

Disciplina:

MODELAGEM E PREPARAÇÃO DE DADOS PARA APRENDIZADO DE MÁQUINA

Professor: Rafael Barroso



PARTE 1

SELEÇÃO DE VARIÁVEIS

SELEÇÃO DE VARIÁVEIS

MALDIÇÃO DA DIMENSIONALIDADE

- Com o aumento de variáveis, mais esparsos tendem a ser a base de dados e, por conseguinte, o hiperespaço de modelagem;
- Muitos modelos tendem a falhar por haver uma razão de variáveis e observações muito grande;
- A adição de variável aumenta a complexidade do problema de forma exponencial;
- Exemplo do tabuleiro de xadrez:
 - Original: 2 dimensões → 64 posições possíveis (8 x 8)
 - Adicionando 1 dimensão: 3 dimensões (+50%) → 512 posições possíveis (+800%)

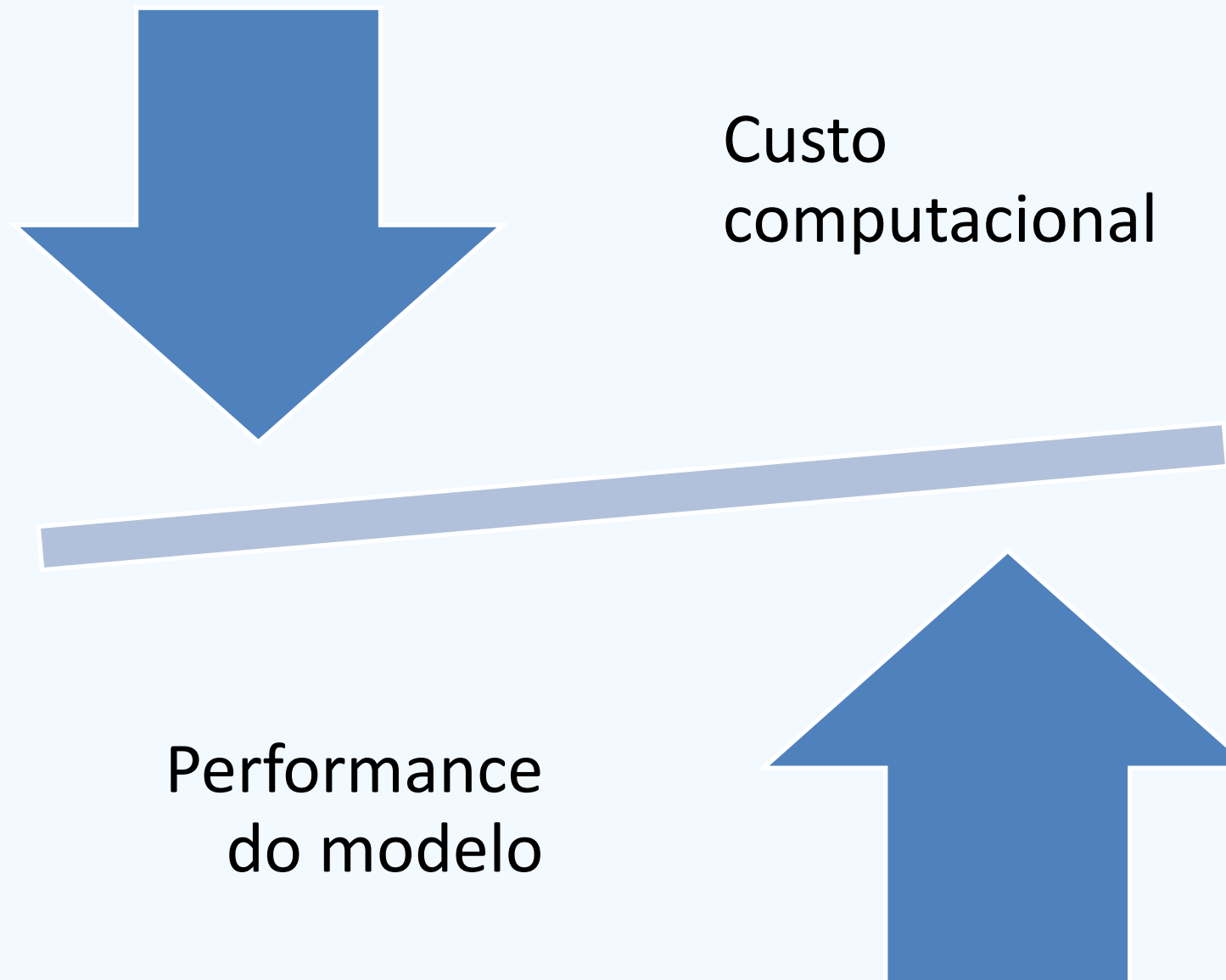
SELEÇÃO DE VARIÁVEIS

MALDIÇÃO DA DIMENSIONALIDADE

- Estatisticamente, a adição de variáveis (dimensões) afasta as observações;
- Num limite, tudo passa a estar tão distante que passa a ser considerado ruído;
- Logo, estruturas e padrões passam a não possíveis de serem identificados;
- Num contexto de *big data*, encontrar e selecionar as variáveis ideais passa a ser essencial para possibilitar o sucesso de um projeto.

SELEÇÃO DE VARIÁVEIS

OBJETIVO



SELEÇÃO DE VARIÁVEIS

TIPOS

Não Supervisionado

- Busca eliminar redundâncias
- Não utiliza a variável resposta
- Faz uso de análises de correlação, *missings* e variância, por exemplo

Supervisionado

- Busca eliminar irrelevâncias
- Utiliza a variável resposta
- Se divide em 3 métodos:
 - Filtros,
 - *Wrappers* e
 - Intrínsecos

SELEÇÃO DE VARIÁVEIS

NÃO SUPERVISIONADO: CORRELAÇÕES

Pearson's

- Mede a relação linear entre variáveis;
- Sempre é um valor entre -1 e +1;
- Quanto mais distante de 0, maior a correlação;
- Idealmente, as 2 variáveis:
 - São contínuas,
 - Têm distribuição normal,
 - Têm relação linear,
 - Devem ter os outliers tratados

Spearman's e Kendall's

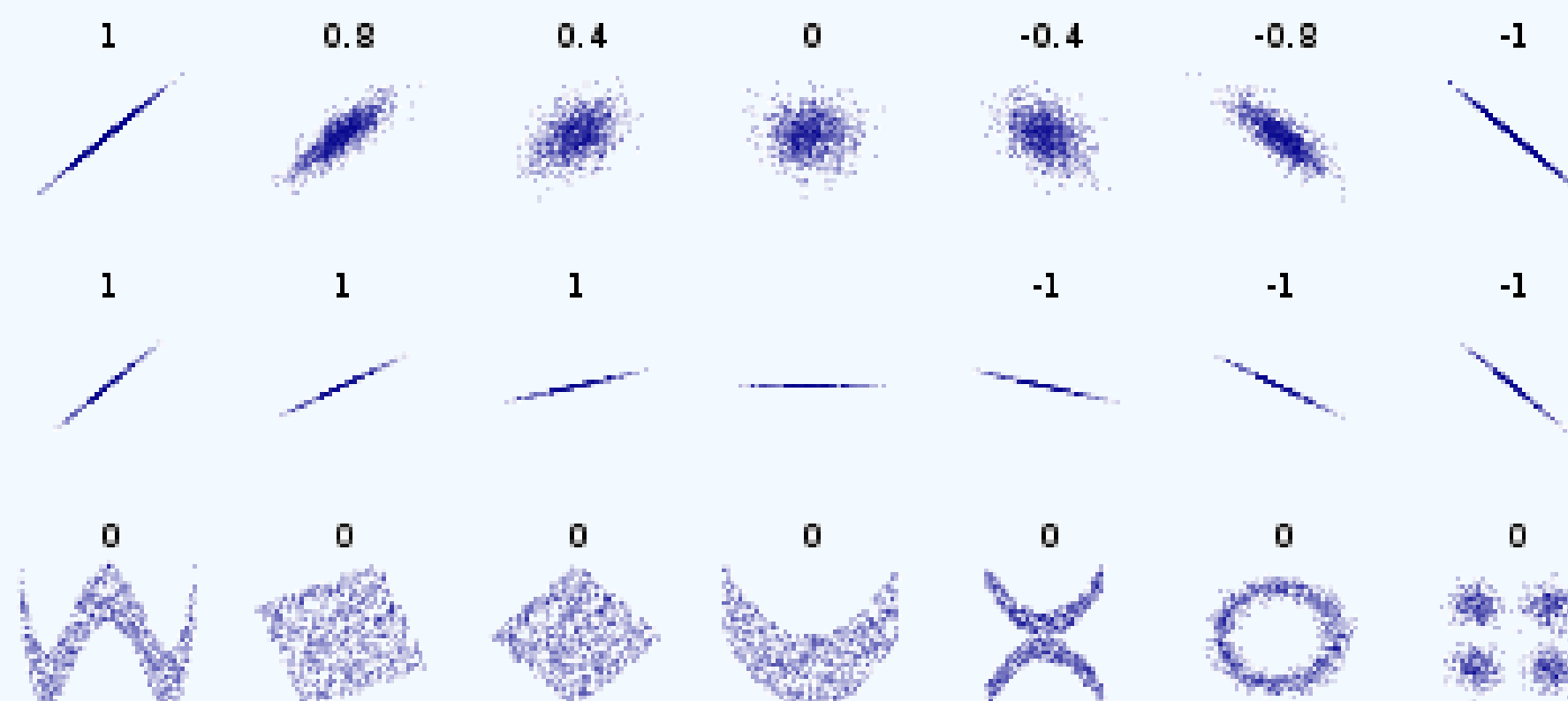
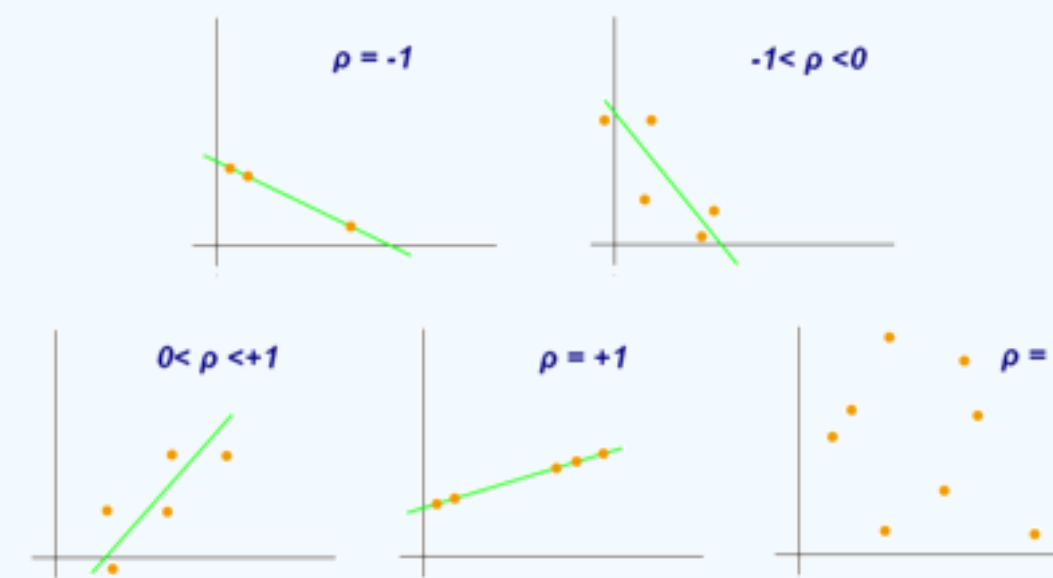
- Medem a relação monotônica entre variáveis;
- Em outras palavras, avaliam a tendência de duas variáveis “moverem na mesma direção”, mas não necessariamente de forma constante;
- Também retornam valores entre -1 e +1;
- São não-paramétricas, ou seja, não exigem comportamento normal das variáveis;
- Não demandam que os dados sejam contínuos (podem ser ordinais);
- Spearman's é matematicamente mais simples.

