```
# feature engineering
linear_reg = LinearRegression()
rfe = RFE(estimator=linear_reg, n_features_to_select=6)
rfe_fit = rfe.fit(X, y)

# print the feature ranking
rfe_fit.ranking_
'Output': array([3, 5, 4, 1, 1, 1, 8, 1, 2, 6, 1, 7, 1])
```

From the result, the 4th, 5th, 6th, 8th, 11th, and 13th features are the top 6 features in the Boston dataset.

Feature Importances

Tree-based or ensemble methods in Scikit-learn have a **feature_importances_** attribute which can be used to drop irrelevant features in the dataset using the **SelectFromModel** module contained in the **sklearn.feature_selection** package.

Let's used the ensemble method **AdaBoostClassifier** in this example.

```
# import packages
from sklearn.ensemble import AdaBoostClassifier
from sklearn.feature_selection import SelectFromModel
from sklearn import datasets

# load dataset
data = datasets.load_iris()

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

# original data shape
X.shape

# feature engineering
ada_boost_classifier = AdaBoostClassifier()
ada_boost_classifier.fit(X, y)
```

```
'Output':
AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None,
          learning_rate=1.0, n_estimators=50, random_state=None)

# print the feature importances
ada_boost_classifier.feature_importances_
'Output': array([0.  , 0.  , 0.58, 0.42])

# create a subset of data based on the relevant features
model = SelectFromModel(ada_boost_classifier, prefit=True)
new_data = model.transform(X)

# the irrelevant features have been removed
new_data.shape
'Output': (150, 2)
```

Resampling Methods

Resampling methods are a set of techniques that involve selecting a subset of the available dataset, training on that data subset, and using the remainder of the data to evaluate the trained model. Let's review the techniques for resampling using Scikit-learn. This section covers

- k-Fold cross-validation
- Leave-one-out cross-validation

k-Fold Cross-Validation

In k-fold cross validation, the dataset is divided into k-parts or folds. The model is trained using $k-1$ folds and evaluated on the remaining kth fold. This process is repeated k-times so that each fold can serve as a test set. At the end of the process, k-fold averages the result and reports a mean score with a standard deviation. Scikit-learn implements K-fold CV in the module **KFold**. The module **cross_val_score** is used to evaluate the cross-validation score using the splitting strategy, which is **KFold** in this case.