Project Research Laboratory Report

Laboratoire de Biologie Structurale de la Cellule BIOC

Ecole Polytechnique, France

Extending Boruta
- a popular feature selection Machine Learning algorithm –

Clémence Mottez

Supervisors: Thomas SIMONSON, Director of BIOC

Referent instructor: Ivan REVEGUK, PhD student at BIOC

Semester 4 Bachelor of Science

1st June 2023

Abstract

This report provides an overview of the Boruta feature selection algorithm and its extension: “eBoruta”. Feature selection is an important step in machine learning that involves identifying the most relevant features in a dataset. The Boruta algorithm is a popular method for feature selection that is based on random forests. It works by creating a shadow feature set and comparing the performance of each feature with its shadow feature to determine its importance. This report explains the eBoruta algorithm in detail and provides examples of its applications and its optimization. The results show that the eBoruta algorithm is effective in reducing the number of features in a dataset and improving the performance of machine learning models.

Introduction

Feature selection encompasses a broad spectrum of tools and approaches growing in tandem with machine learning discipline expansion witnessed over the last three decades. On the one hand, feature selection reduces machine learning data size and the models’ complexity. On the other hand, it facilitates interpretability and aids in explaining the application domain. The Boruta algorithm is a popular choice for feature selection built upon the Random Forest algorithm. It is well suited for medico/biological fields where the number of input features is large while the number of observations is limited. The initial algorithm’s formulation is bound to random forest due to a straightforward assessment of feature importance. While the latter is integral for the algorithm, alternative ways emerged, enabling decoupling from the random forest dependency. The first attempts to use Boruta with models homologous to random forest occasionally appeared within the machine learning community. However, no systematic attempts were made to properly expand Boruta to a fully model-agnostic implementation.

Recently, the model agnostic Boruta was successfully applied to a specific problem of supervised partitioning of a multiple sequence alignment. This implementation needs further systematic analysis and a thorough comparison with the initial algorithm’s formulation regarding feature selection efficiency and dynamics.

The main task is to benchmark different models and feature importance formulations with Boruta and study them. Finally, data analysis and extraction of theoretical insights will conclude the project.

1- Feature selection

Feature selection is an important step in the application of machine learning methods to various problems. In many cases, the data sets available to machine learning algorithms contain far too many variables to be practically useful for model building. Most of these variables are often irrelevant to the classification task at hand, and their relevance is not known in advance. Therefore, the selection of a small and optimal feature set that provides the best possible classification results is essential for practical reasons. This problem is known as the minimal-optimal problem and has been intensively studied, with many algorithms developed to reduce the feature set to a manageable size.

There are several disadvantages to working with an overlarge feature set. Firstly, dealing with a large feature set can slow down algorithms, requiring more resources and making the process inconvenient. Secondly, many machine learning algorithms exhibit a decrease in accuracy when the number of variables is significantly higher than optimal. Therefore, selecting the right feature set is crucial for producing accurate and effective models.

One of the key advantages of feature selection is that it can help to reduce the time and resources required to train and test machine learning models. By reducing the size of the feature set, it is possible to train and test models more quickly and with less computational power. This is particularly important in applications where models need to be updated frequently, or where real-time processing is required.

Another important advantage of feature selection is that it can help to improve the accuracy and generalizability of machine learning models. When too many features are included in a model, it can become overfit to the training data, resulting in poor generalization to new data. By selecting the most relevant features, it is possible to improve the generalization of models, leading to better performance on new data.

There are several approaches to feature selection, each with its own advantages and disadvantages. One common approach is to use filter methods, which select features based on statistical or other objective criteria. Another approach is to use embedded methods, which integrate feature selection into the model building process itself. These methods select features by considering their importance during model training. A third approach is to use wrapper methods, which evaluate the performance of different feature subsets by training and testing the model on different subsets of data. Boruta specifically is a wrapper-based feature selection algorithm that identifies the most important features

in a data set by comparing their importance to the importance of randomly generated shadow features.

Despite the importance of feature selection, there are some challenges associated with this process. One of the main challenges is that the optimal feature set can be highly dependent on the specific problem and data set being studied. Therefore, it is important to carefully evaluate different feature selection methods and their performance on different data sets.

2- Boruta

The Boruta algorithm is a feature selection method for machine learning that is based on random forests. Its purpose is to identify the most relevant features in a dataset. The algorithm was developed by Miron B. Kursa and Witold R. Rudnicki in 2010.

The Boruta algorithm works by creating a shadow feature set, which is a randomly permuted version of the original feature set. This is useful to eliminate any correlation with the response variable. The algorithm then trains a random forest model on the combined feature set, including the original and shadow features. The importance of each feature is calculated by comparing its performance in the model with its performance in the shadow feature set. If a feature's performance is not significantly different from that of its shadow feature, then it is considered unimportant and is removed from the feature set. If a feature's performance is significantly better than its shadow feature, then it is considered important and is kept in the feature set. Thus, it iteratively removes the features which are proved by a statistical test to be less relevant than random probes.

On a statistical point of view, Z scores are computed for each attribute. The maximum Z score among shadow attributes (MZSA) is identified, and any attribute with a better Z score is assigned a "hit". For attributes with undetermined importance, a two-sided test of equality is performed with the MZSA. Attributes with importance significantly lower than MZSA are deemed "unimportant" and permanently removed from the data set, while attributes with importance significantly higher than MZSA are deemed "important". The idea is that a feature is useful only if it’s capable of doing better than the best randomized feature.

The Boruta algorithm continues this process until all the features have been tested. At the end of the process, the algorithm returns a set of important features that are likely to be relevant for the machine learning model.

One of the benefits of the Boruta algorithm is that it can handle correlated features, which can be a challenge for other feature selection methods. It can also handle datasets with missing values, by randomly assigning values to missing features during the creation of the shadow feature set.

Thus, the Boruta algorithm is a useful tool for reducing the number of features in a dataset and improving the performance of machine learning models by identifying the most important features.

3- SHAP and eBoruta

1) Shap

SHapley Additive exPlanations can be used in various feature selection algorithms to provide interpretability and explainability. What SHAP does is quantifying the contribution that each feature brings to the prediction made by the model.

In wrapper-based feature selection, using SHAP can help to gain insights into the importance of features at each step and select subsets that maximize the model's performance. SHAP values are computed for each feature based on their contribution to the model's predictions. These values are then used to rank and select the most important features. At each iteration, the model is trained on the selected features, and SHAP values are recalculated to update the feature rankings.

In filter-based feature selection, SHAP can be used to provide insights into the relevance of features based on their individual contributions to the model's output. SHAP values are computed for each feature by considering the model's predictions when that feature is included or excluded. The absolute magnitude of the SHAP value represents the importance of a feature. Features can be ranked based on their SHAP values, and a threshold can be set to select the most relevant features.

In embedded feature selection, SHAP can be used to interpret the feature importance scores. SHAP values can be calculated for each feature using the model's feature importance scores. This allows for interpreting the importance of features in a more granular and individualized manner. SHAP values provide insights into the contribution of each feature to the model's predictions, taking into account the interactions between features.