SELEÇÃO DE VARIÁVEIS

NÃO SUPERVISIONADO: CORRELAÇÕES

Pearson's

- Mede a relação linear entre variáveis;
- Sempre é um valor entre -1 e +1;
- Quanto mais distante de 0, maior a correlação;
- Idealmente, as 2 variáveis:
 - São contínuas,
 - Têm distribuição normal,
 - Têm relação linear,
 - Devem ter os outliers tratados

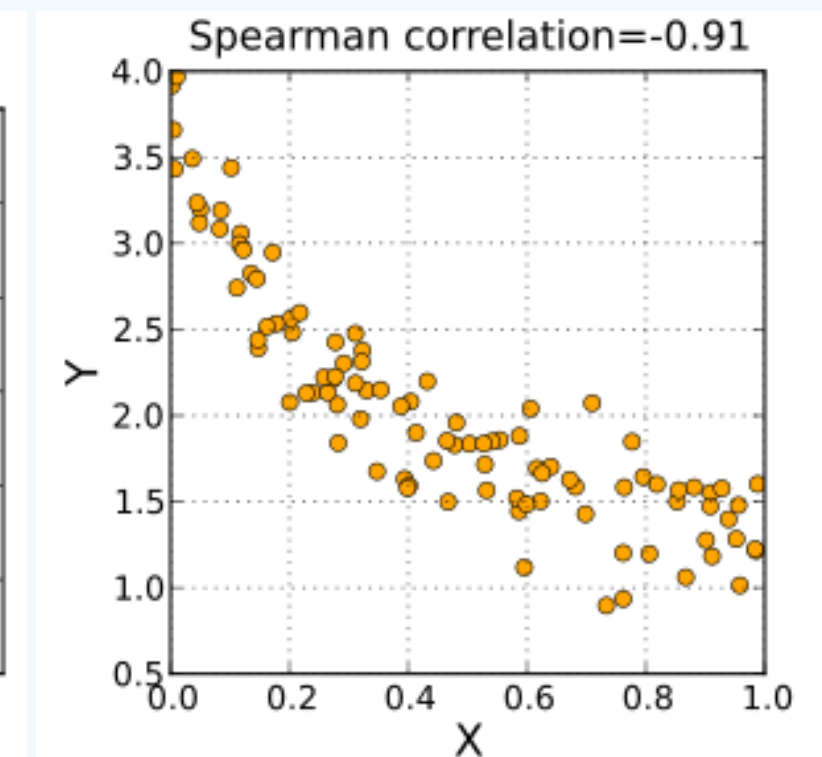
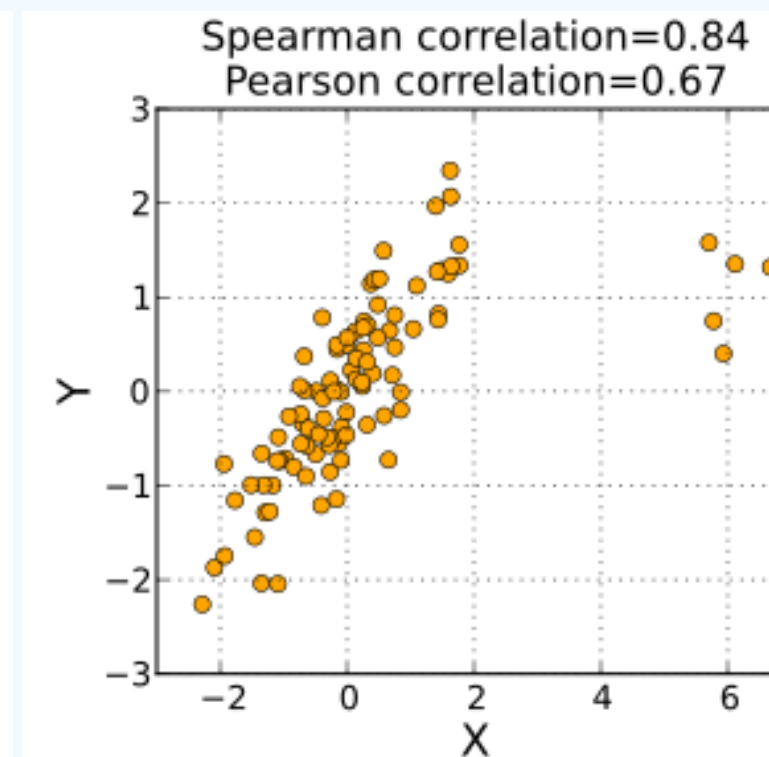
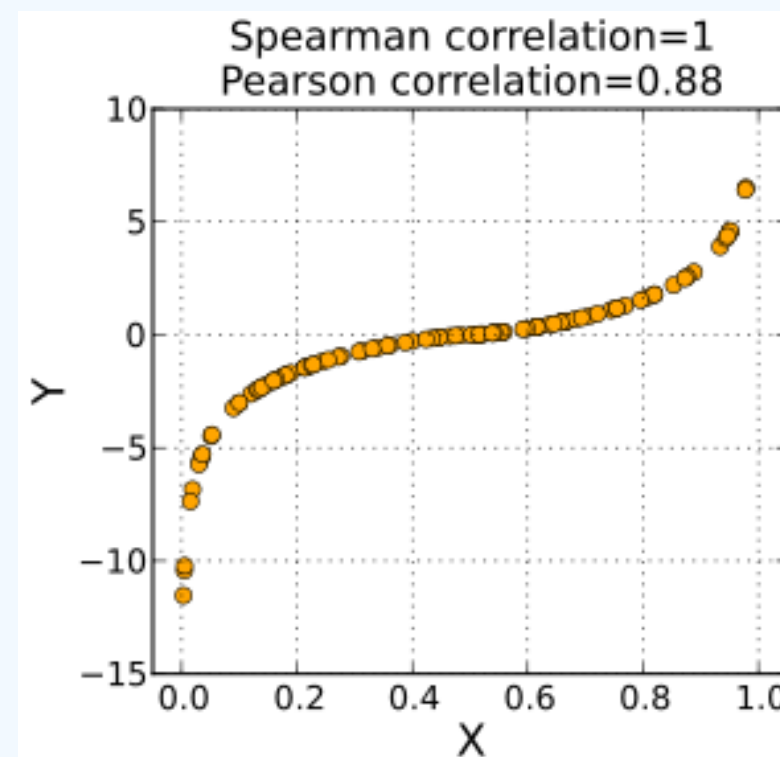


SELEÇÃO DE VARIÁVEIS

NÃO SUPERVISIONADO: CORRELAÇÕES

Spearman's e Kendall's

- Medem a relação monotônica entre variáveis;
- Em outras palavras, avaliam a tendência de duas variáveis “moverem na mesma direção”, mas não necessariamente de forma constante;
- Também retornam valores entre -1 e +1;
- São não-paramétricas, ou seja, não exigem comportamento normal das variáveis;
- Não demandam que os dados sejam contínuos (podem ser ordinais);
- Spearman's é matematicamente mais simples.



Fonte: https://en.wikipedia.org/wiki/Spearman%27s_rank_correlation_coefficient

SELEÇÃO DE VARIÁVEIS

NÃO SUPERVISIONADO: CORRELAÇÕES

pandas.DataFrame.rolling

pandas.DataFrame.expanding

pandas.DataFrame.ewm

pandas.DataFrame.abs

pandas.DataFrame.all

pandas.DataFrame.any

pandas.DataFrame.clip

pandas.DataFrame.corr

pandas.DataFrame.corrwith

pandas.DataFrame.count

pandas.DataFrame.cov

pandas.DataFrame.cummax

pandas.DataFrame.cummin

pandas.DataFrame.cumprod

pandas.DataFrame.cumsum

pandas.DataFrame.describe

pandas.DataFrame.diff

pandas.DataFrame.eval

pandas.DataFrame.kurt

pandas.DataFrame.kurtosis

pandas.DataFrame.mad

pandas.DataFrame.max

pandas.DataFrame.mean

pandas.DataFrame.median

pandas.DataFrame.min

pandas.DataFrame.mode

pandas.DataFrame.pct_change

pandas.DataFrame.corr — panda

pandas.pydata.org/docs/reference/api/pandas.DataFrame.corr.html

Getting started User Guide **API reference** Development Release notes

1.4.3

pandas.DataFrame.corr

`DataFrame.corr(method='pearson', min_periods=1)` [\[source\]](#)

Compute pairwise correlation of columns, excluding NA/null values.

Parameters: **method** : {'pearson', 'kendall', 'spearman'} or callable

Method of correlation:

- pearson : standard correlation coefficient
- kendall : Kendall Tau correlation coefficient
- spearman : Spearman rank correlation
- callable**: callable with input two 1d ndarrays

and returning a float. Note that the returned matrix from corr will have 1 along the diagonals and will be symmetric regardless of the callable's behavior.

min_periods : int, optional

Minimum number of observations required per pair of columns to have a valid result. Currently only available for Pearson and Spearman correlation.

Returns: **DataFrame**

Correlation matrix.

See also

DataFrame.corrwith

Compute pairwise correlation with another DataFrame or Series.

Series.corr

Compute the correlation between two Series.

Examples

SELEÇÃO DE VARIÁVEIS

NÃO SUPERVISIONADO: VARIÂNCIA

- Busca remover todas as variáveis cuja variância não atinge um valor mínimo;
- Se baseia no conceito de que algoritmos de *machine learning* necessitam de certa variância nos dados para efetivamente aprenderem (casos

de baixa variância costumam gerar o que chamamos de *underfit*)

SELEÇÃO DE VARIÁVEIS

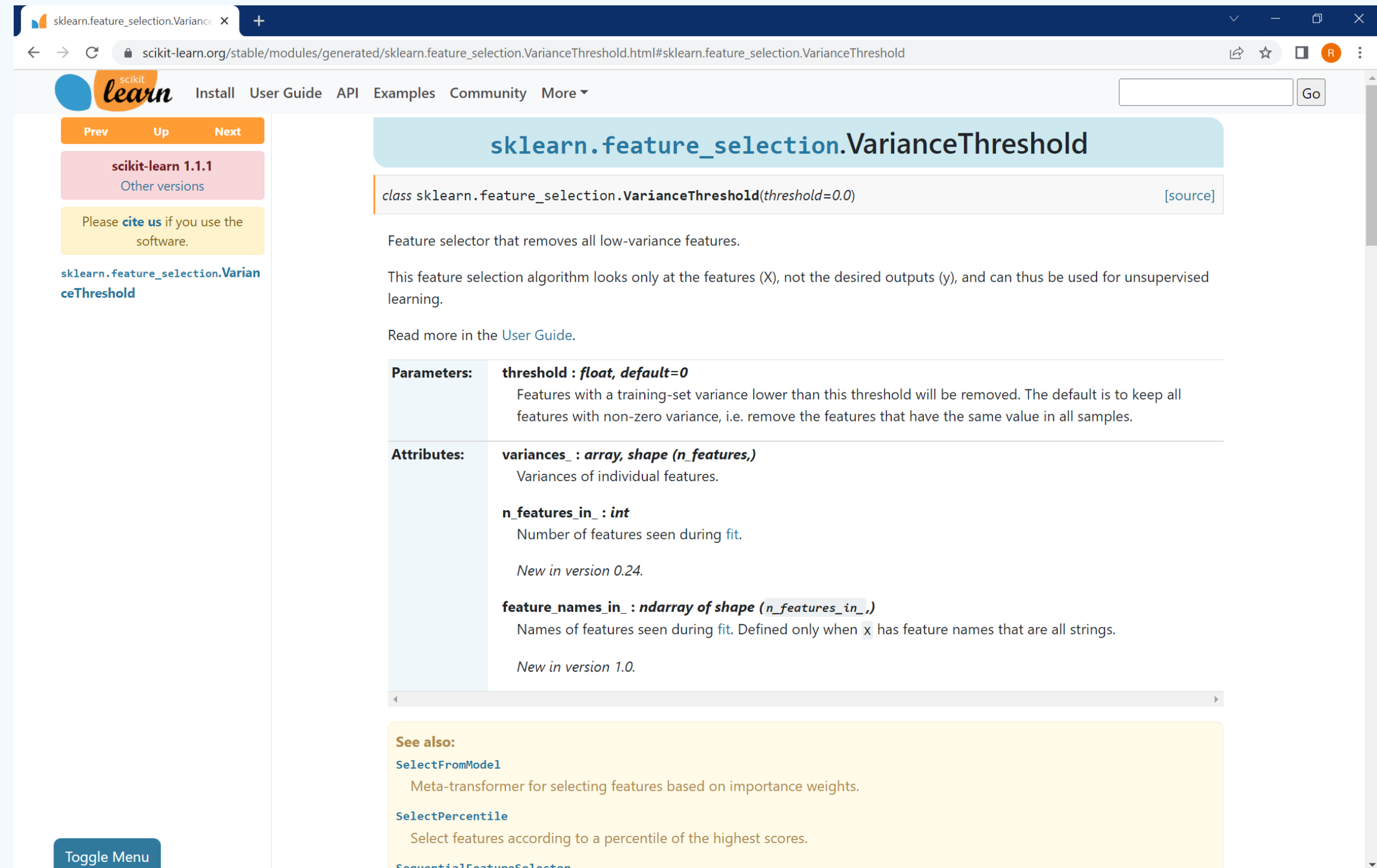
NÃO SUPERVISIONADO: VARIÂNCIA

- Para variáveis booleanas (variáveis aleatórias de Bernoulli), quando queremos uma variância de 0,8, devemos calcular o *threshold* da seguinte maneira:

$$\text{Var}[X] = p(1 - p)$$

$$\text{Var}[X] = 0,8(1 - 0,8)$$

$$\text{Var}[X] = 0,8(0,2) = 0,16$$



The screenshot shows the documentation for `sklearn.feature_selection.VarianceThreshold` on the scikit-learn website. The page includes the class name, a brief description, a detailed explanation of the algorithm, and a table of parameters and attributes.

sklearn.feature_selection.VarianceThreshold

```
class sklearn.feature_selection.VarianceThreshold(threshold=0.0)
```

Feature selector that removes all low-variance features.

This feature selection algorithm looks only at the features (X), not the desired outputs (y), and can thus be used for unsupervised learning.

Read more in the [User Guide](#).

