**Powerful Feature Selection with Recursive Feature Elimination (RFE) of Sklearn**

Get the same model performance even after dropping 93 features

The basic feature selection methods are mostly about individual properties of features and how they interact with each other. *[Variance thresholding](https://towardsdatascience.com/how-to-use-variance-thresholding-for-robust-feature-selection-a4503f2b5c3f?source=your_stories_page-------------------------------------" \t "_blank)* and *[pairwise feature selection](https://towardsdatascience.com/how-to-use-pairwise-correlation-for-robust-feature-selection-20a60ef7d10?source=your_stories_page-------------------------------------" \t "_blank)* are a few examples that remove unnecessary features based on variance and the correlation between them. However, a more pragmatic approach would select features based on how they affect a particular model’s performance. One such technique offered by Sklearn is Recursive Feature Elimination (RFE). It reduces model complexity by removing features one by one until the optimal number of features is left.

It is one of the most popular feature selection algorithms due to its flexibility and ease of use. The algorithm can wrap around any model, and it produces the best possible set of features that gives the highest performance. By completing this tutorial, you will learn how to use its implementation in Sklearn.

**[Join Medium with my referral link - BEXGBoost](https://ibexorigin.medium.com/membership" \t "_blank)**

[Get exclusive access to all my ⚡premium⚡ content and all over Medium without limits. Support my work by buying me a…](https://ibexorigin.medium.com/membership" \t "_blank)

[ibexorigin.medium.com](https://ibexorigin.medium.com/membership" \t "_blank)

Get the best and latest ML and AI papers chosen and summarized by a powerful AI — Alpha Signal:

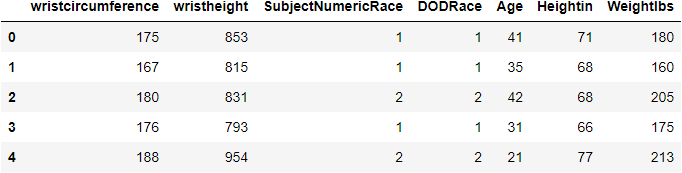
**[Alpha Signal | The Best of Machine Learning. Summarized by AI.](https://alphasignal.ai/?referrer=Bex" \t "_blank)**

[Stay in the loop without spending countless hours browsing for the next breakthrough; our algorithm identifies the…](https://alphasignal.ai/?referrer=Bex" \t "_blank)

[alphasignal.ai](https://alphasignal.ai/?referrer=Bex" \t "_blank)

**The idea behind Recursive Feature Elimination**

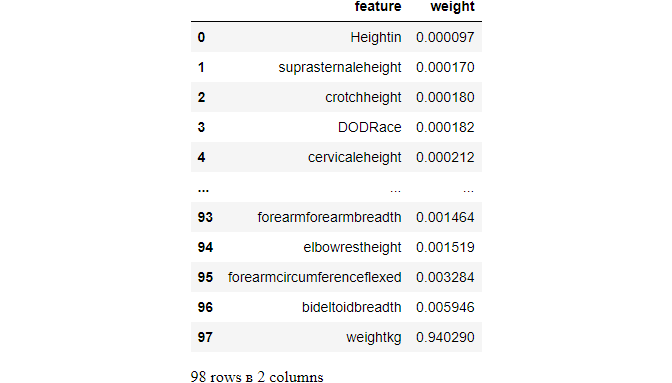
Consider this subset of the [Ansur Male dataset](https://www.kaggle.com/seshadrikolluri/ansur-ii" \t "_blank):



It records more than 100 different types of body measurements of more than 6000 US Army Personnel. Our goal is to predict the weight in pounds using as few features as possible. (There are 93 numeric features in the dataset)

Let’s establish a base performance with Random Forest Regressor. We will first build the feature and target arrays and divide them into train and test sets. Then, we will fit the estimator and score its performance using R-squared:

We achieved an excellent R-squared of 0.948. We could do this using all 98 features, which is much more than we might need. All Sklearn estimators have special attributes that show feature weights (or coefficients), either given as coef\_ or .feature\_importances\_. Let's see the computed coefficients for our Random Forest Regressor model:



To reduce model complexity, always start by removing features with close to 0 weights. Since all weights are multiplied by the values of features, such small weights contribute very little to the overall predictions. Looking at the above weights, we can see that many weights are close to 0.

