Evaluating Models

Learn how to evaluate classification and regression models.

Chapter Goals:

- Learn how to evaluate regression and classification models

A. Making predictions

Each of the models we've worked with has a `predict` function, which is used to predict values for new data observations (i.e. data observations not in the training set).

The code below shows an example of making predictions with a regression decision tree.

```
reg = tree.DecisionTreeRegressor()
# predefined train and test sets
reg.fit(train_data, train_labels)
predictions = reg.predict(test_data)
```

B. Evaluation metrics

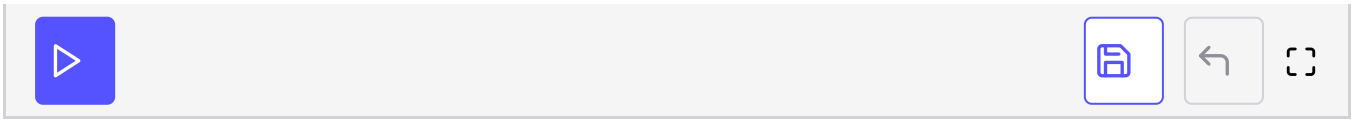
For classification models, we use the classification accuracy on the test set as the evaluation metric. For regression models, we normally use either the R^2 value, mean squared error, or mean absolute error on the test set. The most commonly used regression metric is mean absolute error, since it represents the natural definition of error. We use mean squared error when we want to penalize really bad predictions, since the error is squared. We use the R^2 value when we want to evaluate the fit of the regression model on the data.

The `metrics` module of scikit-learn provides functions for each of these metrics. Each of the evaluation functions takes in the actual testing labels as the first argument and the predictions as the second argument.

The code below evaluates a regression model's predictions based on the testing labels.

```
reg = tree.DecisionTreeRegressor()
# predefined train and test sets
reg.fit(train_data, train_labels)
predictions = reg.predict(test_data)

from sklearn import metrics
r2 = metrics.r2_score(test_labels, predictions)
print('R2: {}\n'.format(r2))
mse = metrics.mean_squared_error(
  test_labels, predictions)
print('MSE: {}\n'.format(mse))
mae = metrics.mean_absolute_error(
  test_labels, predictions)
print('MAE: {}\n'.format(mae))
```



The code below evaluates a classification model's predictions based on the testing labels.

```
clf = tree.DecisionTreeClassifier()
# predefined train and test sets
clf.fit(train_data, train_labels)
predictions = clf.predict(test_data)

from sklearn import metrics
acc = metrics.accuracy_score(test_labels, predictions)
print('Accuracy: {}\n'.format(acc))
```