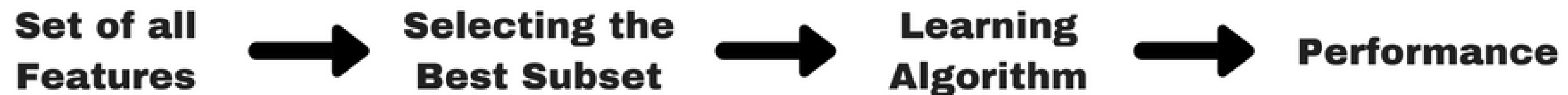
Parameters:	threshold : float, default=0 Features with a training-set variance lower than this threshold will be removed. The default is to keep all features with non-zero variance, i.e. remove the features that have the same value in all samples.
Attributes:	variances_ : array, shape (n_features,) Variances of individual features.
	n_features_in_ : int Number of features seen during <code>fit</code> . <i>New in version 0.24.</i>
	feature_names_in_ : ndarray of shape (n_features_in_,) Names of features seen during <code>fit</code> . Defined only when <code>x</code> has feature names that are all strings. <i>New in version 1.0.</i>

See also:

- [SelectFromModel](#)
Meta-transformer for selecting features based on importance weights.
- [SelectPercentile](#)
Select features according to a percentile of the highest scores.
- [SequentialFeatureSelector](#)

SELEÇÃO DE VARIÁVEIS

SUPERVISIONADO: FILTROS



Fonte: <https://www.analyticsvidhya.com/blog/2016/12/introduction-to-feature-selection-methods-with-an-example-or-how-to-select-the-right-variables/>

- As variáveis são selecionadas a partir do resultado de testes estatísticos que as relacionam com a variável resposta;
- Pearson's, Spearman's e Kendall's podem ser utilizadas, sendo que sempre uma das variáveis de entrada será a variável resposta;
- Outras possibilidades existem, entre elas:
 - ANOVA,
 - Chi Quadrado,
 - Permutation Importance e
 - Mutual Information

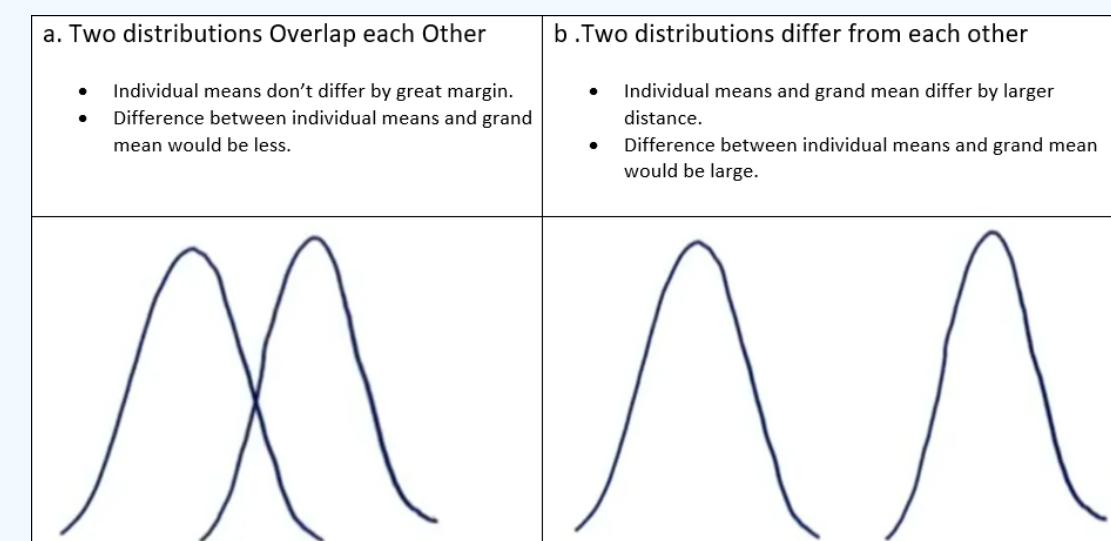
SELEÇÃO DE VARIÁVEIS

SUPERVISIONADO: FILTROS

- É uma análise de variância entre a variável resposta e a variável explicativa;
- Caso a variância entre o par observado for igual/muito próxima, conclui-se que a variável explicativa não tem impacto na resposta;

ANOVA

- Responde à pergunta: “a variável X impacta a variável Y?”, se H_0 for rejeitado, a resposta é sim.



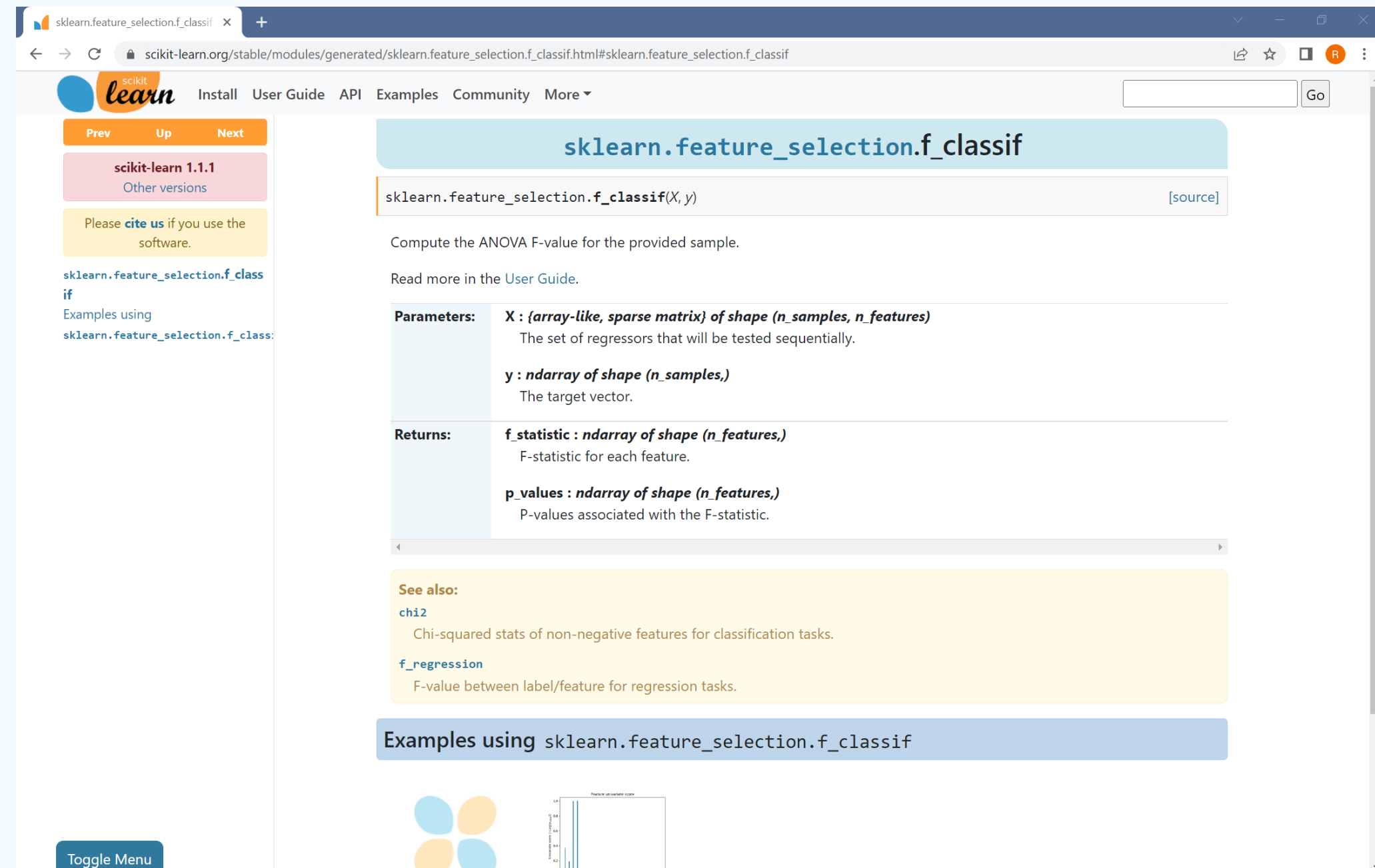
H_0 : Means of all groups are equal.

H_1 : At least one mean of the groups are different.

SELEÇÃO DE VARIÁVEIS

SUPERVISIONADO: FILTROS

ANOVA



The screenshot displays the official documentation for `sklearn.feature_selection.f_classif` on the scikit-learn website. The page is titled "sklearn.feature_selection.f_classif" and provides a detailed description of the function. The main content area includes the function signature `sklearn.feature_selection.f_classif(X, y)` and a brief description: "Compute the ANOVA F-value for the provided sample." Below this, there is a section for "Parameters" with two entries: `X` (array-like, sparse matrix of shape $(n_samples, n_features)$) and `y` (ndarray of shape $(n_samples,)$). The "Returns" section lists two outputs: `f_statistic` (ndarray of shape $(n_features,)$) and `p_values` (ndarray of shape $(n_features,)$). A "See also" section recommends `chi2` and `f_regression`. The page also features a "Toggle Menu" button at the bottom left and a small bar chart visualization at the bottom right.

sklearn.feature_selection.f_classif

sklearn.feature_selection.f_classif(X, y) [source]

Compute the ANOVA F-value for the provided sample.

Read more in the [User Guide](#).

Parameters:

- X** : {array-like, sparse matrix} of shape $(n_samples, n_features)$
The set of regressors that will be tested sequentially.
- y** : ndarray of shape $(n_samples,)$
The target vector.

Returns:

- f_statistic** : ndarray of shape $(n_features,)$
F-statistic for each feature.
- p_values** : ndarray of shape $(n_features,)$
P-values associated with the F-statistic.

See also:

- [chi2](#)
Chi-squared stats of non-negative features for classification tasks.
- [f_regression](#)
F-value between label/feature for regression tasks.

Examples using sklearn.feature_selection.f_classif

Toggle Menu

SELEÇÃO DE VARIÁVEIS

SUPERVISIONADO: FILTROS

- Baseia-se na análise de independência entre dois eventos;
- Buscamos selecionar as variáveis mais dependentes em relação à resposta;
- Logo, buscamos as variáveis que tem maior valor

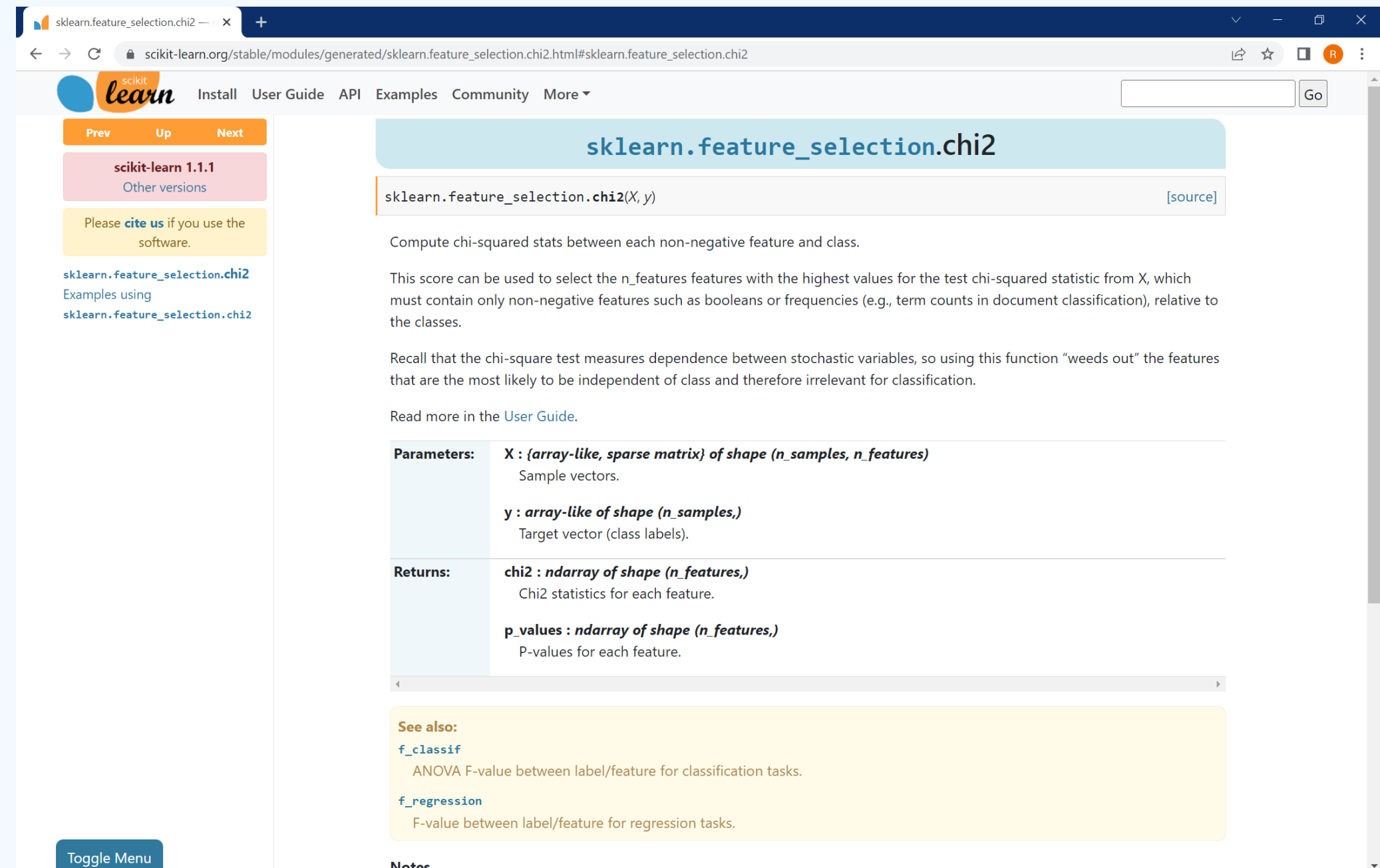
CHI QUADRADO

para o teste.

SELEÇÃO DE VARIÁVEIS

SUPERVISIONADO: FILTROS

CHI QUADRADO



The screenshot shows the documentation for `sklearn.feature_selection.chi2` on the scikit-learn website. The page title is `sklearn.feature_selection.chi2`. The URL is `scikit-learn.org/stable/modules/generated/sklearn.feature_selection.chi2.html#sklearn.feature_selection.chi2`. The page content includes a description of the function, its parameters, and its returns.

sklearn.feature_selection.chi2

`sklearn.feature_selection.chi2(X, y)` [\[source\]](#)

Compute chi-squared stats between each non-negative feature and class.

This score can be used to select the `n_features` features with the highest values for the test chi-squared statistic from `X`, which must contain only non-negative features such as booleans or frequencies (e.g., term counts in document classification), relative to the classes.