By integrating SHAP into feature selection algorithms, we can make more informed decisions about feature inclusion or exclusion, leading to improved model performance, transparency, and interpretability.

2) eBoruta

The Boruta algorithm has been around for a while, but not many people have realized the advantages of incorporating SHAP importance, which makes the method model-agnostic. In the original publication, the Random Forest model was likely chosen due to its versatility and optimal balance between speed and accuracy. However, the algorithm itself does not necessitate a strict dependency on Random Forest. With eBoruta, any model can be used as long as the method to compute feature importance is defined.

Furthermore, this extension of Boruta offers additional parameters to experiment with, such as the p-value, percentile, test size, and more. These parameters can assist in guiding the selection of important features, providing more flexibility in the feature selection process.

4- Contribution of the student - Extension of Boruta

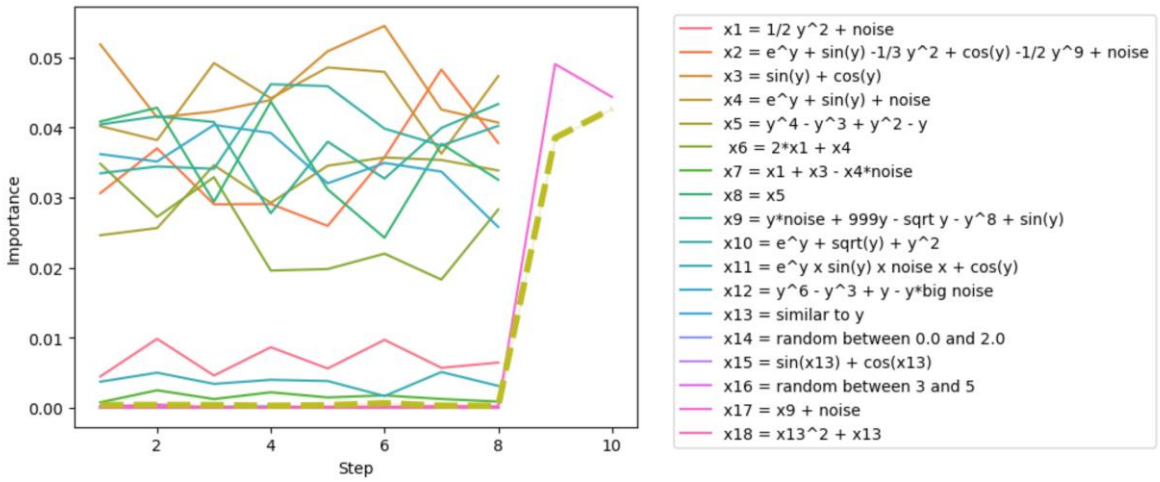
1) Find a dataset

To begin my work, I needed a dataset, so I decided to create one myself. By doing so, I could have precise control over the columns and tailor them according to my study objectives, particularly in predicting the algorithm's normal outcome.

I initialized a variable called "y" as a random integer ranging from 0 to 5. Subsequently, I created several "x" variables that were more or less correlated with "y." Here are some of the specific formulas and variables I used:

X1	$\frac{1}{2}y^2 + Noise$
X2	$e^y + sin(y) - \frac{1}{3}y^2 + cos(y) - \frac{1}{2}y^9 + Noise$
X3	$5y^4 - \frac{1}{2}y^3 + y^4 - \frac{1}{5}y^2 + y + Noise$
…	…
X17	$Random\ number\ between\ 0\ and\ 5 + Noise$
X18	$Noise$

The variables x1 to x12 have a link with y, and variables from x13 to x18 are not. Despite numerous attempts involving complex functions for the "x_i" variables and the inclusion of noise, I couldn't mislead the Boruta algorithm. It consistently selected the "x_i" variables that were genuinely connected to "y" while rejecting the others, regardless of the algorithm's parameters.



Plot of importance history

At step 8, all the variables from x1 to x12 are accepted and all the variables from x13 to x18 are rejected (except x17 that is in "tentative", meaning that the algorithm needs more step to decide).

Feature	Importance
x3 = sin(y) + cos(y)	0.048538
x4 = e^y + sin(y) + noise	0.043161
x10 = e^y + sqrt(y) + y^2	0.041615
x9 = y*noise + 999y - sqrt y - y^8 + sin(y)	0.039067
x5 = y^4 - y^3 + y^2 - y	0.034180
x8 = x5	0.034147
x2 = e^y + sin(y) -1/3 y^2 + cos(y) -1/2 y^9...	0.031540
x12 = y^6 - y^3 + y - y*big noise	0.030831
x6 = 2*x1 + x4	0.022180
x1 = 1/2 y^2 + noise	0.010080
x11 = e^y x sin(y) x noise x + cos(y)	0.004636
x7 = x1 + x3 - x4*noise	0.001747
shadow_x17 = x9 + noise	0.000431
shadow_x7 = x1 + x3 - x4*noise	0.000405
shadow_x14 = random between 0.0 and 2.0	0.000342
shadow_x1 = 1/2 y^2 + noise	0.000324
x14 = random between 0.0 and 2.0	0.000230
shadow_x9 = y*noise + 999y - sqrt y - y^8 + si...	0.000223
shadow_x4 = e^y + sin(y) + noise	0.000203
x17 = x9 + noise	0.000198
shadow_ x6 = 2*x1 + x4	0.000145
shadow_x2 = e^y + sin(y) -1/3 y^2 + cos(y) -...	0.000112
shadow_x13 = similar to y	0.000079
shadow_x11 = e^y x sin(y) x noise x + cos(y)	0.000078
shadow_x12 = y^6 - y^3 + y - y*big noise	0.000060
x18 = x13^2 + x13	0.000049
x16 = random between 3 and 5	0.000037
shadow_x15 = sin(x13) + cos(x13)	0.000032
x15 = sin(x13) + cos(x13)	0.000031
shadow_x3 = sin(y) + cos(y)	0.000018
x13 = similar to y	0.000014
shadow_x5 = y^4 - y^3 + y^2 - y	0.000009
shadow_x18 = x13^2 + x13	0.000008
shadow_x16 = random between 3 and 5	0.000008
shadow_x8 = x5	0.000000
shadow_x10 = e^y + sqrt(y) + y^2	0.000000

Ranking of the features

If we look at the ranking, we can see that, again, the features from x1 to x12 are the more important, and then the features from x13 to x18 are mixed with the shadow features, meaning that they are as important as random values.

I extensively tried to hyperparameterize the model, but I consistently obtained identical results, leaving me perplexed and questioning the effectiveness of my very noisy data.

However, his outcome confirms the effectiveness of Boruta. It is great! The algorithm works very well! But I can't analyse the algorithm with a "perfect" dataset. Moreover, its application in real-world scenarios warrants further exploration. Studies could focus on applying Boruta to diverse domains and datasets to validate its performance and generalize its effectiveness in practical settings.

