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

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

# initialize KFold - with shuffle = True, shuffle the data before splitting
kfold = KFold(n_splits=3, shuffle=True)

# create the model
logistic_reg = LogisticRegression()

# fit the model using cross validation - score with accuracy
accuracy_cv_result = cross_val_score(logistic_reg, X, y, cv=kfold,
scoring="accuracy")
# print accuracy cross validation output
print("Accuracy: %.3f%% (%.3f%%)" % (accuracy_cv_result.mean(), accuracy_
cv_result.std()))
'Output':
Accuracy: 0.953% (0.025%)

# fit the model using cross validation - score with Log-Loss
logloss_cv_result = cross_val_score(logistic_reg, X, y, cv=kfold,
scoring="neg_log_loss")
# print mse cross validation output
print("Log-Loss likelihood: %.3f%% (%.3f%%)" % (logloss_cv_result.mean(),
logloss_cv_result.std()))
'Output':
Log-Loss likelihood: -0.348% (0.027%)
```

Pipelines: Streamlining Machine Learning Workflows

The concept of pipelines in Scikit-learn is a compelling tool for chaining a bunch of operations together to form a tidy process flow of data transforms from one state to another. The operations that constitute a pipeline can be any of Scikit-learn's transformers (i.e., modules with a **fit** and **transform** method, or a **fit_transform** method) or classifiers (i.e., modules with a **fit** and **predict** method, or a **fit_predict** method). Classifiers are also called predictors.

For a typical machine learning workflow, the steps taken may involve cleaning the data, feature engineering, scaling the dataset, and then fitting a model. Pipelines can be used in this case to chain these operations together into a coherent workflow. They have the advantage of providing a convenient and consistent interface for calling at once a sequence of operations.

These transformers or predictors are collectively called estimators in Scikit-learn terminology. In the last two paragraphs, we called them operations.

Another advantage of pipelines is that it safeguards against accidentally fitting a transform on the entire dataset and thereby leaking statistics influenced by the test data to the machine learning model while training. For example, if a standardizer is fitted on the whole dataset, the test set will be compromised because the test observations have contributed in estimating the mean and standard deviation for scaling the training set before fitting the model.

Finally, only the last step of the pipeline can be a classifier or predictor. All the stages of the pipeline must contain a **transform** method except the final stage, which can be a transformer or a classifier.

To begin using Scikit-learn pipelines, first import

from sklearn.pipeline import Pipeline

Let's see some examples of working with Pipelines in Scikit-learn. In the following example, we'll apply a scaling transform to standardize our dataset and then use a support vector classifier to train the model.

```
# import packages
from sklearn.svm import SVC
from sklearn import datasets
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import StandardScaler

from sklearn.pipeline import Pipeline

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

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

# create the pipeline
estimators = [
    ('standardize' , StandardScaler()),
    ('svc', SVC())
]

# build the pipeline model
pipe = Pipeline(estimators)

# run the pipeline
kfold = KFold(n_splits=3, shuffle=True)
cv_result = cross_val_score(pipe, X, y, cv=kfold)

# evaluate the model performance
print("Accuracy: %.3f%% (%.3f%%)" % (cv_result.mean()*100.0, cv_result.
std()*100.0))
'Output':
Accuracy: 94.667% (0.943%)
```

Pipelines Using make_pipeline

Another method for building machine learning pipelines is by using the **make_pipeline** method. For the next example, we use PCA to select the best six features and reduce the dimensionality of the dataset, and then we'll fit the model using Random forests for regression.

```
from sklearn.pipeline import make_pipeline
from sklearn.svm import SVR
from sklearn import datasets
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.decomposition import PCA
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestRegressor
```