Recall that the chi-square test measures dependence between stochastic variables, so using this function “weeds out” the features that are the most likely to be independent of class and therefore irrelevant for classification.

Read more in the [User Guide](#).

Parameters:

- X** : *{array-like, sparse matrix} of shape (n_samples, n_features)*
Sample vectors.
- y** : *array-like of shape (n_samples,)*
Target vector (class labels).

Returns:

- chi2** : *ndarray of shape (n_features,)*
Chi2 statistics for each feature.
- p_values** : *ndarray of shape (n_features,)*
P-values for each feature.

See also:

- [f_classif](#)
ANOVA F-value between label/feature for classification tasks.
- [f_regression](#)
F-value between label/feature for regression tasks.

Notes

SELEÇÃO DE VARIÁVEIS

SUPERVISIONADO: FILTROS

- Baseia-se em quebrar a relação entre uma variável explicativa e a variável resposta a partir de um modelo;
- O impacto gerado expressa o quanto a relação é importante para as predições;

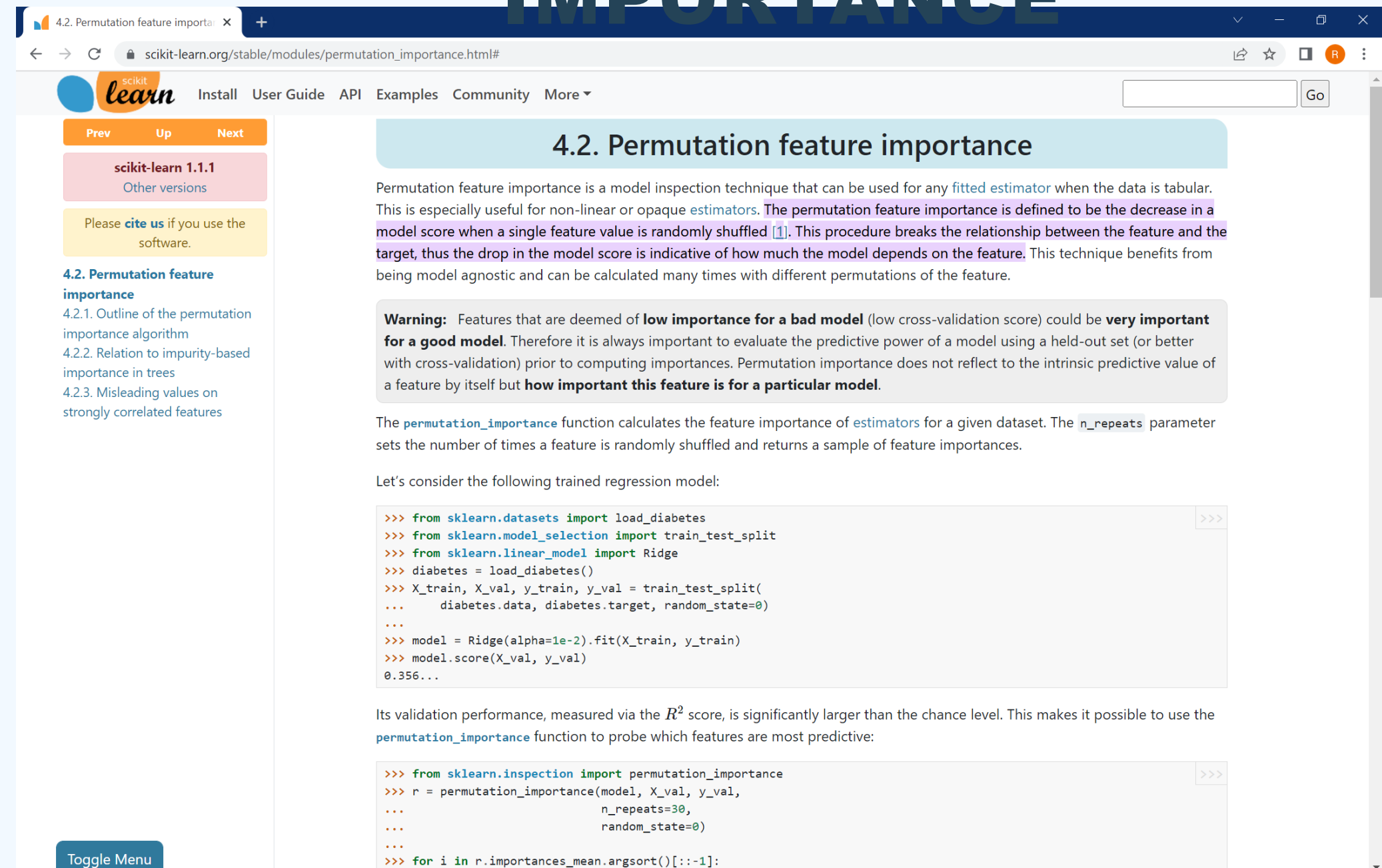
PERMUTATION IMPORTANCE

- A variável sendo analisada é aleatoriamente embaralhada para que a relação se “quebre” e a queda no escore do modelo é o impacto

SELEÇÃO DE VARIÁVEIS

SUPERVISIONADO: FILTROS

PERMUTATION IMPORTANCE



4.2. Permutation feature importance

Permutation feature importance is a model inspection technique that can be used for any [fitted estimator](#) when the data is tabular. This is especially useful for non-linear or opaque [estimators](#). The permutation feature importance is defined to be the decrease in a model score when a single feature value is randomly shuffled [1]. This procedure breaks the relationship between the feature and the target, thus the drop in the model score is indicative of how much the model depends on the feature. This technique benefits from being model agnostic and can be calculated many times with different permutations of the feature.

Warning: Features that are deemed of **low importance for a bad model** (low cross-validation score) could be **very important for a good model**. Therefore it is always important to evaluate the predictive power of a model using a held-out set (or better with cross-validation) prior to computing importances. Permutation importance does not reflect to the intrinsic predictive value of a feature by itself but **how important this feature is for a particular model**.

The `permutation_importance` function calculates the feature importance of [estimators](#) for a given dataset. The `n_repeats` parameter sets the number of times a feature is randomly shuffled and returns a sample of feature importances.

Let's consider the following trained regression model:

```
>>> from sklearn.datasets import load_diabetes
>>> from sklearn.model_selection import train_test_split
>>> from sklearn.linear_model import Ridge
>>> diabetes = load_diabetes()
>>> X_train, X_val, y_train, y_val = train_test_split(
...     diabetes.data, diabetes.target, random_state=0)
...
>>> model = Ridge(alpha=1e-2).fit(X_train, y_train)
>>> model.score(X_val, y_val)
0.356...
```

Its validation performance, measured via the R^2 score, is significantly larger than the chance level. This makes it possible to use the `permutation_importance` function to probe which features are most predictive:

```
>>> from sklearn.inspection import permutation_importance
>>> r = permutation_importance(model, X_val, y_val,
...                           n_repeats=30,
...                           random_state=0)
...
>>> for i in r.importances_mean.argsort()[::-1]:
...     print(f"Feature {i} importance: {r.importances_mean[i]:.3f}")
```

SELEÇÃO DE VARIÁVEIS

SUPERVISIONADO: FILTROS

- Baseia-se na dependência mútua entre duas variáveis;
- Quantifica a “informação” de uma das variáveis que é obtida ao observar-se uma segunda;
- Associa-se com o conceito de entropia, que, nesse

MUTUAL INFORMATION

sentido, compreende a quantidade de informação sobre a variável resposta que está presente em uma dada variável explicativa



Two bits of entropy: In the case of two fair coin tosses, the information entropy in bits is the base-2 logarithm of the number of possible outcomes; with two coins there are four possible outcomes, and two bits of entropy. Generally, information **entropy is the average amount of information conveyed by an event**, when considering all possible outcomes.

SELEÇÃO DE VARIÁVEIS

SUPERVISIONADO: FILTROS

MUTUAL INFORMATION

sklearn.feature_selection.mutual_

scikit-learn.org/stable/modules/generated/sklearn.feature_selection.mutual_info_classif.html#sklearn.feature_selection.mutual_info_classif

scikit-learn

Install User Guide API Examples Community More

Prev Up Next

scikit-learn 1.1.1
Other versions

Please cite us if you use the software.

sklearn.feature_selection.mutual_info_classif

Toggle Menu

sklearn.feature_selection.mutual_info_classif

```
sklearn.feature_selection.mutual_info_classif(X, y, *, discrete_features='auto', n_neighbors=3, copy=True, random_state=None)
```

[source]

Estimate mutual information for a discrete target variable.

Mutual information (MI) [1] between two random variables is a non-negative value, which measures the dependency between the variables. It is equal to zero if and only if two random variables are independent, and higher values mean higher dependency.

The function relies on nonparametric methods based on entropy estimation from k-nearest neighbors distances as described in [2] and [3]. Both methods are based on the idea originally proposed in [4].

It can be used for univariate features selection, read more in the [User Guide](#).

Parameters:

X : array-like or sparse matrix, shape (n_samples, n_features)
Feature matrix.

y : array-like of shape (n_samples,)
Target vector.

discrete_features : {'auto', bool, array-like}, default='auto'
If bool, then determines whether to consider all features discrete or continuous. If array, then it should be either a boolean mask with shape (n_features,) or array with indices of discrete features. If 'auto', it is assigned to False for dense X and to True for sparse X.

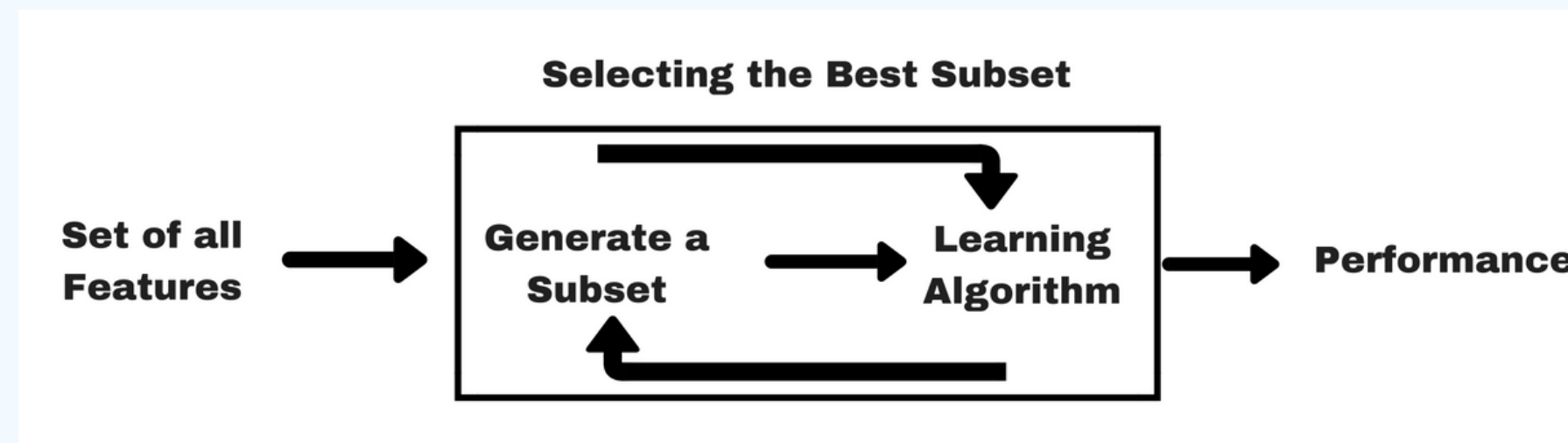
n_neighbors : int, default=3
Number of neighbors to use for MI estimation for continuous variables, see [2] and [3]. Higher values reduce variance of the estimation, but could introduce a bias.

copy : bool, default=True
Whether to make a copy of the given data. If set to False, the initial data will be overwritten.

random_state : int, RandomState instance or None, default=None
Determines random number generation for adding small noise to continuous variables in order to remove repeated values. Pass an int for reproducible results across multiple function calls. See [Glossary](#).

SELEÇÃO DE VARIÁVEIS

SUPERVISIONADO: *WRAPPERS*

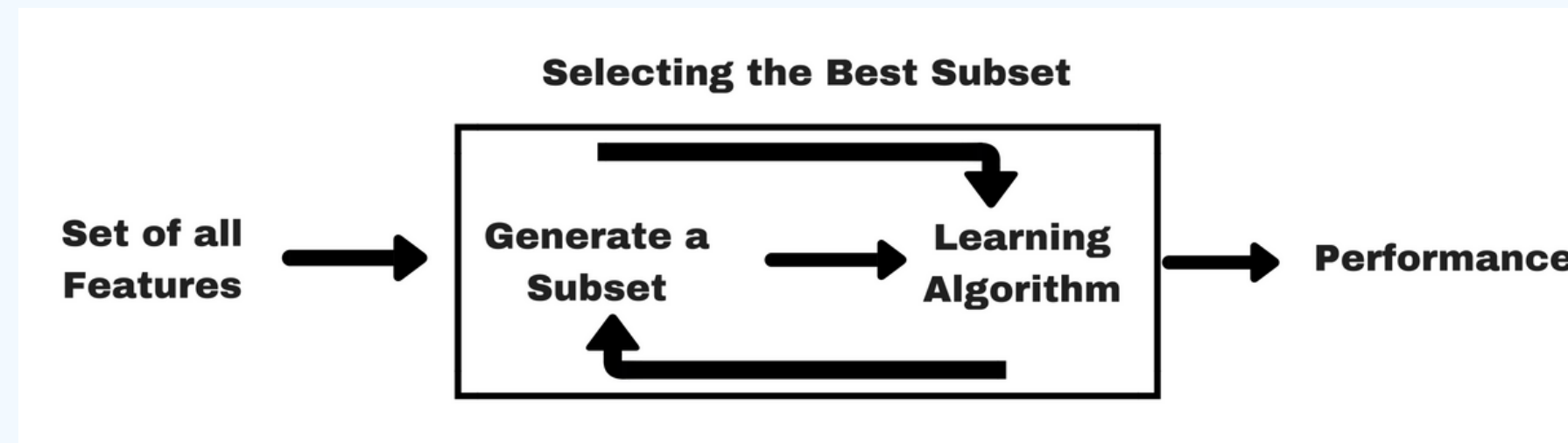


Fonte: <https://www.analyticsvidhya.com/blog/2016/12/introduction-to-feature-selection-methods-with-an-example-or-how-to-select-the-right-variables/>

- Uma amostra das variáveis explicativas é selecionada e um modelo é treinado com elas;
- Baseado no resultado do modelo anterior, adiciona-se ou retira-se variáveis dessa amostra;
- Costumam ser computacionalmente mais robustos.