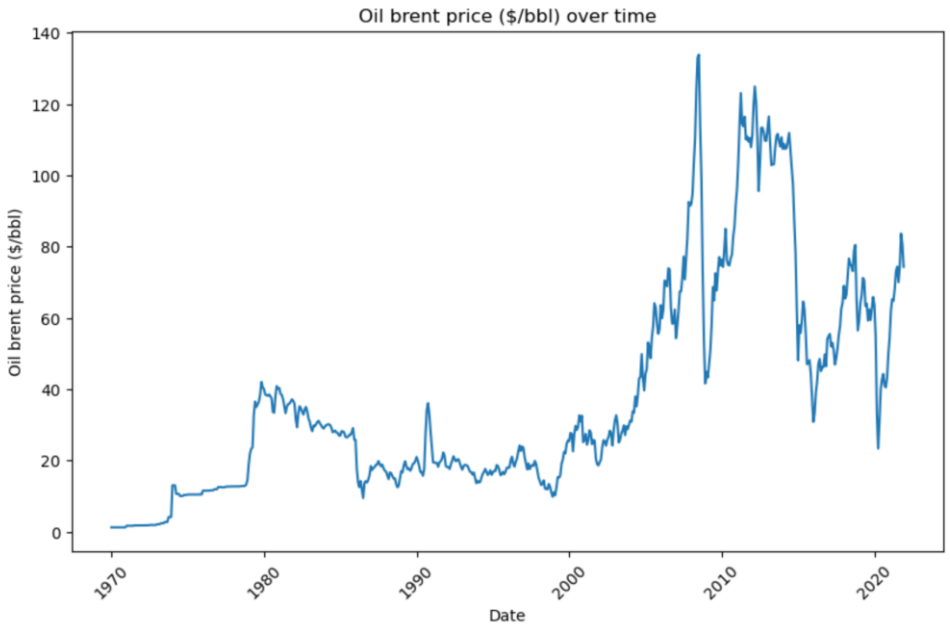
As a result, I decided to utilize real-world data for my study. I focused on predicting the price of oil. For that, I carefully created a dataset by considering various parameters that could potentially influence oil prices. I chose some parameters that were obviously related to oil prices, while I intentionally chosen others for their weaker correlation.

The final selection of variables is as follows:

y: Oil brent price

x: Year, Us crude oil reserves, Quantity oil embarked, World oil imports, World oil exports, Value of Solar Consumption, Inflation, Value of Wind Consumption, Value of Nuclear Consumption, Value of Natural Gas Consumption, Population, OPEC cuts on production, Price of gold, GDP Growth, Crude oil and NGL production, Quantity goods embarked, War, Electric car registrations, World-oil demand, Value of Freight Transport, Price cononut oil and Price of sugar.

We want to determine the most important x variables to predict the price of oil:

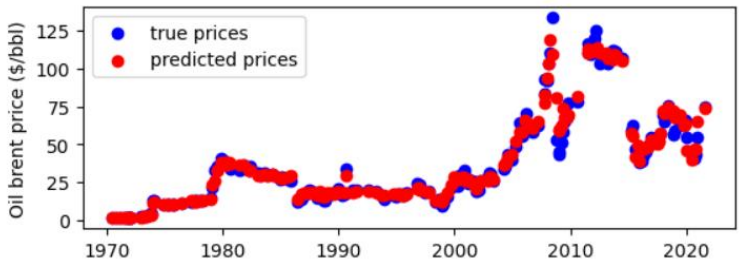


2) Other feature selection algorithm

To be sure of the consistency of the Boruta algorithm, I first studied other feature selection algorithms to be able to compare my results.

Before proceeding, I conducted a search to determine the most suitable machine learning models for my oil dataset. I performed cross-validation on various models, including Random Forest, Decision Tree, Linear Regression, and Polynomial Regression. I did some Hyperparameter tuning, and I aimed to maximize the R2 score and minimize the Cross-validation with mean squared error.

Among these models, the Decision Tree exhibited promising performance. Consequently, I decided to explore the XGBRegressor model, which utilizes an ensemble of decision trees for prediction. This model is renowned for its exceptional performance and capability to handle intricate datasets. To my satisfaction, the XGBRegressor model produced excellent results.



Predicted versus actual values using the xgbRegressor model

	XGB Regressor	Decision Tree	Random Forest
MSE	**23**	40	29

MSE of the different models

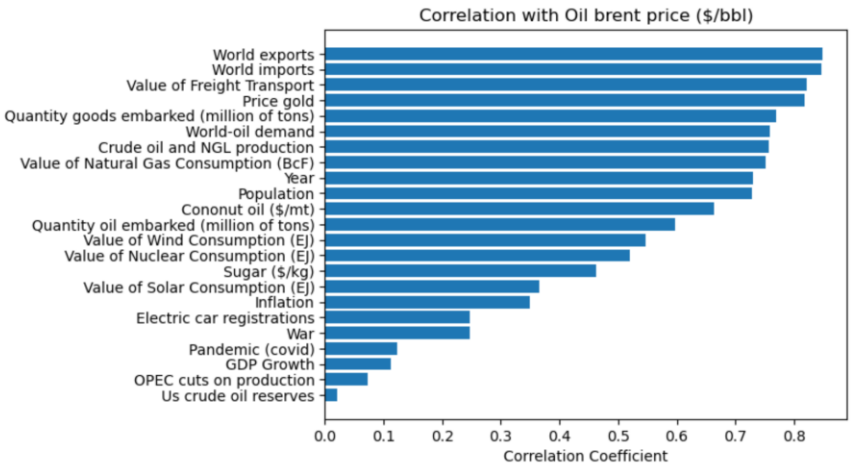
Random Forest also had a very good performance. Since it is the default model of the Boruta algorithm I will also study it.

Thus, I will explore both the XGBRegressor and the Random Forest Regressor models for my analysis.

As I explained before, there are different approaches to feature selection:

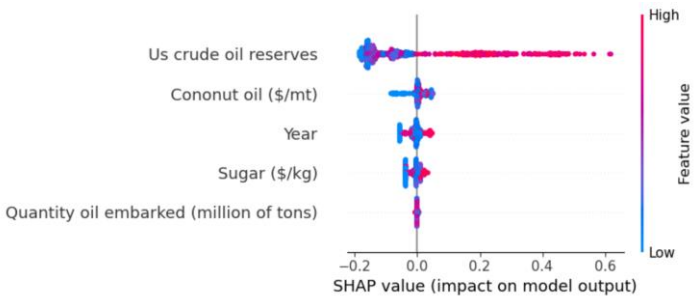
- One common approach is to use filter methods, which select features based on statistical or other objective criteria.

Correlation is a commonly used filter method in feature selection, where features are assessed based on their correlation with the target variable or with other features in the dataset. It helps identify relationships and dependencies between variables, allowing for the selection of informative and uncorrelated features.



- Another approach is to use embedded methods, which integrate feature selection into the model building process itself. These methods select features by considering their importance during model training.

