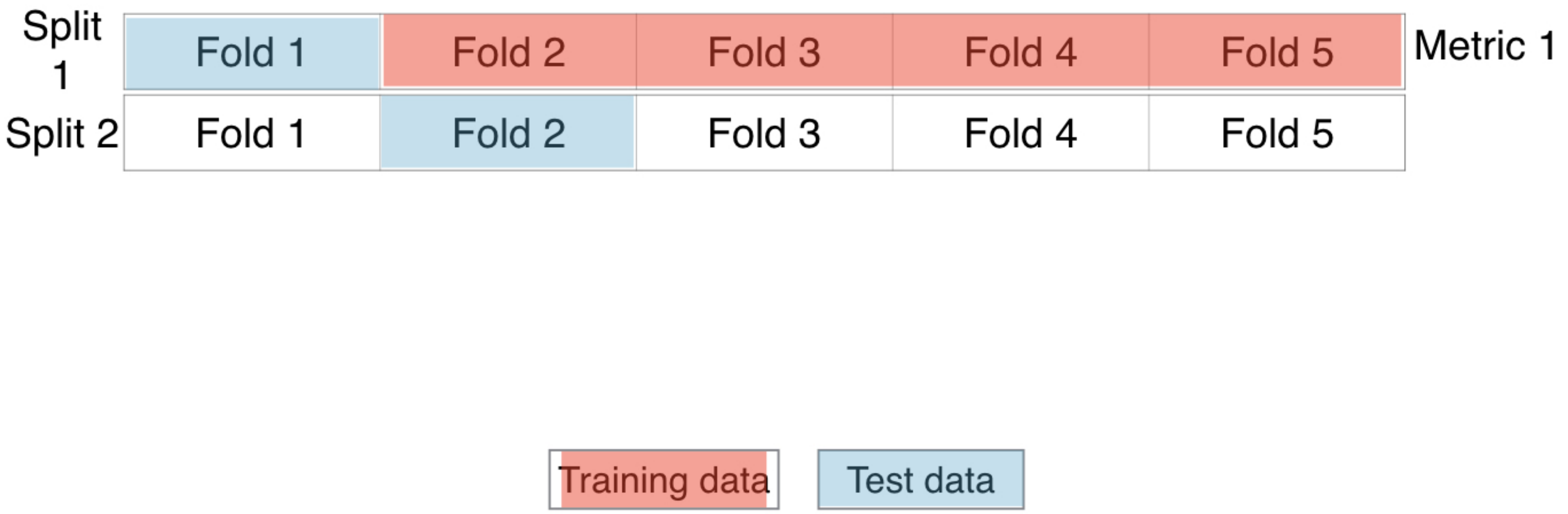
Cross-validation basics



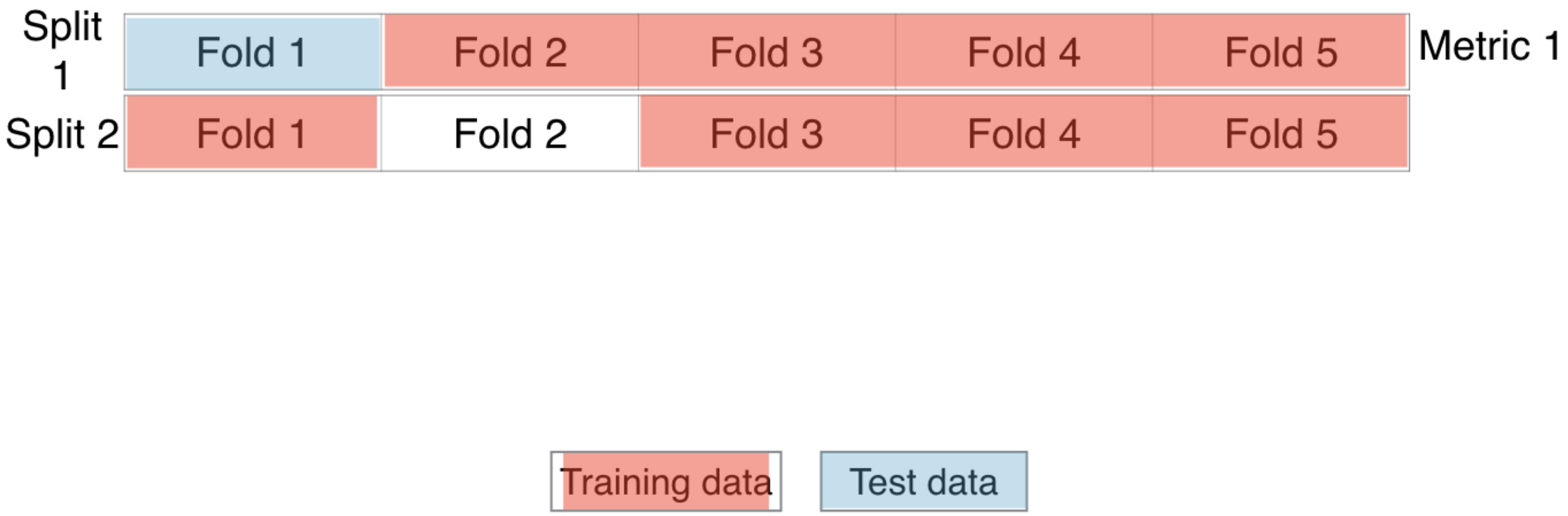
Cross-validation basics



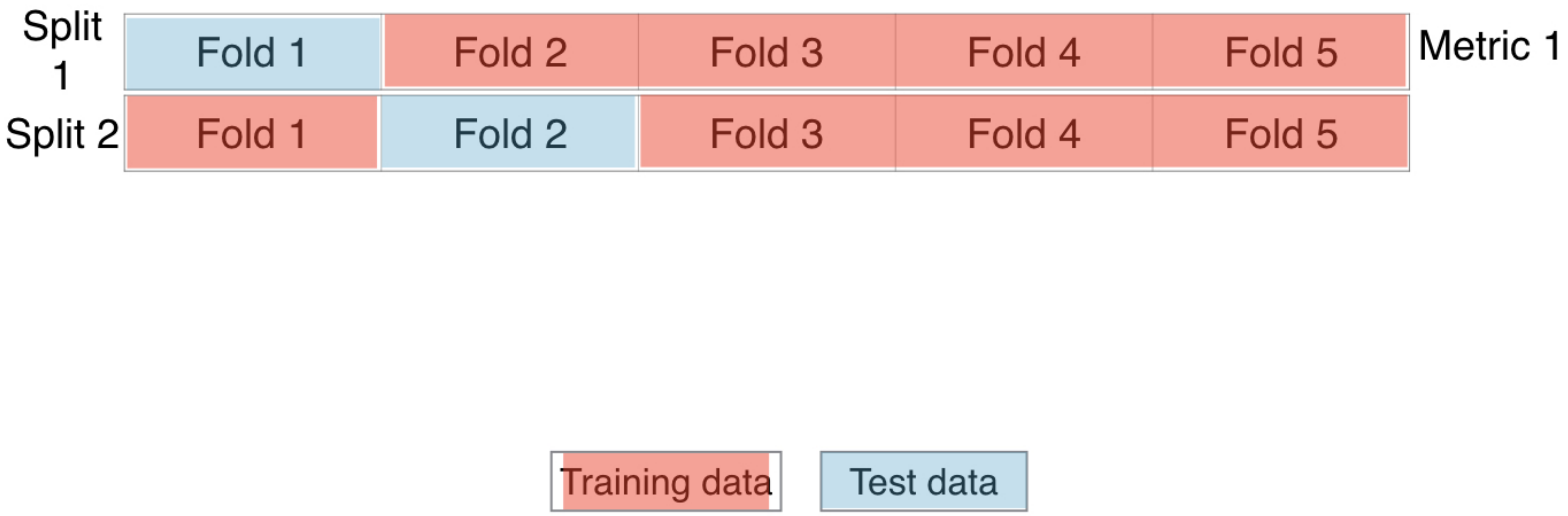
Cross-validation basics



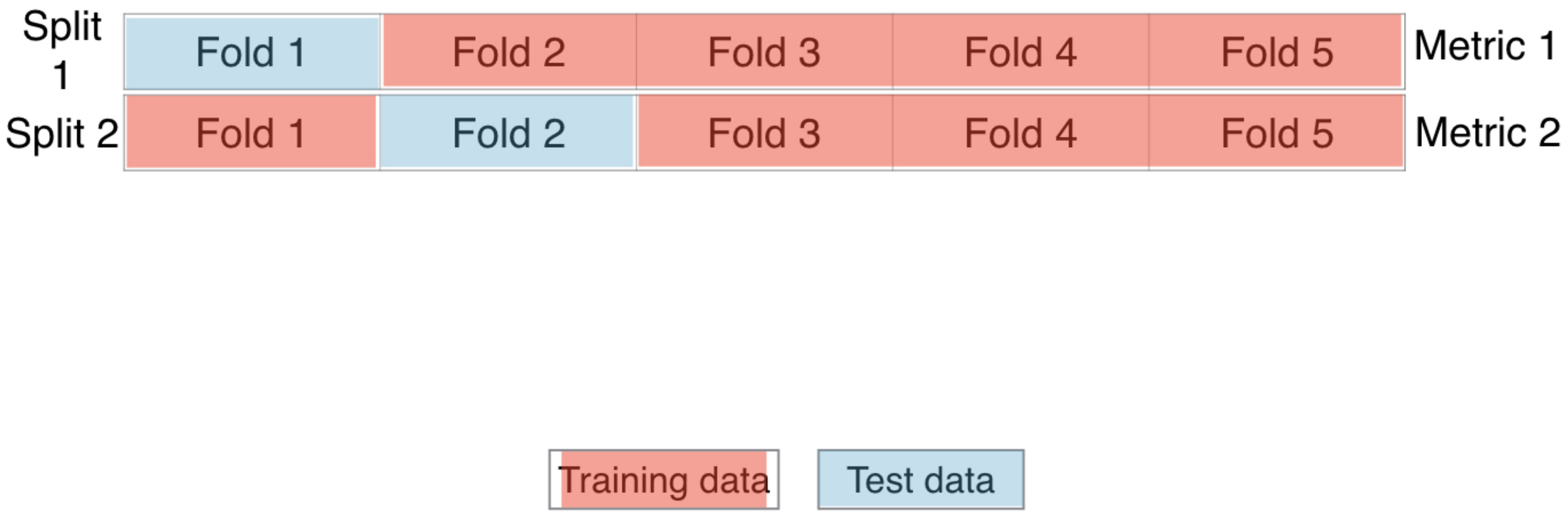
Cross-validation basics



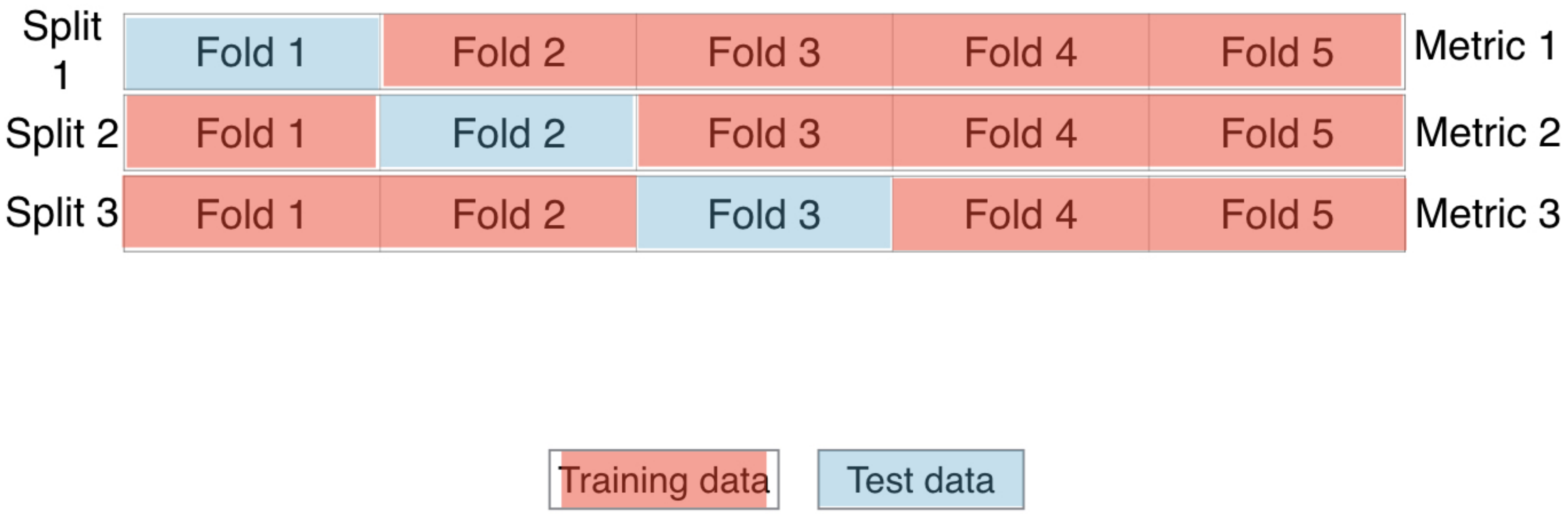
Cross-validation basics



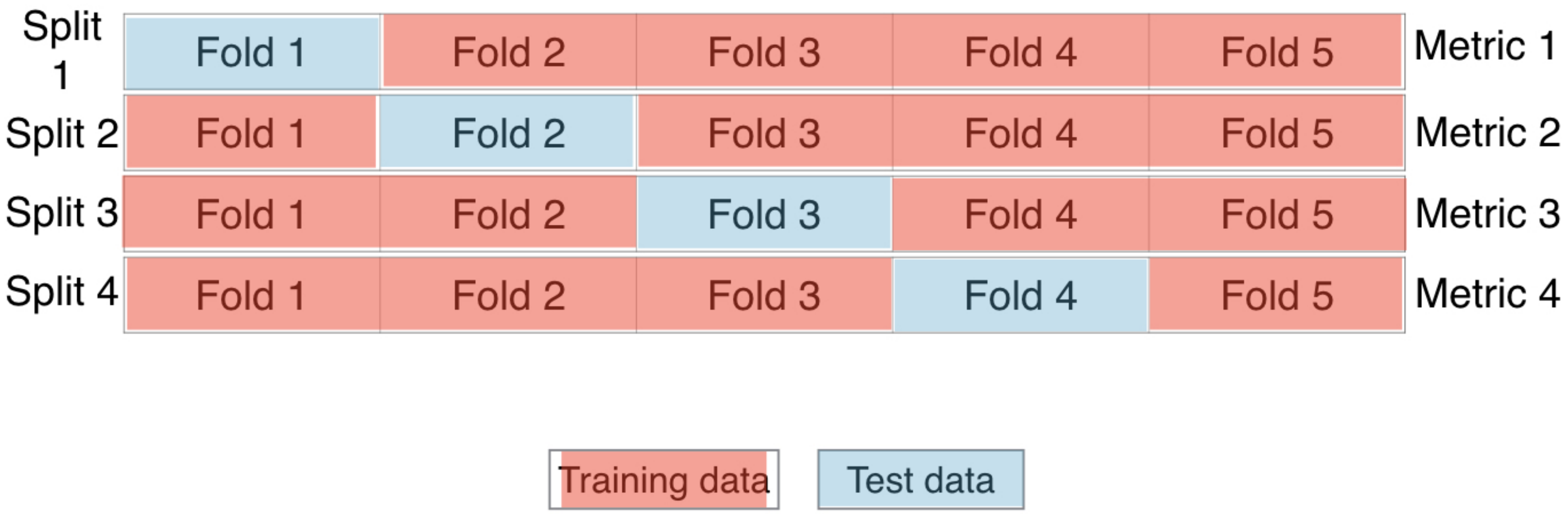
Cross-validation basics



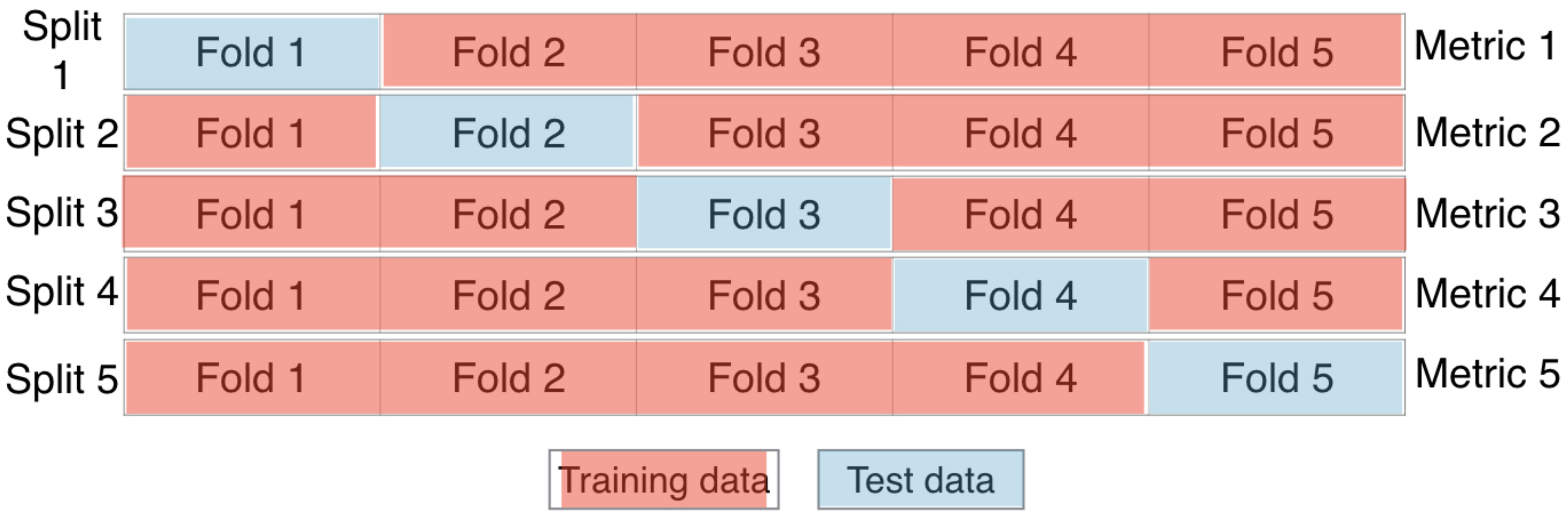
Cross-validation basics



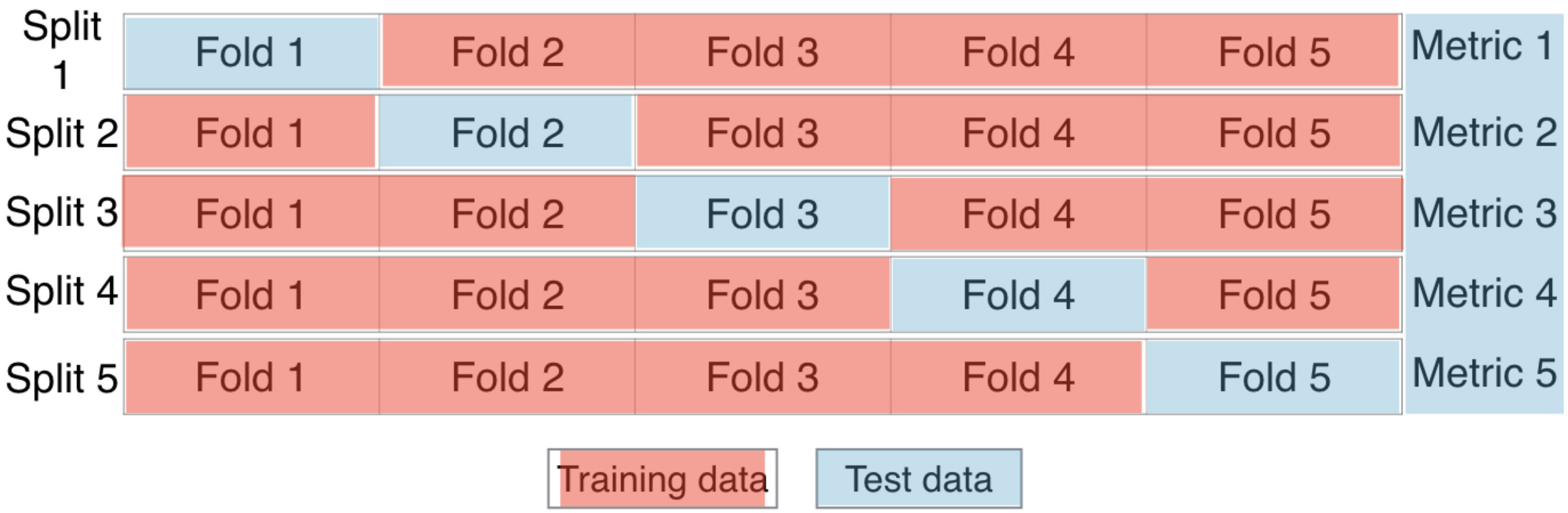
Cross-validation basics



Cross-validation basics



Cross-validation basics



Cross-validation and model performance

- 5 folds = 5-fold CV
- 10 folds = 10-fold CV