SELEÇÃO DE VARIÁVEIS

SUPERVISIONADO: *WRAPPERS*



Fonte: <https://www.analyticsvidhya.com/blog/2016/12/introduction-to-feature-selection-methods-with-an-example-or-how-to-select-the-right-variables/>

- Como exemplos, temos:
 - Forward Selection,
 - Backward Elimination e
 - Recursive Feature Elimination

SELEÇÃO DE VARIÁVEIS

SUPERVISIONADO: *WRAPPERS*

- Baseia-se em adicionar variáveis até que a performance do modelo se estabilize;
- O modelo começa com uma única variável (a melhor) e vai adicionando novas em cada rodada;
- O processo se inicia por avaliar cada variável individualmente para ordenar entre a melhor e a

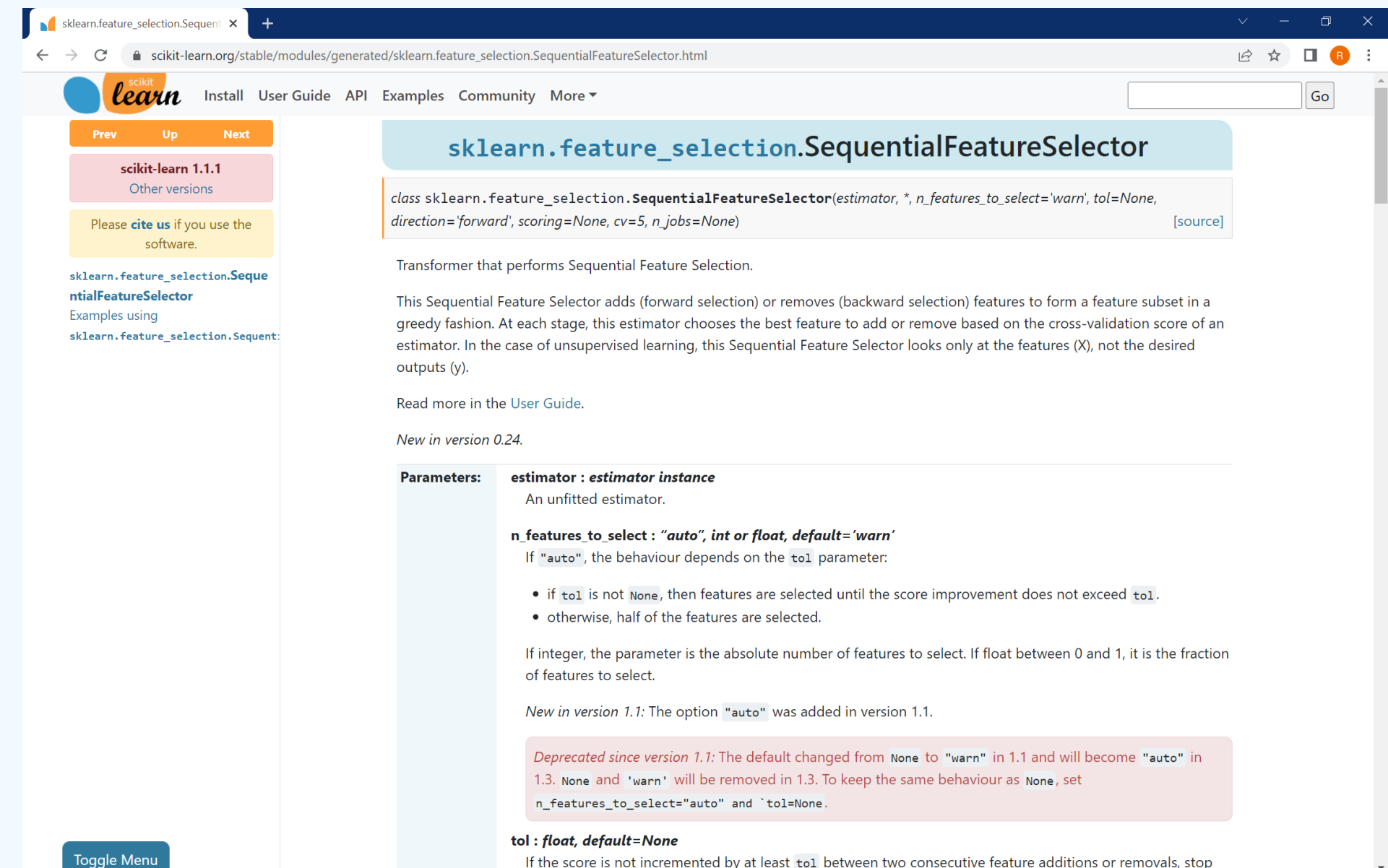
FORWARD SELECTION

- pior individualmente;
- A cada rodada o algoritmo avalia todas as combinações possíveis entre a variável selecionada e a próxima que culmine no melhor resultado para o modelo.

SELEÇÃO DE VARIÁVEIS

SUPERVISIONADO: *WRAPPERS*

FORWARD SELECTION



The screenshot shows the scikit-learn documentation page for `sklearn.feature_selection.SequentialFeatureSelector`. The page includes a sidebar with navigation links (Prev, Up, Next) and a main content area with the class name, signature, description, and parameters.

sklearn.feature_selection.SequentialFeatureSelector

```
class sklearn.feature_selection.SequentialFeatureSelector(estimator, *, n_features_to_select='warn', tol=None, direction='forward', scoring=None, cv=5, n_jobs=None)
```

Transformer that performs Sequential Feature Selection.

This Sequential Feature Selector adds (forward selection) or removes (backward selection) features to form a feature subset in a greedy fashion. At each stage, this estimator chooses the best feature to add or remove based on the cross-validation score of an estimator. In the case of unsupervised learning, this Sequential Feature Selector looks only at the features (X), not the desired outputs (y).

Read more in the [User Guide](#).

New in version 0.24.

Parameters:

- estimator : estimator instance**
An unfitted estimator.
- n_features_to_select : "auto", int or float, default='warn'**
If "auto", the behaviour depends on the `tol` parameter:
 - if `tol` is not `None`, then features are selected until the score improvement does not exceed `tol`.
 - otherwise, half of the features are selected.If integer, the parameter is the absolute number of features to select. If float between 0 and 1, it is the fraction of features to select.

New in version 1.1: The option "auto" was added in version 1.1.

Deprecated since version 1.1: The default changed from `None` to "warn" in 1.1 and will become "auto" in 1.3. `None` and 'warn' will be removed in 1.3. To keep the same behaviour as `None`, set `n_features_to_select="auto"` and `tol=None`.

- tol : float, default=None**
If the score is not incremented by at least `tol` between two consecutive feature additions or removals, stop

SELEÇÃO DE VARIÁVEIS

SUPERVISIONADO: *WRAPPERS*

- Segue o mesmo racional do Forward Selection, mas de forma invertida;
- Portanto, começa com todas as variáveis e, a cada iteração, elimina a variável que tem a menor importância no modelo;

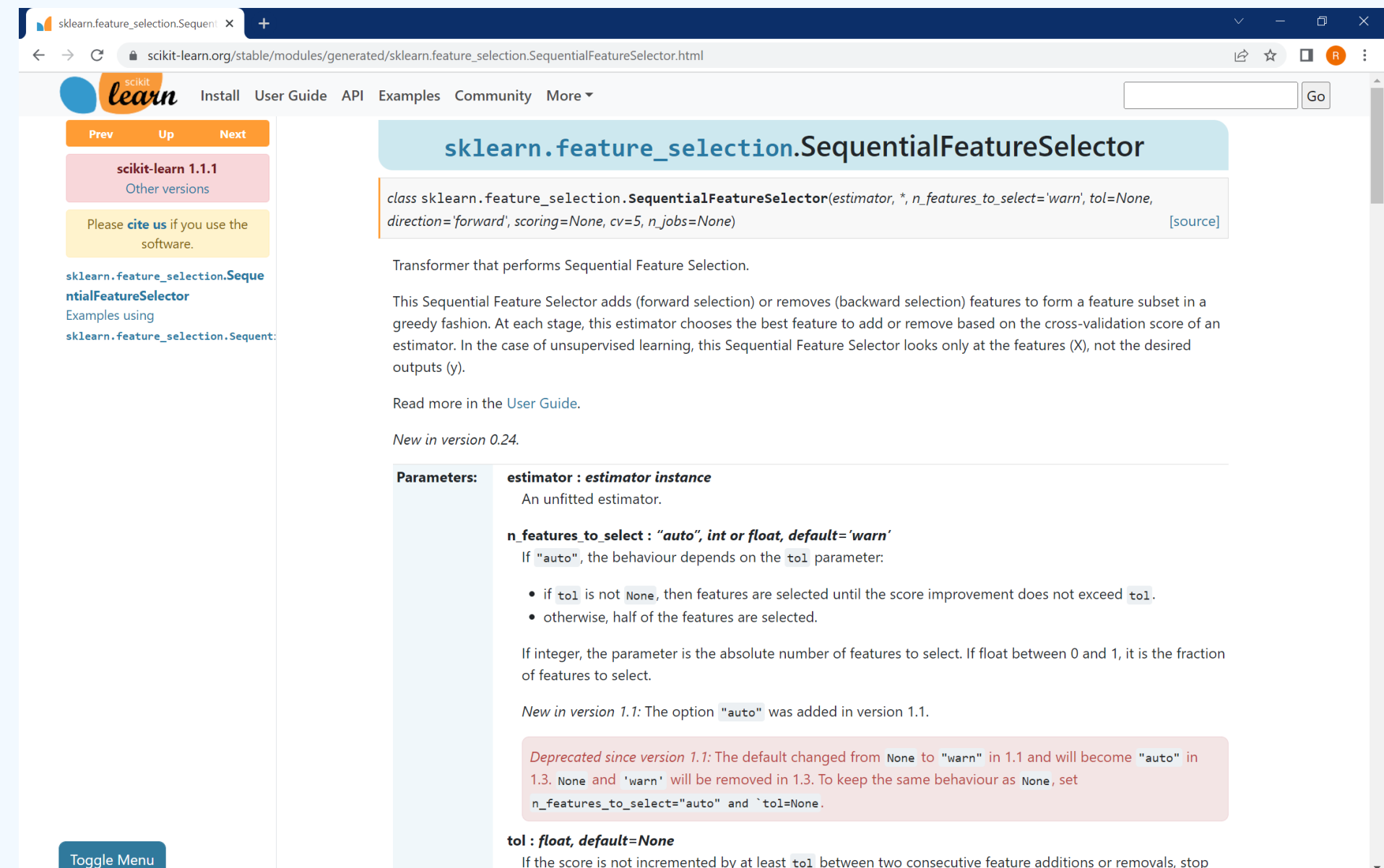
BACKWARD ELIMINATION

- Este processo acontece até que nenhuma melhora nas métricas seja observada.

SELEÇÃO DE VARIÁVEIS

SUPERVISIONADO: *WRAPPERS*

BACKWARD ELIMINATION



The screenshot shows the sklearn documentation page for `SequentialFeatureSelector`. The page is titled "sklearn.feature_selection.SequentialFeatureSelector" and includes a navigation bar with links for "Install", "User Guide", "API", "Examples", "Community", and "More". The main content area describes the class as a transformer that performs Sequential Feature Selection. It explains that the selector adds or removes features in a greedy fashion based on cross-validation scores. The page also lists parameters: `estimator` (an unfitted estimator), `n_features_to_select` (default "warn", can be "auto", int, or float), and `tol` (float, default None). A deprecation notice states that the default for `n_features_to_select` changed from "None" to "warn" in version 1.1 and will become "auto" in version 1.3. The page also includes a "Toggle Menu" button at the bottom left.

sklearn.feature_selection.SequentialFeatureSelector

```
class sklearn.feature_selection.SequentialFeatureSelector(estimator, *, n_features_to_select='warn', tol=None, direction='forward', scoring=None, cv=5, n_jobs=None)
```

Transformer that performs Sequential Feature Selection.

This Sequential Feature Selector adds (forward selection) or removes (backward selection) features to form a feature subset in a greedy fashion. At each stage, this estimator chooses the best feature to add or remove based on the cross-validation score of an estimator. In the case of unsupervised learning, this Sequential Feature Selector looks only at the features (X), not the desired outputs (y).

Read more in the [User Guide](#).

New in version 0.24.

Parameters:

- estimator : estimator instance**
An unfitted estimator.
- n_features_to_select : "auto", int or float, default='warn'**
If "auto", the behaviour depends on the `tol` parameter:
 - if `tol` is not `None`, then features are selected until the score improvement does not exceed `tol`.
 - otherwise, half of the features are selected.If integer, the parameter is the absolute number of features to select. If float between 0 and 1, it is the fraction of features to select.
- tol : float, default=None**
If the score is not incremented by at least `tol` between two consecutive feature additions or removals, stop

Deprecated since version 1.1: The default changed from `None` to `"warn"` in 1.1 and will become `"auto"` in 1.3. `None` and `'warn'` will be removed in 1.3. To keep the same behaviour as `None`, set `n_features_to_select="auto"` and `tol=None`.