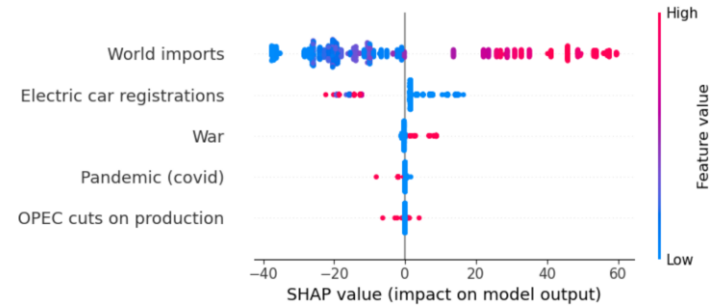
XGBoost (eXtreme Gradient Boosting) is a gradient boosting algorithm that excels in handling structured data and is known for its high performance and efficiency. XGBoost calculates feature importance scores based on how frequently a feature is used in decision trees during the boosting process. The higher the importance score, the more influential the feature is in predicting the target variable. It is also capable of performing feature selection as part of its training process. XGBoost has a built-in regularization technique called "regularized gradient boosting." This technique penalizes the inclusion of unnecessary features during the training process. The algorithm automatically learns to assign lower weights to less important features, effectively downplaying their influence on the model. As a result, the model focuses on the most informative features, which can aid in feature selection.



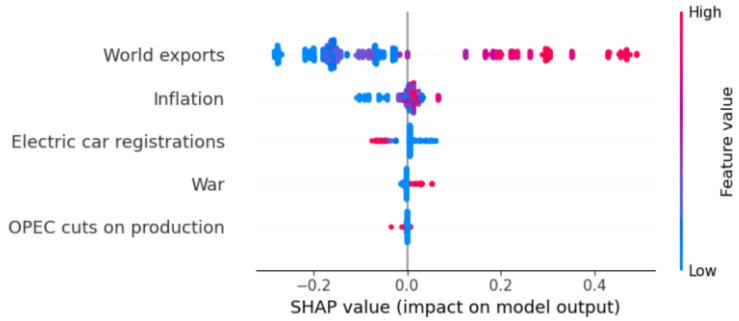
SHAP plot for XGboost*

- A third approach is to use wrapper methods (like Boruta), which evaluate the performance of different feature subsets by training and testing the model on different subsets of data.

Sequential Feature Selector (SFS) is an iterative method that sequentially adds or removes features from the feature subset based on their individual performance in combination with the existing subset. It explores different combinations of features and evaluates their impact on model performance using a performance metric. I used it with the "forward" parameter: the algorithm starts with an empty feature subset and iteratively adds one feature at a time, selecting the one that maximizes the performance metric. It continues adding features until a specified number is reached or until there is no improvement in performance.

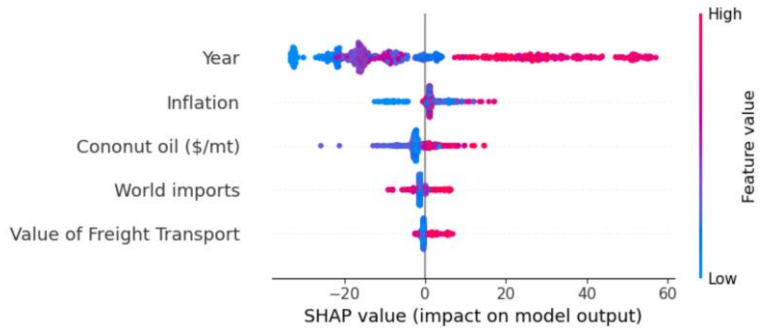


SHAP plot for SFS with XGBRegressor model

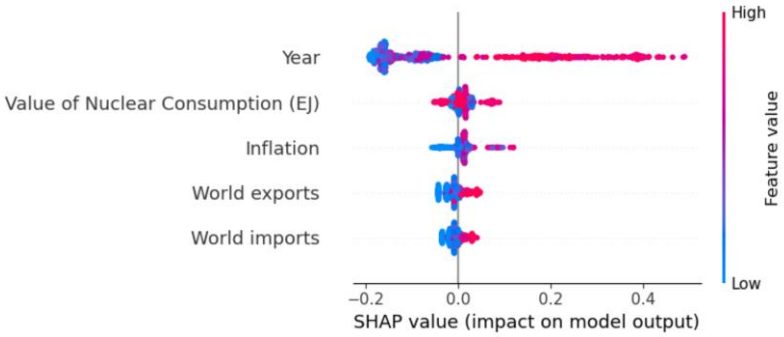


SHAP plot for SFS with RandomForestRegressor model

Then I also used Recursive Feature Elimination (RFE), a backward elimination method that recursively removes less important features from the feature set. It utilizes a machine learning model and assigns weights or ranks to each feature based on their importance. The algorithm then removes the least important feature and repeats the process until the desired number of features remains.



SHAP plot for RFE with XGBRegressor model



SHAP plot for RFE with RandomForestRegressor model

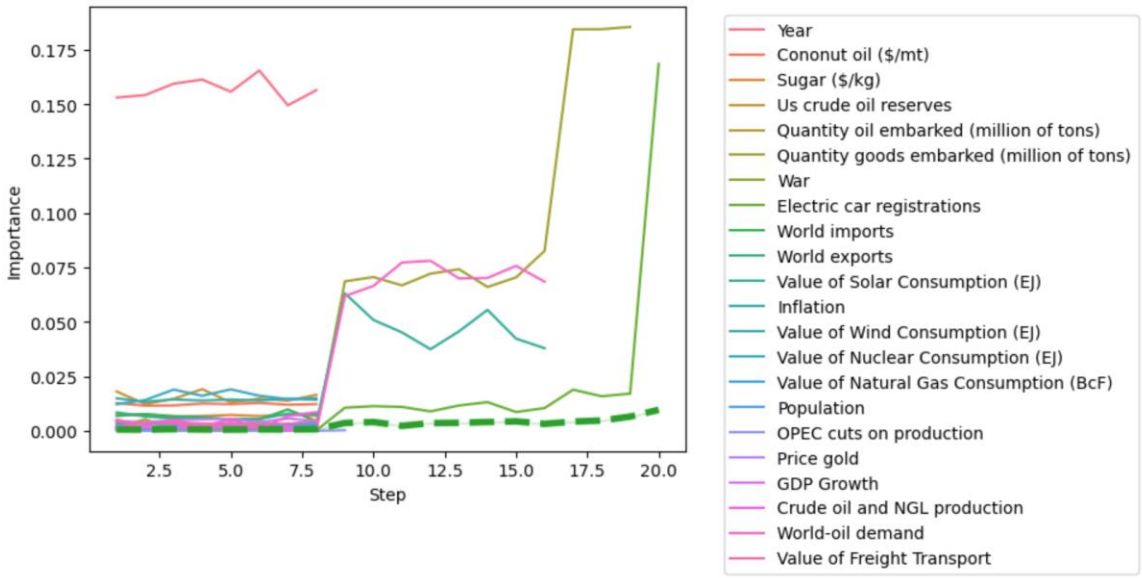
As a conclusion of this study, we should expect our Boruta algorithm to select more or less these following features: world export, world import, price coconut oil, price gold, year, inflation, US crude oil reserve.