- k folds = k-fold CV
- More folds = More computationally expensive

Cross-validation in scikit-learn

```
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
cv_results = cross_val_score(reg, X, y, cv=5)
print(cv_results)
```

```
[ 0.63919994  0.71386698  0.58702344  0.07923081 -0.25294154]
```

```
np.mean(cv_results)
```

```
0.35327592439587058
```

Let's practice!

SUPERVISED LEARNING WITH SCIKIT-LEARN

Regularized regression

SUPERVISED LEARNING WITH SCIKIT-LEARN

Hugo Bowne-Anderson
Data Scientist, DataCamp

Why regularize?

- Recall: Linear regression minimizes a loss function
- It chooses a coefficient for each feature variable
- Large coefficients can lead to overfitting
- Penalizing large coefficients: Regularization

Ridge and Lasso Regression are types of Regularization techniques. Regularization techniques are used to deal with overfitting and when the dataset is large. Ridge and Lasso Regression involve adding penalties to the regression function.

Ridge regression

- Loss function = OLS loss function +

$$\alpha * \sum_{i=1}^{n} {a_i}^2$$

- Alpha: Parameter we need to choose
- Picking alpha here is similar to picking k in k-NN
- Hyperparameter tuning (More in Chapter 3)
- Alpha controls model complexity
    - Alpha = 0: We get back OLS (Can lead to overfitting)
    - Very high alpha: Can lead to underfitting