SELEÇÃO DE VARIÁVEIS

RFE

SUPERVISIONADO: *WRAPPERS*

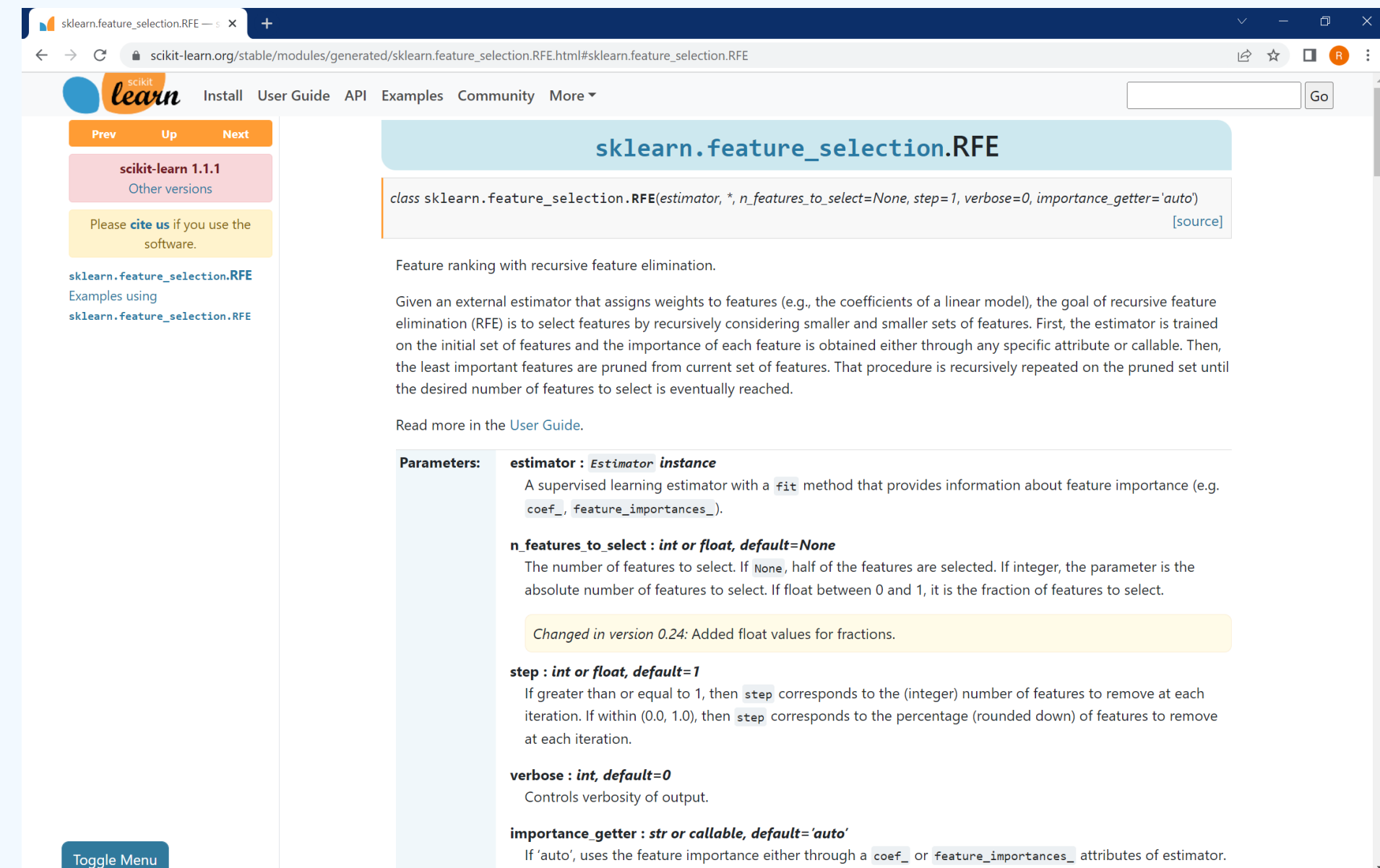
- Cria modelos em cada iteração para identificar as piores variáveis, ou seja, as que podem/devem ser eliminadas;
- Em cada iteração, retira a variável que teve a pior performance;

- Em seguida, o próximo modelo é construído usando as variáveis do anterior que foram mantidas.

SELEÇÃO DE VARIÁVEIS

SUPERVISIONADO: *WRAPPERS*

RFE



The screenshot shows the official documentation for `sklearn.feature_selection.RFE` on the scikit-learn website. The page is titled "sklearn.feature_selection.RFE" and includes a navigation bar with links for "Install", "User Guide", "API", "Examples", "Community", and "More". The main content area is divided into two columns. The left column contains a sidebar with links for "Prev", "Up", "Next", "scikit-learn 1.1.1", "Other versions", and a "cite us" notice. The right column contains the main documentation text. The text describes RFE as a feature ranking method with recursive feature elimination. It explains that RFE selects features by recursively considering smaller and smaller sets of features, starting from the initial set and pruning the least important features until the desired number of features is reached. The documentation also includes a list of parameters: `estimator` (a supervised learning estimator with a `fit` method), `n_features_to_select` (the number of features to select, defaulting to `None`), `step` (the number of features to remove at each iteration, defaulting to 1), `verbose` (controls verbosity of output, defaulting to 0), and `importance_getter` (a string or callable, defaulting to 'auto').

sklearn.feature_selection.RFE

```
class sklearn.feature_selection.RFE(estimator, *, n_features_to_select=None, step=1, verbose=0, importance_getter='auto')
```

Feature ranking with recursive feature elimination.

Given an external estimator that assigns weights to features (e.g., the coefficients of a linear model), the goal of recursive feature elimination (RFE) is to select features by recursively considering smaller and smaller sets of features. First, the estimator is trained on the initial set of features and the importance of each feature is obtained either through any specific attribute or callable. Then, the least important features are pruned from current set of features. That procedure is recursively repeated on the pruned set until the desired number of features to select is eventually reached.

Read more in the [User Guide](#).

Parameters:

- estimator : Estimator instance**
A supervised learning estimator with a `fit` method that provides information about feature importance (e.g. `coef_`, `feature_importances_`).
- n_features_to_select : int or float, default=None**
The number of features to select. If `None`, half of the features are selected. If integer, the parameter is the absolute number of features to select. If float between 0 and 1, it is the fraction of features to select.
Changed in version 0.24: Added float values for fractions.
- step : int or float, default=1**
If greater than or equal to 1, then `step` corresponds to the (integer) number of features to remove at each iteration. If within (0.0, 1.0), then `step` corresponds to the percentage (rounded down) of features to remove at each iteration.
- verbose : int, default=0**
Controls verbosity of output.
- importance_getter : str or callable, default='auto'**
If 'auto', uses the feature importance either through a `coef_` or `feature_importances_` attributes of estimator.

SELEÇÃO DE VARIÁVEIS

SUPERVISIONADO: INTRÍNSECOS

- São métodos já presentes internamente em alguns algoritmos de aprendizado de máquina;
- Ou seja, no próprio funcionamento do algoritmo há uma seleção/definição de importância para as variáveis disponíveis;
- Avaliaremos dois casos:
 - LASSO
 - Random Forest Importance

SELEÇÃO DE VARIÁVEIS

SUPERVISIONADO: INTRÍNSECOS

- Regressão LASSO é o mesmo que Regressão L1;
- Nesse tipo de regressão há a atribuição de uma penalidade no algoritmo;
- Essa penalidade é aplicada a cada parâmetro do modelo, o que diminuiu o grau de liberdade do

LASSO

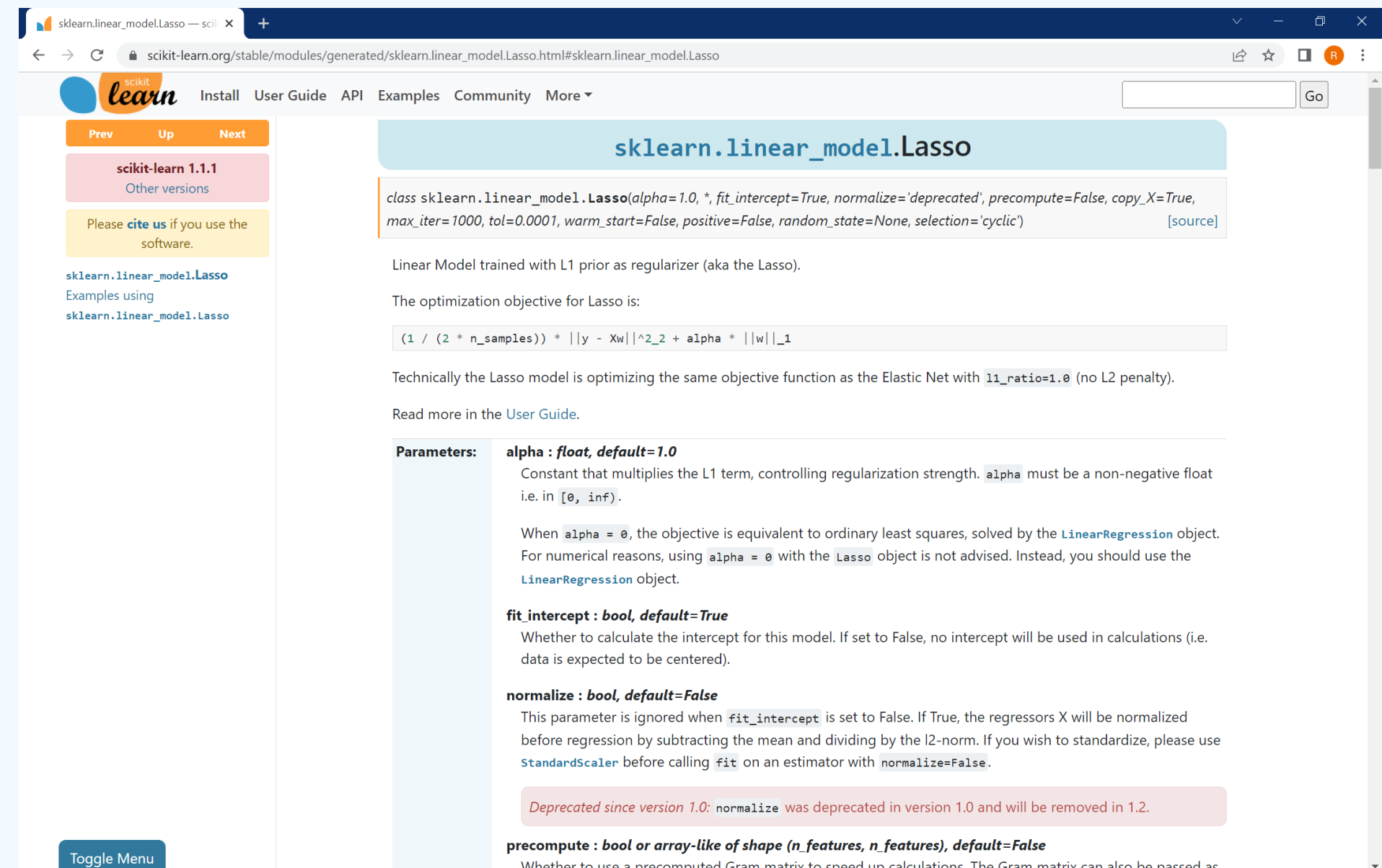
problema (evita *overfit*);

- Caso nesse processo algum dos coeficientes da regressão se torne 0, isso significa que a variável associada a ele não tem importância para o problema.

SELEÇÃO DE VARIÁVEIS

SUPERVISIONADO: INTRÍNSECOS

LASSO



The screenshot shows the official documentation for `sklearn.linear_model.Lasso` on the scikit-learn website. The page includes the following content:

- Navigation:** Links for 'Prev', 'Up', and 'Next' are at the top left. Below them are links for 'scikit-learn 1.1.1' and 'Other versions'. A note asks users to 'cite us' if they use the software. At the bottom left is a 'Toggle Menu' button.
- Header:** The title 'sklearn.linear_model.Lasso' is displayed in a light blue box.
- Code:** A code block shows the class signature: `class sklearn.linear_model.Lasso(alpha=1.0, *, fit_intercept=True, normalize='deprecated', precompute=False, copy_X=True, max_iter=1000, tol=0.0001, warm_start=False, positive=False, random_state=None, selection='cyclic')`. A '[source]' link is provided.
- Description:** A paragraph states: 'Linear Model trained with L1 prior as regularizer (aka the Lasso).'.
- Objective:** A paragraph states: 'The optimization objective for Lasso is:' followed by the mathematical formula:
$$\left(\frac{1}{2} * n_samples \right) * ||y - Xw||^2_2 + \alpha * ||w||_1$$
- Technical Note:** A paragraph states: 'Technically the Lasso model is optimizing the same objective function as the Elastic Net with `l1_ratio=1.0` (no L2 penalty).'.
- User Guide:** A link 'Read more in the User Guide.' is provided.
- Parameters:** A section titled 'Parameters:' lists the following:
 - alpha : float, default=1.0**
Constant that multiplies the L1 term, controlling regularization strength. `alpha` must be a non-negative float i.e. in `[0, inf)`.
When `alpha = 0`, the objective is equivalent to ordinary least squares, solved by the `LinearRegression` object. For numerical reasons, using `alpha = 0` with the `Lasso` object is not advised. Instead, you should use the `LinearRegression` object.
 - fit_intercept : bool, default=True**
Whether to calculate the intercept for this model. If set to False, no intercept will be used in calculations (i.e. data is expected to be centered).
 - normalize : bool, default=False**
This parameter is ignored when `fit_intercept` is set to False. If True, the regressors X will be normalized before regression by subtracting the mean and dividing by the l2-norm. If you wish to standardize, please use `StandardScaler` before calling `fit` on an estimator with `normalize=False`.
- Deprecation Notice:** A red box states: 'Deprecated since version 1.0: `normalize` was deprecated in version 1.0 and will be removed in 1.2.'
- precompute :** A paragraph states: 'precompute : bool or array-like of shape (n_features, n_features), default=False
Whether to use a precomputed Gram matrix to speed up calculations. The Gram matrix can also be passed as

SELEÇÃO DE VARIÁVEIS

SUPERVISIONADO: INTRÍNSECOS

LASSO

sklearn.feature_selection.SelectFromModel

scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectFromModel.html#sklearn.feature_selection.SelectFromModel

scikit-learn

Install User Guide API Examples Community More

Prev Up Next

scikit-learn 1.1.1
Other versions

Please cite us if you use the software.

sklearn.feature_selection.SelectFromModel
Examples using
sklearn.feature_selection.SelectFromModel

Toggle Menu

sklearn.feature_selection.SelectFromModel

```
class sklearn.feature_selection.SelectFromModel(estimator, *, threshold=None, prefit=False, norm_order=1, max_features=None, importance_getter='auto')
```

[\[source\]](#)

Meta-transformer for selecting features based on importance weights.