**** How to interpret a SHAP plot: The y-axis of the plot represents the features in descending order of importance. The importance is determined based on the absolute magnitude of the SHAP values. The x-axis represents the range of the SHAP values. Positive SHAP values indicate features that push the model's prediction higher, while negative SHAP values indicate features that push the prediction lower. The color of each feature bar indicates the value of the feature. Darker shades represent higher feature values, while lighter shades represent lower values. The length of each feature bar represents the impact of that feature on the model's output. Longer bars indicate greater influence on the prediction, while shorter bars indicate lesser influence.*

*SHAP can show us both the global contribution by using the feature importances, and the local feature contribution for each instance of the problem. For example, in the first SHAP plot, we can see that high values of "US crude oil reserve" have a positive contribution on the prediction, while low values have a negative contribution. ****

3) eBoruta

I initially conducted a test using Boruta with the default parameters. Most of the features were "accepted," meaning they were considered linked to the response variable, except for "OPEC cuts on production," which was "rejected." Another feature, "Electric car registrations," was marked as "tentative," indicating that further steps were needed to make a decision about it.



Plot of importance history

I generated a plot showing the importance history during the Boruta process. The plot revealed that most features were selected by the algorithm at step 8. For the features that initially fell below the threshold, the algorithm continued evaluating them until they were either accepted or rejected. From the importance history plot, it is evident that the "Year" feature has the highest importance. "OPEC cuts on production" was quickly identified as "rejected." However, determining the status of "Quantity goods embarked," "Value of solar consumption," and "GDP growth" required more careful consideration, which occurred at steps 16 or 19. The plot also indicated that "Electric car registrations" was still undergoing evaluation at step 20. (If the algorithm were to continue, it would eventually reject this feature.)

In addition to the importance history plot, the Boruta algorithm also provides a "rank" function that orders the features based on their importance.

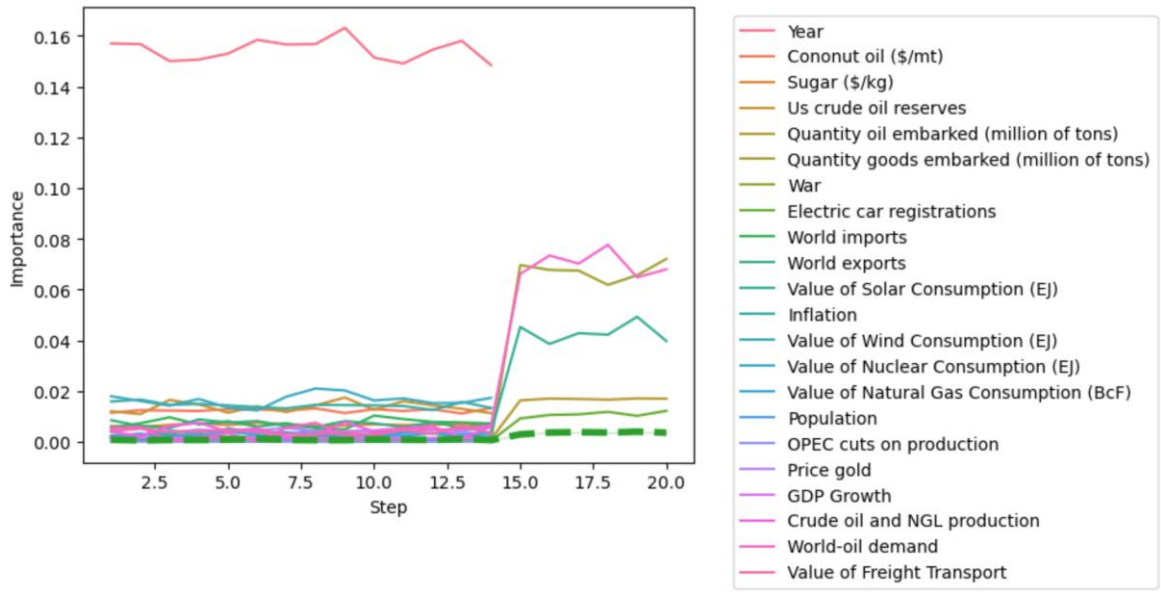
Feature	Importance
Year	0.159842
Inflation	0.015150
Value of Nuclear Consumption (EJ)	0.014833
Cononut oil ($/mt)	0.012274
Us crude oil reserves	0.012178
Sugar ($/kg)	0.008447
World imports	0.006078
Value of Freight Transport	0.005856
World exports	0.005784
Price gold	0.005459
World-oil demand	0.003815
Value of Wind Consumption (EJ)	0.003145
Population	0.003127
Value of Natural Gas Consumption (BcF)	0.002677
Crude oil and NGL production	0.002076
War	0.001907
GDP Growth	0.001743
Value of Solar Consumption (EJ)	0.001622
Quantity goods embarked (million of tons)	0.001508
Quantity oil embarked (million of tons)	0.000988
Electric car registrations	0.000724
OPEC cuts on production	0.000092

Rank of Boruta with default parameters

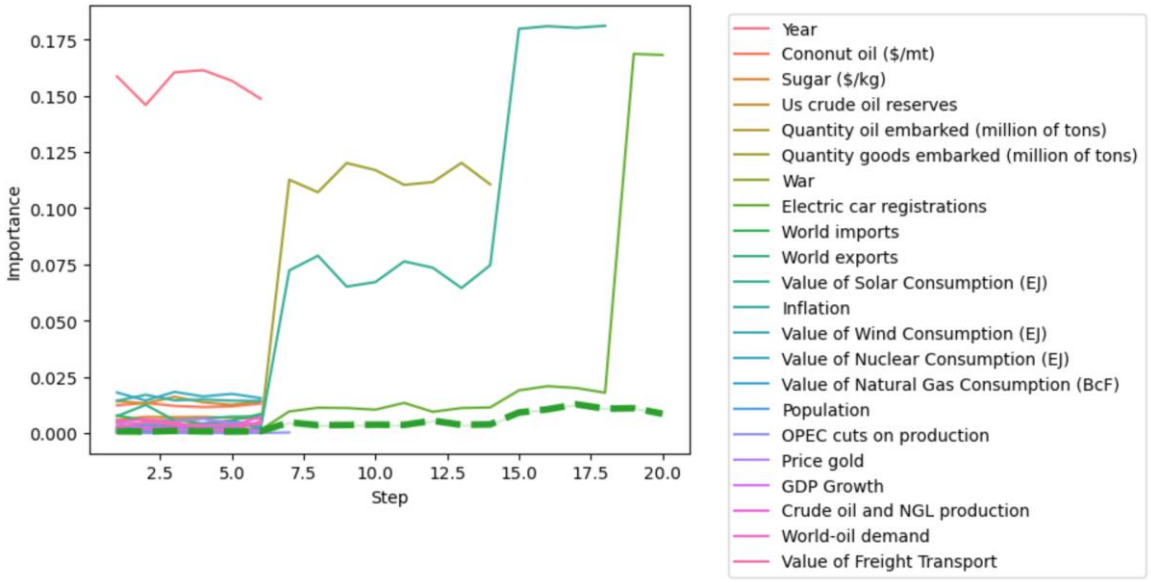
Benchmarking Boruta is an intriguing task as it allows us to gain insights and describe certain aspects of the algorithm. To ensure the reliability of our findings, it is essential to verify the stability of the results or minimize the impact of stochasticity. In order to achieve this, I conducted multiple iterations of the Boruta algorithm using the default parameters and compared the outcomes. I observed that the selected features, steps of selection, rankings, and other relevant factors remained consistently similar across iterations.