We could set a low threshold and filter out features based on it. But we have to remember that even removing a single feature forces other coefficients to change. So, we have to eliminate them step-by-step, leaving out the lowest weighted feature by sorting the fitted model coefficients. Doing this manually for 98 features would be cumbersome, but thankfully Sklearn provides us with Recursive Feature Elimination — [RFE class](https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html) to do the task.

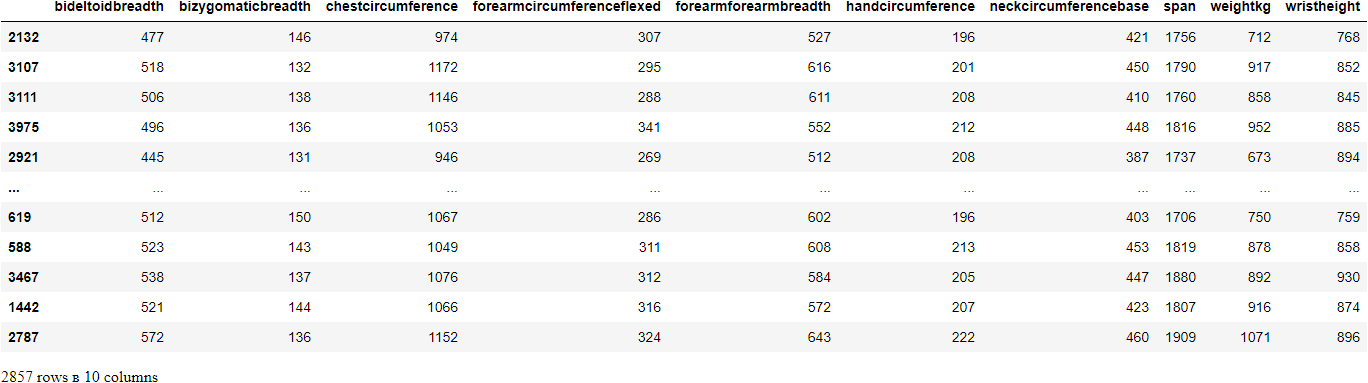
**Sklearn Recursive Feature Elimination Class**

RFE is a transformer estimator, which means it follows the familiar fit/transform pattern of Sklearn. It is a popular algorithm due to its easy configurable nature and robust performance. As the name suggests, it removes features one at a time based on the weights given by a model of our choice in each iteration.

Below, you will see an example of RFE using the above Random Forest Regressor model:

After fitting the estimator, it has a .support\_ attribute that gives a boolean mask with False values for discarded features. We can use it to subset our data:

X\_train.loc[:, rfe.support\_]



Or you can directly call .transform() to get a new numpy array with the relevant features. Let's use this smaller subset to test Random Forest Regressor once again:

Even after dropping almost 90 features, we got the same score which is very impressive!

**RFE Performance Considerations**

Since RFE trains the given model on the full dataset every time it drops a feature, the computation time will be heavy for large datasets with many features as ours. To control this behavior, RFE provides step parameter that lets us drop an arbitrary number of features in each iteration instead of one:

**Choosing the number of features to keep automatically**

The most important hyperparameters of RFE are *estimator* and *n\_features\_to\_select*. In the last example, we arbitrarily chose 10 features and hoped for the best. However, as RFE can be wrapped around any model, we have to choose the number of relevant features based on their performance.

To achieve this, Sklearn provides a similar RFECV class which implements Recursive Feature Elimination with cross-validation and automatically finds the optimal number of features to keep. Below is an example that uses RFECV around a simple Linear Regression. We will be choosing Linear regression because we can guess there will be a linear correlation between body measurements. Besides, combined with cross-validation, Random Forest Regressor will become more computationally expensive:

I provided the default values to cv and scoring parameters. A new hyperparameter is min\_features\_to\_select - you can probably guess what it does from the name. Let's see how many features the estimator computed to keep:

RFECV tells us to keep only 5 out of 98. Let's train the model only on those 5 and look at its performance:

Even after dropping 93 features, we still got an impressive score of 0.956.

**Summary**

By reading this tutorial, you learned:

* the idea behind Recursive Feature Elimination
* how to use the implementation of the algorithm using Sklearn RFE class
* how to decide the number of features to keep automatically using RFECV class

If you want a deeper look at the algorithm, you can read this [post](https://machinelearningmastery.com/rfe-feature-selection-in-python/).