Ridge Regression is a technique for analyzing multiple regression data that suffer from multicollinearity. By adding a degree of bias to the regression estimates, ridge regression reduces the standard errors. It is hoped that the net effect will be to give estimates that are more reliable.

Ridge regression is a term used to refer to a linear regression model whose coefficients are not estimated by ordinary least squares (OLS), but by an estimator, called ridge estimator, that is biased but has lower variance than the OLS estimator.

Ridge regression in scikit-learn

```
from sklearn.linear_model import Ridge
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size = 0.3, random_state=42)
ridge = Ridge(alpha=0.1, normalize=True)
ridge.fit(X_train, y_train)
ridge_pred = ridge.predict(X_test)
ridge.score(X_test, y_test)
```

```
0.6996938275127313
```

Lasso regression (Least Absolute Shrinkage & Selection Operator)

- Loss function = OLS loss function +

$$\alpha * \sum_{i=1}^{n} |a_i|$$

performs both variable selection and regularization in order to enhance the prediction accuracy and interpretability of the resulting statistical model.

Lasso method overcomes the disadvantage of Ridge regression by not only punishing high values of the coefficients **β** but actually setting them to zero if they are not relevant. Therefore, you might end up with fewer features included in the model than you started with, which is a huge advantage.

The difference between ridge and lasso regression is that it tends to make coefficients to absolute zero as compared to Ridge which never sets the value of coefficient to absolute zero.

Lasso regression in scikit-learn

```
from sklearn.linear_model import Lasso
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size = 0.3, random_state=42)
lasso = Lasso(alpha=0.1, normalize=True)
lasso.fit(X_train, y_train)
lasso_pred = lasso.predict(X_test)
lasso.score(X_test, y_test)
```

```
0.5950229535328551
```

Lasso regression for feature selection

- Can be used to select important features of a dataset
- Shrinks the coefficients of less important features to exactly 0