With the stability confirmed, it is now time to embark on the benchmarking process. I explored and experimented with various Boruta parameters to assess their impact on the results. By manipulating these parameters, I aimed to evaluate their effect on feature selection, ranking, and other aspects of the algorithm's performance.

- The p-value is a parameter that controls the statistical significance of the feature importance. It is the threshold that determines whether a feature is considered important or not. A feature is considered important if its importance score is significantly higher than the importance score of the random shadow features with a p-value of p.



Plot of importance history with p-value = 0.001

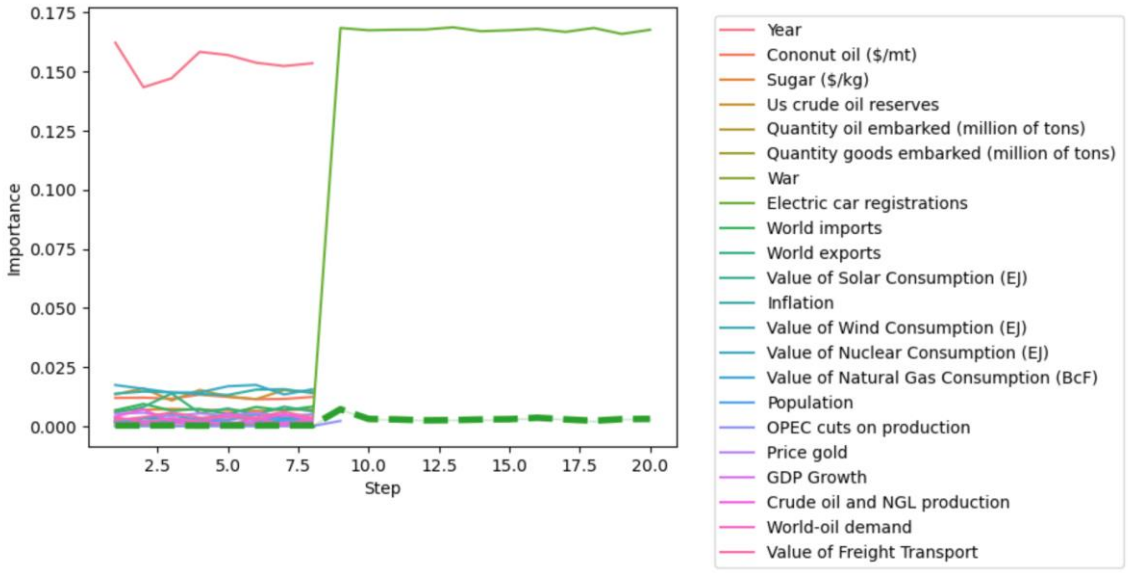


Plot of importance history with p-value = 0.1

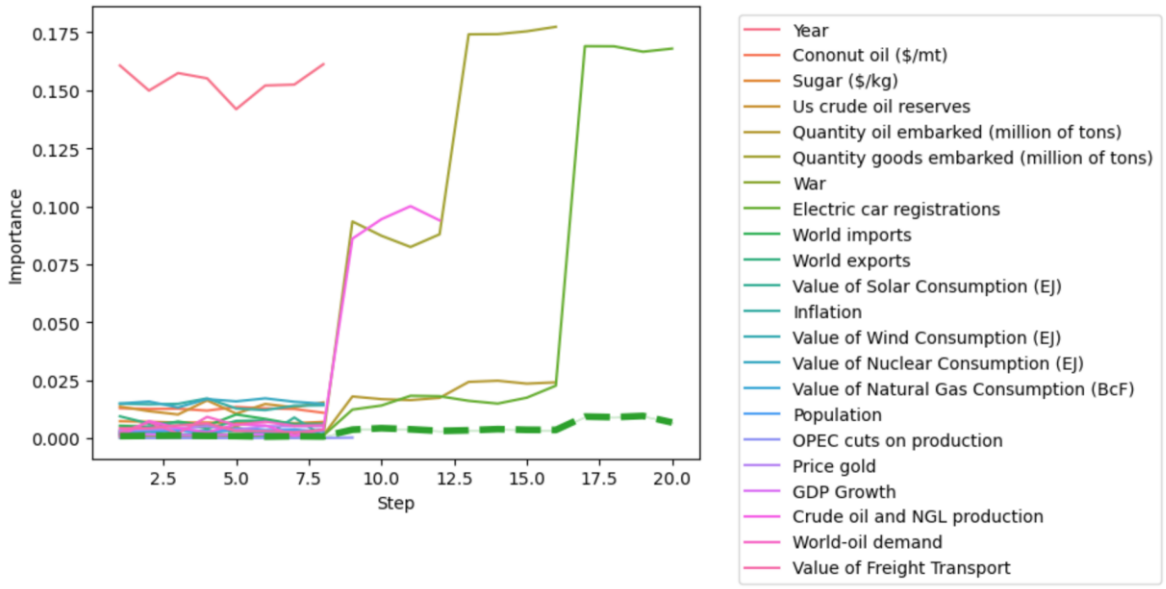
The p-value determines the level of significance required for a feature to be considered relevant. As we can see, a lower p-value threshold leads to a stricter selection criterion, resulting in a more narrow selection of features. Conversely, a higher p-value threshold leads to a higher number of features being considered relevant. We can easily see that important features are selected in more steps (around 14) with a lower p-value and in less steps (around 6) with a higher one.

The p-value threshold can affect precision by controlling the likelihood of including false positives (=irrelevant features labelled as "accepted"). A lower p-value threshold increases precision by reducing the chances of including false positives. By setting a stricter criterion for statistical significance, only features with strong evidence of importance are considered relevant, reducing the risk of including irrelevant features.

- The percentile is a parameter that controls the threshold for the feature importance. If the percentile is set to 90%, the threshold will be set to the 90th percentile of the feature importance scores. Any feature with an importance score above the threshold is considered important.



Plot of importance history with percentile = 70%

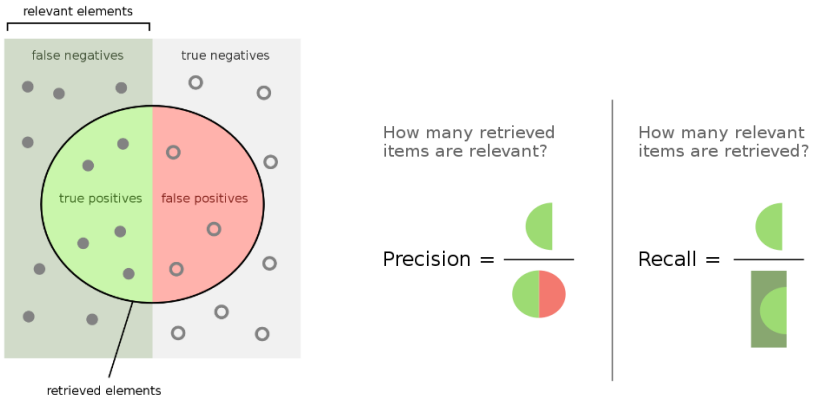


Plot of importance history with percentile = 100%

As we can see, a higher percentile value will result in fewer features being labeled as "confirmed" since it sets a higher importance threshold. This leads to a more selective feature selection process, where only the top-ranked features above the percentile cutoff are considered relevant. On the other hand, a lower percentile value will label more features as "confirmed," resulting in a larger set of relevant features.

