```
# print test scores
fit_test.scores_
'Output': array([ 10.81782088,   3.59449902, 116.16984746,  67.24482759])
```

From the test scores, the top 3 important features in the dataset are ranked from feature 3 to 4 to 1 and to 2 in order. The data scientist can choose to drop the second column and observe the effect on the model performance.

We can transform the dataset to subset only the important features.

```
adjusted_features = fit_test.transform(X)
adjusted_features[0:5,:]
'Output':
array([[5.1, 1.4, 0.2],
       [4.9, 1.4, 0.2],
       [4.7, 1.3, 0.2],
       [4.6, 1.5, 0.2],
       [5. , 1.4, 0.2]])
```

The result drops the second column of the dataset.

Recursive Feature Elimination (RFE)

RFE is used together with a learning model to recursively select the desired number of top performing features.

Let's use RFE with **LinearRegression**.

```
# import packages
from sklearn.feature_selection import RFE
from sklearn.linear_model import LinearRegression
from sklearn import datasets

# load dataset
data = datasets.load_boston()

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

```
# 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)
```