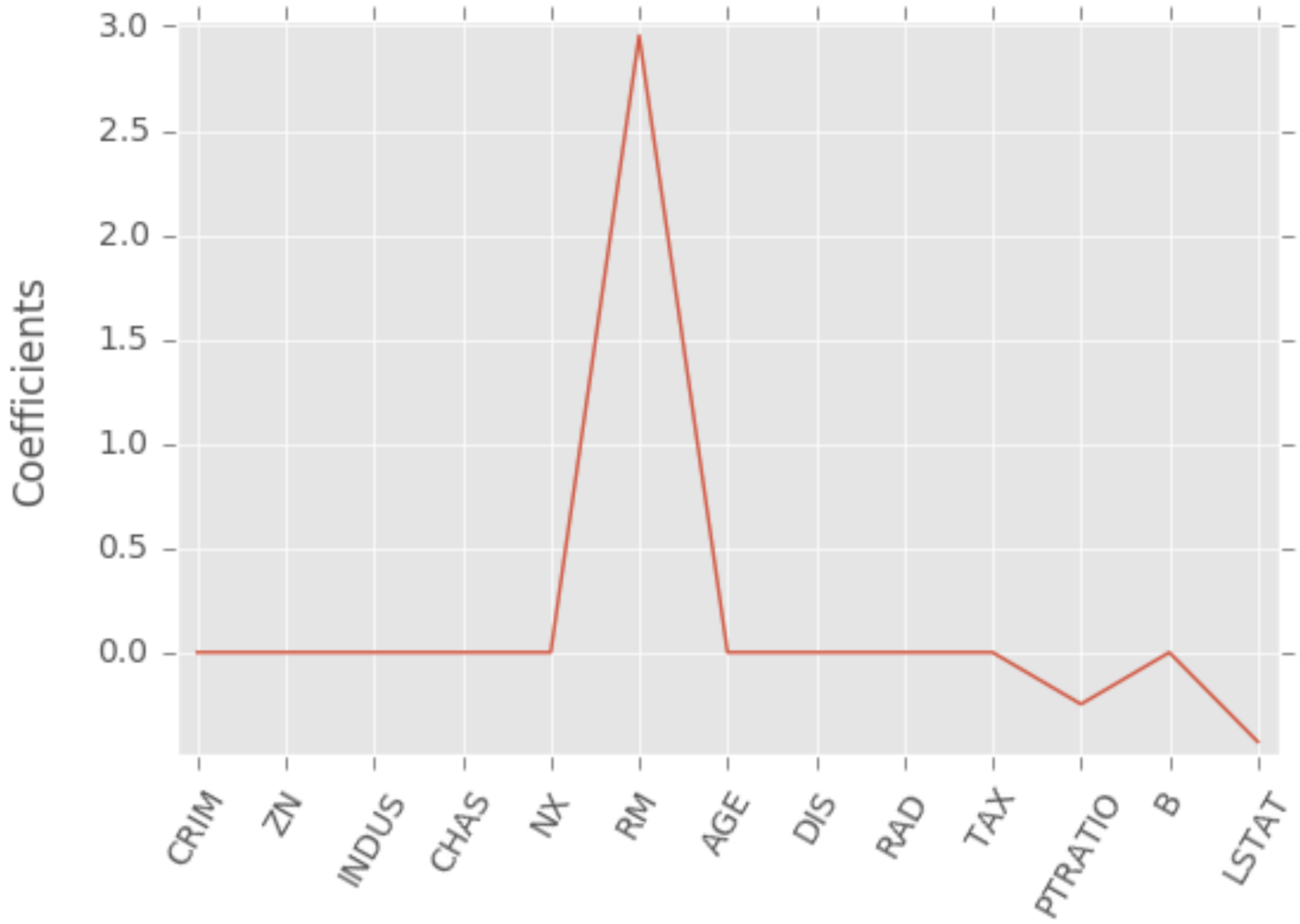
Lasso for feature selection in scikit-learn

```python
from sklearn.linear_model import Lasso
names = boston.drop('MEDV', axis=1).columns
lasso = Lasso(alpha=0.1)
lasso_coef = lasso.fit(X, y).coef_
_ = plt.plot(range(len(names)), lasso_coef)
_ = plt.xticks(range(len(names)), names, rotation=60)
_ = plt.ylabel('Coefficients')
plt.show()
```

_ is used in the interactive interpreter to store the result of the last evaluation.

Represent the values that we don't care.

Lasso for feature selection in scikit-learn



Let's practice!

SUPERVISED LEARNING WITH SCIKIT-LEARN