The choice of percentile value also affects the balance between precision (selecting truly relevant features) and recall (including all relevant features). A higher percentile value increases precision but may lead to lower recall since only the top features are selected. Conversely, a lower percentile value improves recall by including more features but may result in lower precision as some irrelevant features might be included.

To summarize, the p-value threshold primarily influences precision by controlling the risk of including false positives, while the percentile parameter directly affects both precision and recall by setting the importance threshold for feature selection. The choice of these parameters should be based on the specific requirements of the problem, balancing the trade-off between precision and recall.



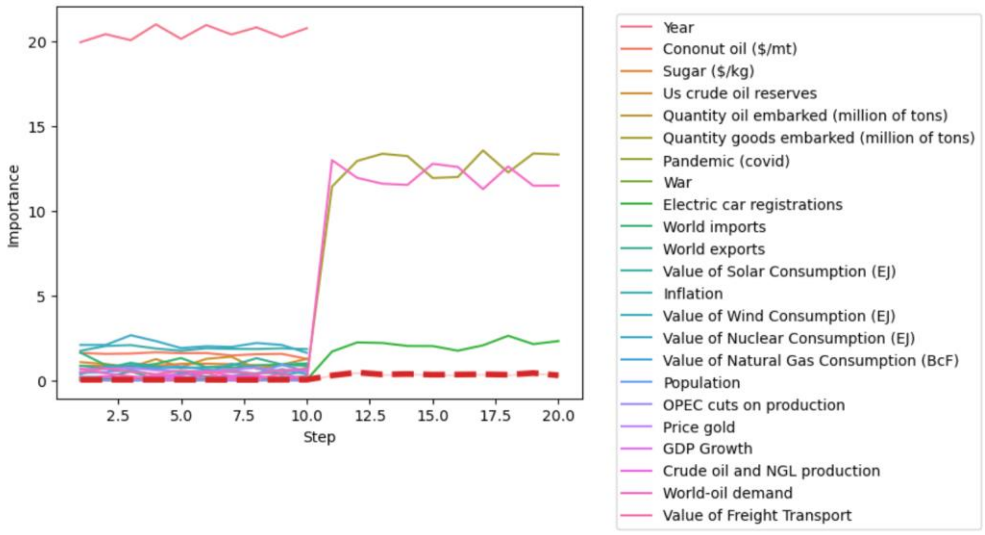
Precision and recall explanations (from Wikipedia)

The choice of these parameters obviously depends on the dataset I am using. In order to find the appropriate parameters for the eBoruta algorithm involves an iterative process of experimentation and evaluation. Here is the approach I followed:

- Set initial parameter values based on common defaults or recommendations. For example, here p-value to 0.05 and percentile = 70
- Run eBoruta on my dataset.
- Evaluate the results: which features has been identified as relevant.

- Train the XGB and Random Forest models using the selected features identified by Boruta and evaluate its performance using evaluation metrics such as accuracy, precision, recall, F1-score.
- Adjust the parameters based on the performance evaluation, analyze the trade-off between precision and recall. If I have too many false positives, I tried to decrease the p-value. If I noticed a lack of important features in the selection, I tried to increase the percentile.

Here are the most satisfactory results:



Plot of importance history with p-value = 0.01 and percentile = 80%

Feature	Importance
Year	20.683761
Value of Nuclear Consumption (EJ)	2.247001
Inflation	1.967047
Cononut oil ($/mt)	1.781908
Us crude oil reserves	1.529773
Sugar ($/kg)	1.073666
World exports	0.846162
World imports	0.788154
Price gold	0.709920
Value of Freight Transport	0.594799
Value of Solar Consumption (EJ)	0.547361
Value of Wind Consumption (EJ)	0.500222
Value of Natural Gas Consumption (BcF)	0.473678
World-oil demand	0.470438
Population	0.451585
GDP Growth	0.217480
War	0.210926
Quantity oil embarked (million of tons)	0.133912
Crude oil and NGL production	0.089163
Quantity goods embarked (million of tons)	0.039116
Electric car registrations	0.014652
OPEC cuts on production	0.014285
Pandemic (covid)	0.003641

Rank of Boruta with with p-value = 0.01 and percentile = 80%

Once again, "OPEC cuts on production" is rejected, as well as “Pandemic”. The features "Quantity goods embarked," "Electric car registrations," and "World-oil demand" remain in a "tentative" state. The ranking changed a little.

4) Comparison Boruta with other features selection algorithms

When examining the feature importance selection of Boruta, we observed that it consistently identified certain features as important. These features include "Year," "Inflation," "Price of coconut oil", "US crude oil reserve," and "World exports" and "Value of Nuclear consumption". Notably, these features align with the selections made by other feature selection algorithms we examined. Therefore, Boruta demonstrates its effectiveness as a reliable feature selection algorithm.

We can even verify the new accuracy scores by training the selected data:

	XGB Regressor	Random Forest
Raw Data	**23**	38
Selected data	**14**	17

MSE with the whole data and the selected data by eBoruta

Thus, eBoruta reduces the size of our data, enabling less complex models' analysis and more interpretability. And it even makes the model more accurate!

I also trained my data with the Boruta algorithm to have a feature selection. I then compared this selection with the one of the eBoruta algorithm to see how much the SHAP importance is useful with my data. Here are the results:

	XGB Regressor	Random Forest
Raw Data	**23**	38
Boruta selection	22	**21**
eBoruta selection	**14**	17

This proves the usefulness of the SHAP importance in the eBoruta algorithm.

However, if we delve into the rankings and rejected features, we notice that "Pandemic", "War" and "OPEC cuts on production" tend to be either rejected or ranked very low in terms of importance. These features are of a binary nature, denoted as either "0" (indicating no cut on production or no war) or "1" (indicating cut on production or war). Interestingly, some other feature selection wrapper methods, such as the Sequential Feature Selector, consider these features to be important. Hence, it suggests that Boruta might not be the optimal algorithm for handling this specific type of feature.

Overall, while Boruta performs well and consistently identifies relevant features, it may not be the most suitable choice when dealing with binary features such as "War", “Pandemic” and "OPEC cuts on production." Other feature selection wrapper methods may provide better results for such cases.

5) Other importance measures

Using SHAP importance in the Boruta algorithm can be a complex and time-consuming process. The computational complexity of SHAP values grows exponentially with the number of features, making exact computation infeasible for large feature sets. Indeed, it requires calculating Shapley values for each feature, which involves evaluating all possible feature combinations and their respective predictions. As the number of features and data points increases, the computational cost becomes prohibitively high.

