```
# load dataset
data = datasets.load_iris()

# separate features and target
X = data.data
y = data.target

# initialize LOOCV
loocv = LeaveOneOut()

# create the model
knn_clf = KNeighborsClassifier(n_neighbors=3)

# fit the model using cross validation
cv_result = cross_val_score(knn_clf, X, y, cv=loocv)

# evaluate the model performance using accuracy metric
print("Accuracy: %.3f%% (%.3f%%)" % (cv_result.mean()*100.0, cv_result.
std()*100.0))
'Output':
Accuracy: 96.000% (19.596%)
```

Model Evaluation

This chapter has already used a couple of evaluation metrics for assessing the quality of the fitted models. In this section, we survey a couple of other metrics for regression and classification use cases and how to implement them using Scikit-learn. For each metric, we show how to use them as stand-alone implementations, as well as together with cross-validation using the **cross_val_score** method.

What we'll cover here includes

Regression evaluation metrics

- Mean squared error (MSE): The average sum of squared difference between the predicted label, $\hat{y}$, and the true label, y. A score of 0 indicates a perfect prediction without errors.
- Mean absolute error (MAE): The average absolute difference between the predicted label, $\hat{y}$, and the true label, y. A score of 0 indicates a perfect prediction without errors.

- R^2: The amount of variance or variability in the dataset explained by the model. The score of 1 means that the model perfectly captures the variability in the dataset.

Classification evaluation metrics

- Accuracy: Is the ratio of correct predictions to the total number of predictions. The bigger the accuracy, the better the model.
- Logarithmic loss (a.k.a logistic loss or cross-entropy loss): Is the probability that an observation is correctly assigned to a class label. By minimizing the log-loss, conversely, the accuracy is maximized. So with this metric, values closer to zero are good.
- Area under the ROC curve (AUC-ROC): Used in the binary classification case. Implementation is not provided, but very similar in style to the others.
- Confusion matrix: More intuitive in the binary classification case. Implementation is not provided, but very similar in style to the others.
- Classification report: It returns a text report of the main classification metrics.

Regression Evaluation Metrics

The following code is an example of regression evaluation metrics implemented stand-alone.

```
# import packages
from sklearn.linear_model import LinearRegression
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score

# load dataset
data = datasets.load_boston()
```