New in version 0.17.

Read more in the [User Guide](#).

Parameters:

estimator : object
The base estimator from which the transformer is built. This can be both a fitted (if `prefit` is set to `True`) or a non-fitted estimator. The estimator should have a `feature_importances_` or `coef_` attribute after fitting. Otherwise, the `importance_getter` parameter should be used.

threshold : str or float, default=None
The threshold value to use for feature selection. Features whose importance is greater or equal are kept while the others are discarded. If "median" (resp. "mean"), then the `threshold` value is the median (resp. the mean) of the feature importances. A scaling factor (e.g., "1.25*mean") may also be used. If `None` and if the estimator has a parameter penalty set to `l1`, either explicitly or implicitly (e.g. Lasso), the threshold used is `1e-5`. Otherwise, "mean" is used by default.

prefit : bool, default=False
Whether a prefit model is expected to be passed into the constructor directly or not. If `True`, `estimator` must be a fitted estimator. If `False`, `estimator` is fitted and updated by calling `fit` and `partial_fit`, respectively.

norm_order : non-zero int, inf, -inf, default=1
Order of the norm used to filter the vectors of coefficients below `threshold` in the case where the `coef_` attribute of the estimator is of dimension 2.

max_features : int, callable, default=None
The maximum number of features to select.

- If an integer, then it specifies the maximum number of features to allow.
- If a callable, then it specifies how to calculate the maximum number of features allowed by using the out-

SELEÇÃO DE VARIÁVEIS

SUPERVISIONADO: INTRÍNSECOS

- É um método do tipo *ensemble* que constrói vários preditores do tipo árvore com baixa ou nenhuma correlação entre si;
- Por definição do algoritmo de árvore de decisão, a importância de cada variável é calculada para

RF IMPORTANCE

- permitir o crescimento da árvore;
- A importância da variável é obtida pelo grau de pureza da amostra em cada galho/folha subsequente.
- A pureza é medida em GINI ou Entropia.

SELEÇÃO DE VARIÁVEIS

SUPERVISIONADO: INTRÍNSECOS

RF IMPORTANCE

sklearn.ensemble.RandomForestClassifier

scikit-learn 1.1.1
Other versions

Please cite us if you use the software.

sklearn.ensemble.RandomForestClassifier

Examples using sklearn.ensemble.RandomForestClassifier

Toggle Menu

sklearn.ensemble.RandomForestClassifier

```
class sklearn.ensemble.RandomForestClassifier(n_estimators=100, *, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='sqrt', max_leaf_nodes=None, min_impurity_decrease=0.0, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None)
```

A random forest classifier.

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the `max_samples` parameter if `bootstrap=True` (default), otherwise the whole dataset is used to build each tree.

Read more in the User Guide.

Parameters:

n_estimators : int, default=100
The number of trees in the forest.

Changed in version 0.22: The default value of `n_estimators` changed from 10 to 100 in 0.22.

criterion : {"gini", "entropy", "log_loss"}, default="gini"
The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "log_loss" and "entropy" both for the Shannon information gain, see [Mathematical formulation](#). Note: This parameter is tree-specific.

max_depth : int, default=None
The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than `min_samples_split` samples.

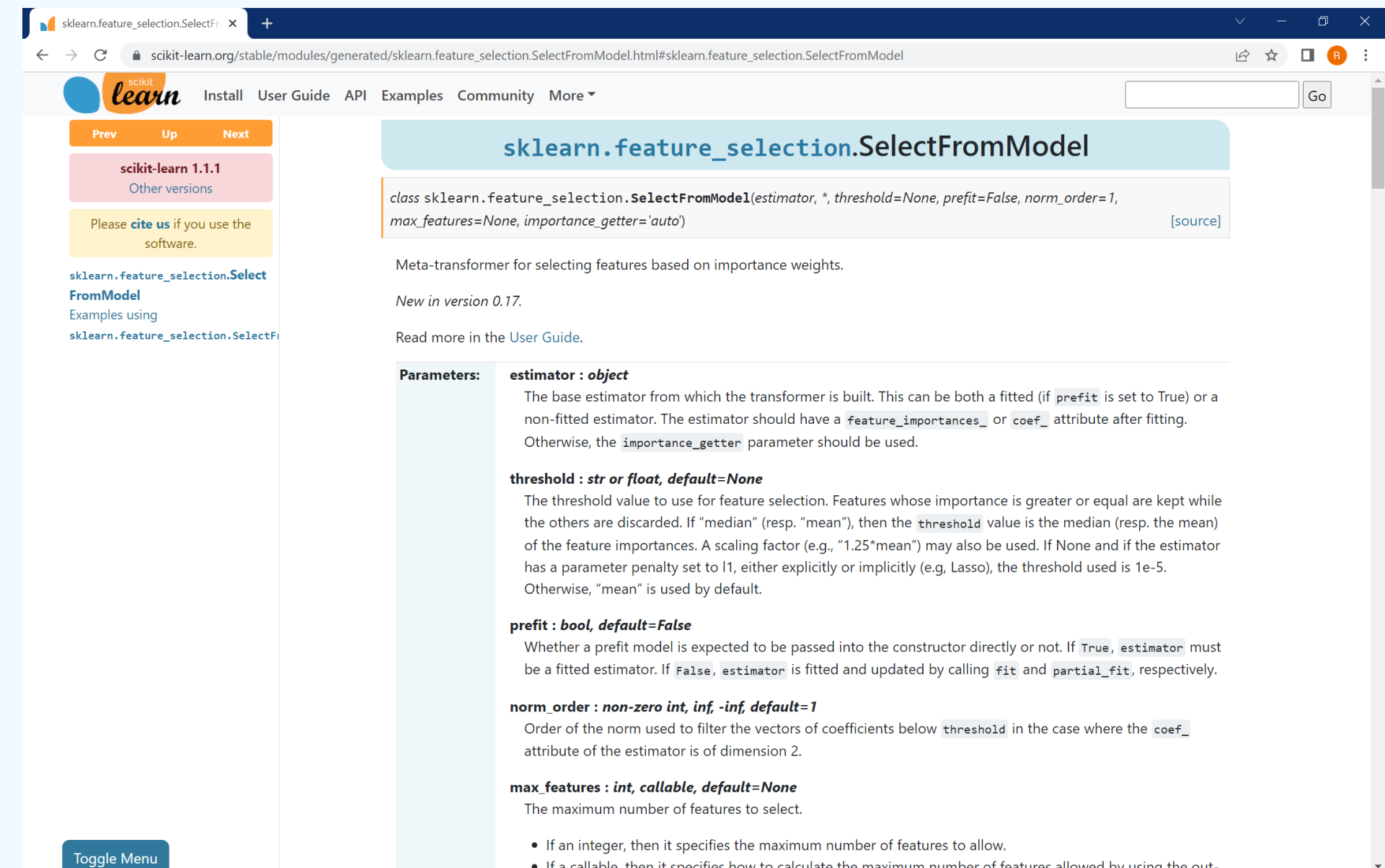
min_samples_split : int or float, default=2
The minimum number of samples required to split an internal node:

- If int, then consider `min_samples_split` as the minimum number.
- If float, then `min_samples_split` is a fraction and `ceil(min_samples_split * n_samples)` are the minimum number of samples for each split.

SELEÇÃO DE VARIÁVEIS

SUPERVISIONADO: INTRÍNSECOS

RF IMPORTANCE



The screenshot shows the official documentation for `sklearn.feature_selection.SelectFromModel` on the scikit-learn website. The page includes a sidebar with navigation links (Prev, Up, Next) and version information (scikit-learn 1.1.1). The main content area displays the class name, its signature, a brief description, and a list of parameters with their default values and descriptions.

sklearn.feature_selection.SelectFromModel

```
class sklearn.feature_selection.SelectFromModel(estimator, *, threshold=None, prefit=False, norm_order=1, max_features=None, importance_getter='auto')
```

Meta-transformer for selecting features based on importance weights.

New in version 0.17.

Read more in the [User Guide](#).

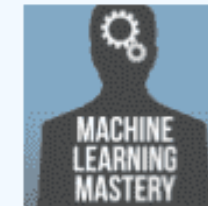
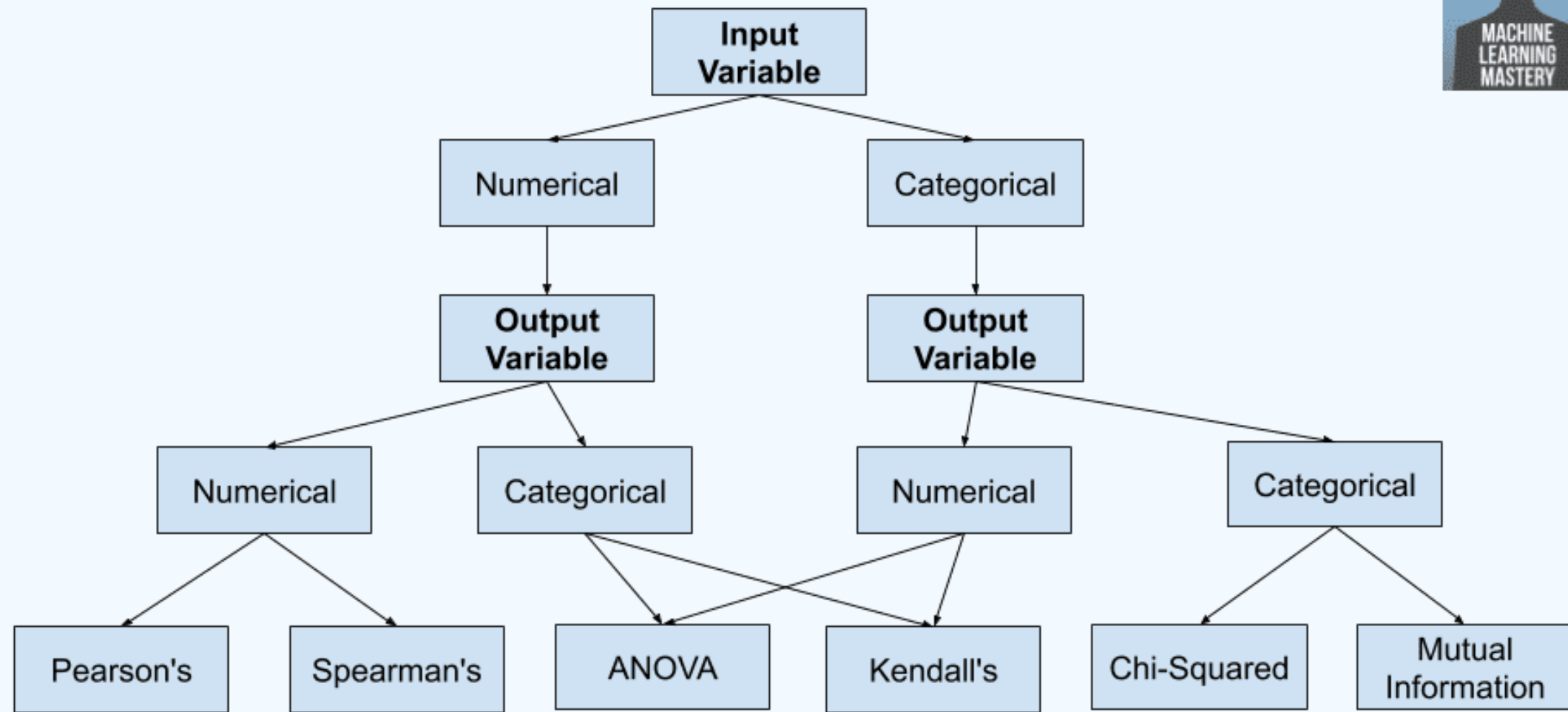
Parameters:

- estimator : object**
The base estimator from which the transformer is built. This can be both a fitted (if `prefit` is set to `True`) or a non-fitted estimator. The estimator should have a `feature_importances_` or `coef_` attribute after fitting. Otherwise, the `importance_getter` parameter should be used.
- threshold : str or float, default=None**
The threshold value to use for feature selection. Features whose importance is greater or equal are kept while the others are discarded. If "median" (resp. "mean"), then the `threshold` value is the median (resp. the mean) of the feature importances. A scaling factor (e.g., "1.25*mean") may also be used. If `None` and if the estimator has a parameter `penalty` set to `l1`, either explicitly or implicitly (e.g. Lasso), the threshold used is `1e-5`. Otherwise, "mean" is used by default.
- prefit : bool, default=False**
Whether a prefit model is expected to be passed into the constructor directly or not. If `True`, `estimator` must be a fitted estimator. If `False`, `estimator` is fitted and updated by calling `fit` and `partial_fit`, respectively.
- norm_order : non-zero int, inf, -inf, default=1**
Order of the norm used to filter the vectors of coefficients below `threshold` in the case where the `coef_` attribute of the estimator is of dimension 2.
- max_features : int, callable, default=None**
The maximum number of features to select.
 - If an integer, then it specifies the maximum number of features to allow.
 - If a callable, then it specifies how to calculate the maximum number of features allowed by using the out-

SELEÇÃO DE VARIÁVEIS

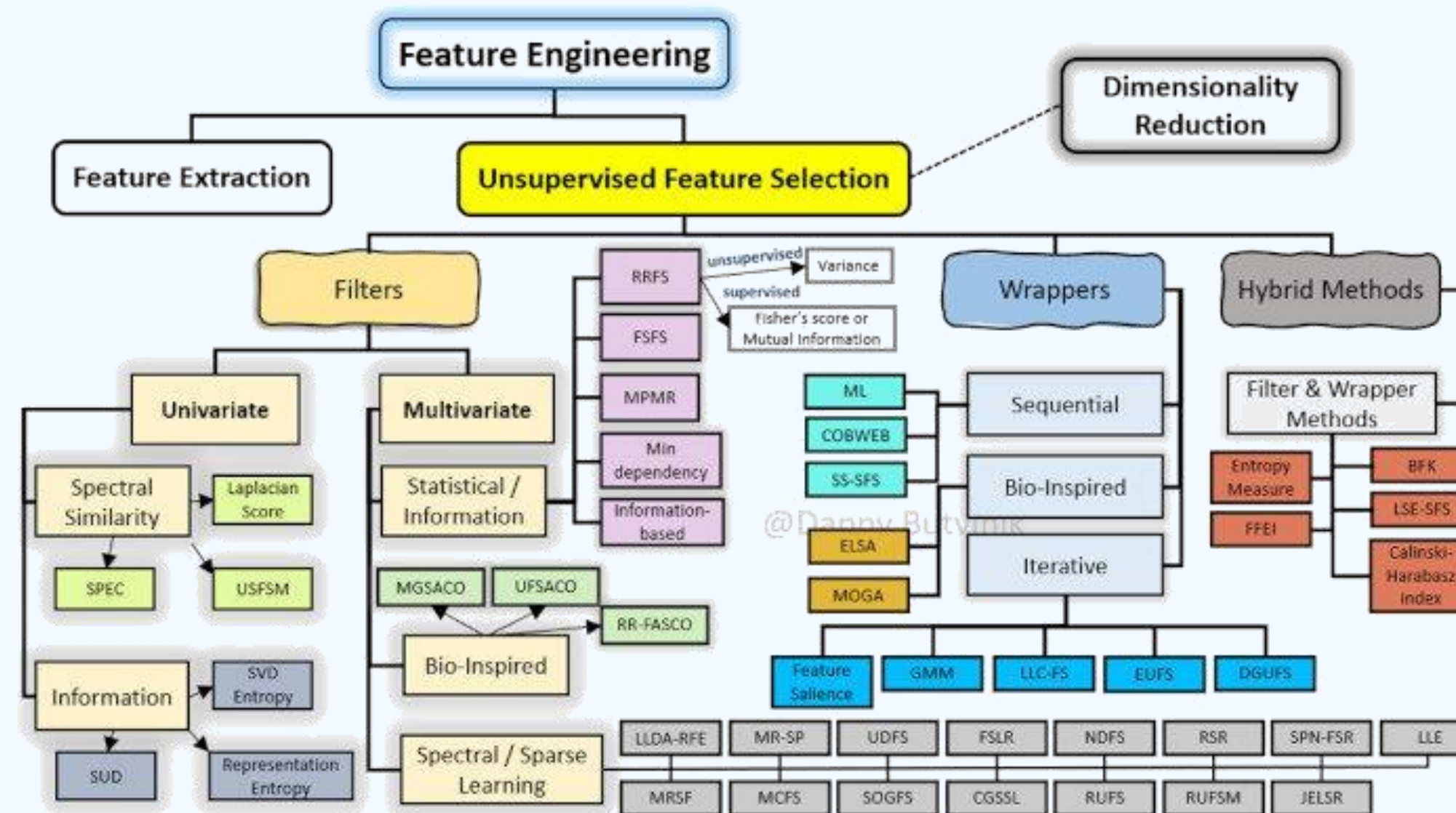
MÉTODOS

How to Choose a Feature Selection Method



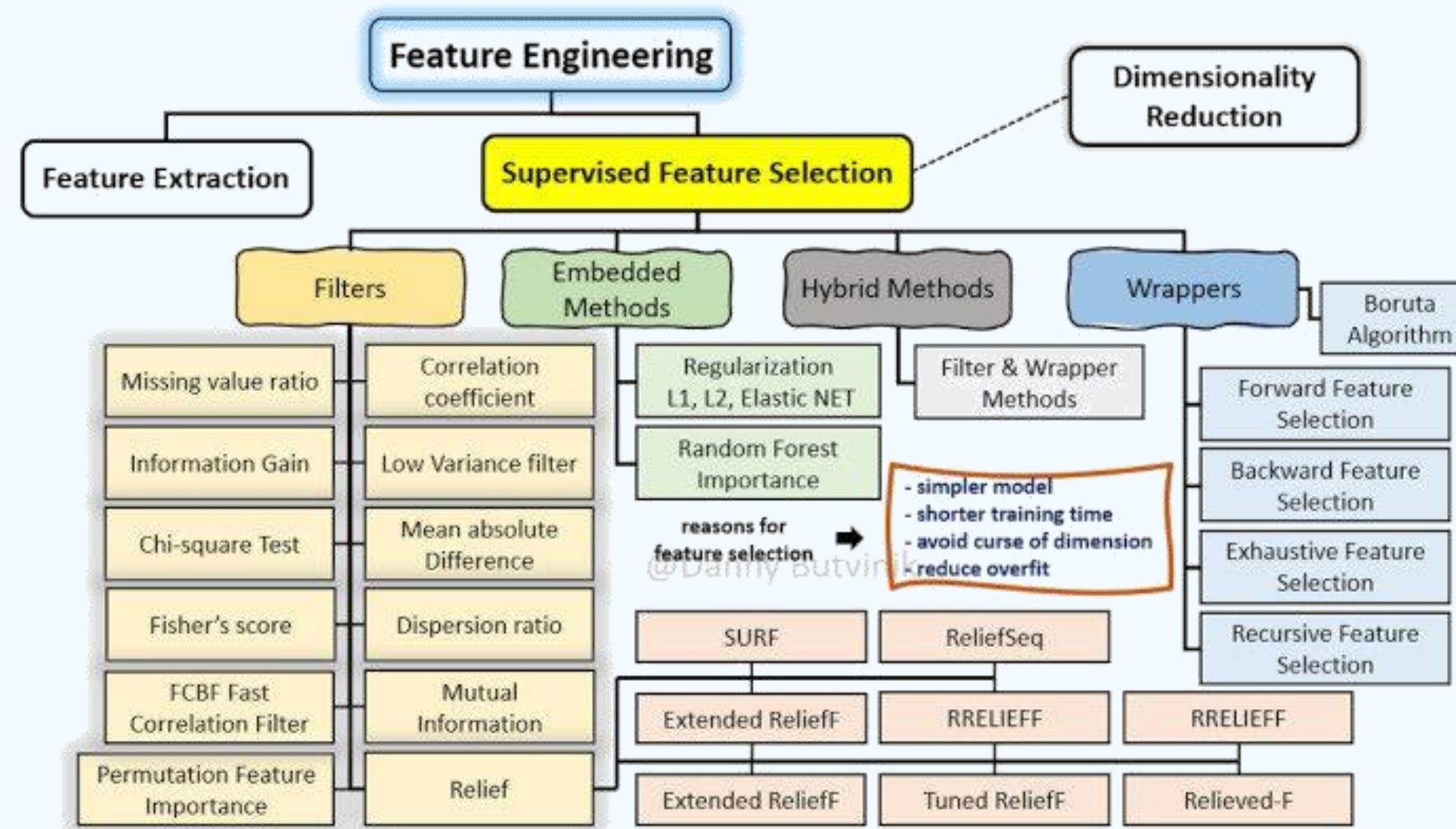
SELEÇÃO DE VARIÁVEIS

MÉTODOS



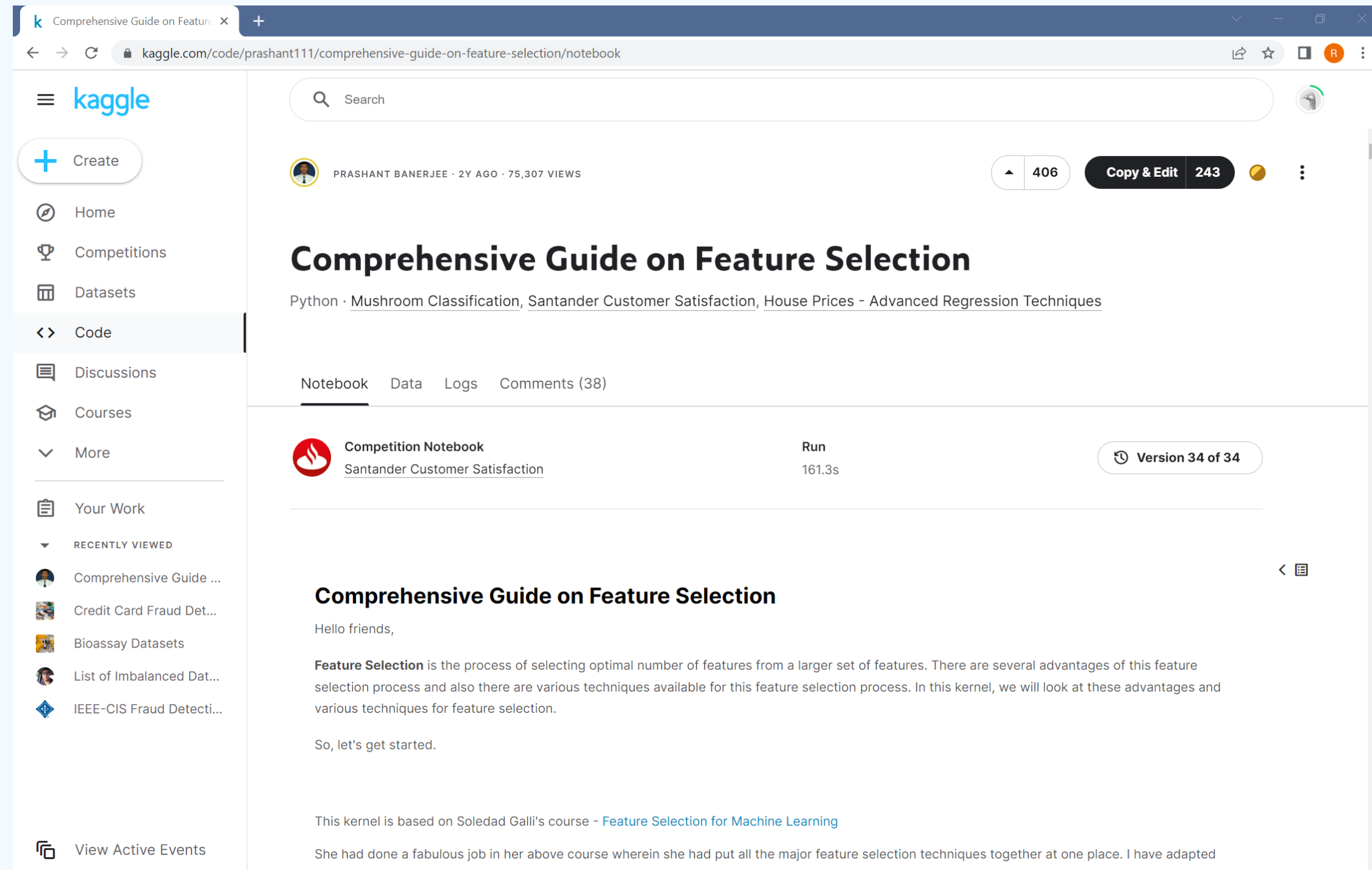
SELEÇÃO DE VARIÁVEIS

MÉTODOS



SELEÇÃO DE VARIÁVEIS

LEITURA RECOMENDADA



The screenshot shows a Kaggle notebook interface. The browser address bar displays the URL: `kaggle.com/code/prashant111/comprehensive-guide-on-feature-selection/notebook`. The notebook is titled "Comprehensive Guide on Feature Selection" by Prashant Banerjee, posted 2 years ago with 75,307 views. It has 406 upvotes and 243 copies. The notebook is in Python and covers topics like Mushroom Classification, Santander Customer Satisfaction, and House Prices using advanced regression techniques. The left sidebar shows the Kaggle navigation menu with options like Home, Competitions, Datasets, Code, Discussions, Courses, and More. The notebook content area shows the title, a brief introduction, and the start of the text: "Hello friends, Feature Selection is the process of selecting optimal number of features from a larger set of features. There are several advantages of this feature selection process and also there are various techniques available for this feature selection process. In this kernel, we will look at these advantages and various techniques for feature selection. So, let's get started." The notebook is running on a "Competition Notebook" environment for "Santander Customer Satisfaction" with a runtime of 161.3s and is at version 34 of 34.