To address this issue, it can be useful to explore other importance measures available in the XGBoost library. By setting importance_type to one of the alternative importance measures, such as 'gain', 'weight', 'cover', 'total_gain', or 'total_cover', you can obtain feature importance scores without the need for calculating SHAP values.

Let us study the impact of the different importance measures with the Boruta algorithm to see which one provides similar results as with SHAP importance for the oil data set:

- ‘weight’: the number of times a feature is used to split the data across all trees.

Feature	Importance
Value of Freight Transport	0.461583
Inflation	0.182272
World imports	0.149120
Year	0.091471
Quantity goods embarked (million of tons)	0.072491
Crude oil and NGL production	0.013580
Cononut oil ($/mt)	0.005383
GDP Growth	0.005216
Quantity oil embarked (million of tons)	0.005203
Value of Nuclear Consumption (EJ)	0.003412
Sugar ($/kg)	0.003081
Us crude oil reserves	0.003043
War	0.002602
Price gold	0.000874
OPEC cuts on production	0.000337
World-oil demand	0.000175
Value of Solar Consumption (EJ)	0.000158
Population	0.000000
Value of Natural Gas Consumption (BcF)	0.000000
World exports	0.000000
Electric car registrations	0.000000
Pandemic (covid)	0.000000
Value of Wind Consumption (EJ)	0.000000

Accepted:

['Year' 'Cononut oil ($/mt)' 'Quantity goods embarked (million of tons)' 'World imports' 'Inflation' 'Crude oil and NGL production' 'Value of Freight Transport']

Rejected:

['Us crude oil reserves' 'Pandemic (covid)' 'Electric car registrations' 'World exports' 'Value of Wind Consumption (EJ)' 'Value of Nuclear Consumption (EJ)' 'Value of Natural Gas Consumption (BcF)' 'Population' 'OPEC cuts on production' 'Price gold' 'World-oil demand']

Tentative:

['Sugar ($/kg)' 'Quantity oil embarked (million of tons)' 'War' 'Value of Solar Consumption (EJ)' 'GDP Growth']

- ‘gain’: the average gain across all splits the feature is used in.

Feature	Importance
Year	0.400158
Cononut oil ($/mt)	0.221474
Price gold	0.157686
Sugar ($/kg)	0.108162
Quantity oil embarked (million of tons)	0.028130
Inflation	0.017036
Us crude oil reserves	0.016244
GDP Growth	0.014263
War	0.008320
World imports	0.007132
Value of Nuclear Consumption (EJ)	0.006339
Value of Freight Transport	0.004358
Crude oil and NGL production	0.003962
OPEC cuts on production	0.003170
Quantity goods embarked (million of tons)	0.001981
World-oil demand	0.001189
Value of Solar Consumption (EJ)	0.000396
World exports	0.000000
Value of Wind Consumption (EJ)	0.000000
Electric car registrations	0.000000
Value of Natural Gas Consumption (BcF)	0.000000
Population	0.000000
Pandemic (covid)	0.000000

Accepted:

['Year' 'Cononut oil ($/mt)' 'Price gold']

Rejected:

['Us crude oil reserves' 'Quantity oil embarked (million of tons)' 'Quantity goods embarked (million of tons)' 'Pandemic (covid)' 'War' 'Electric car registrations' 'World imports' 'World exports' 'Value of Solar Consumption (EJ)' 'Inflation' 'Value of Wind Consumption (EJ)' 'Value of Nuclear Consumption (EJ)' 'Value of Natural Gas Consumption (BcF)' 'Population' 'OPEC cuts on production' 'GDP Growth' 'Crude oil and NGL production' 'World-oil demand' 'Value of Freight Transport']

Tentative:

['Sugar ($/kg)']

- ‘cover’: the average coverage across all splits the feature is used in.

Feature	Importance
Crude oil and NGL production	0.102700
Value of Freight Transport	0.098758
OPEC cuts on production	0.097988
Quantity oil embarked (million of tons)	0.084391
Us crude oil reserves	0.083255
Inflation	0.066092
World imports	0.061682
Quantity goods embarked (million of tons)	0.055969
Cononut oil ($/mt)	0.055191
Value of Nuclear Consumption (EJ)	0.052661
Price gold	0.050852
Sugar ($/kg)	0.046326
GDP Growth	0.044374
Year	0.033532
World-oil demand	0.031244
War	0.028579
Value of Solar Consumption (EJ)	0.006406
Value of Wind Consumption (EJ)	0.000000
Value of Natural Gas Consumption (BcF)	0.000000
Population	0.000000
World exports	0.000000
Electric car registrations	0.000000
Pandemic (covid)	0.000000

Accepted: []

Rejected:

['Year' 'Cononut oil ($/mt)' 'Sugar ($/kg)' 'Us crude oil reserves' 'Quantity oil embarked (million of tons)' 'Quantity goods embarked (million of tons)' 'Pandemic (covid)' 'War' 'Electric car registrations' 'World imports' 'World exports' 'Value of Solar Consumption (EJ)' 'Inflation' 'Value of Wind Consumption (EJ)' 'Value of Nuclear Consumption (EJ)' 'Value of Natural Gas Consumption (BcF)' 'Population' 'Price gold' 'GDP Growth' 'Crude oil and NGL production' 'World-oil demand']

Tentative:

['OPEC cuts on production' 'Value of Freight Transport']

- ‘total_gain’: the total gain across all splits the feature is used in.

Feature	Importance
Year	0.814146
Inflation	0.069070
Value of Freight Transport	0.044745
Cononut oil ($/mt)	0.026517
World imports	0.023654
Sugar ($/kg)	0.007412
Quantity oil embarked (million of tons)	0.003255
Quantity goods embarked (million of tons)	0.003194
Price gold	0.003064
GDP Growth	0.001655
Crude oil and NGL production	0.001197
Us crude oil reserves	0.001099
War	0.000482
Value of Nuclear Consumption (EJ)	0.000481
OPEC cuts on production	0.000024
World-oil demand	0.000005
Value of Solar Consumption (EJ)	0.000001
Population	0.000000
Value of Natural Gas Consumption (BcF)	0.000000
World exports	0.000000
Electric car registrations	0.000000
Pandemic (covid)	0.000000
Value of Wind Consumption (EJ)	0.000000

Accepted:

['Year' 'Cononut oil ($/mt)' 'Sugar ($/kg)' 'Quantity oil embarked (million of tons)' 'Quantity goods embarked (million of tons)' 'World imports' 'Inflation' 'Price gold' 'GDP Growth' 'Value of Freight Transport']

Rejected:

['Pandemic (covid)' 'War' 'Electric car registrations' 'World exports' 'Value of Solar Consumption (EJ)' 'Value of Wind Consumption (EJ)' 'Value of Nuclear Consumption (EJ)' 'Value of Natural Gas Consumption (BcF)' 'Population' 'OPEC cuts on production' 'World-oil demand']

Tentative:

['Us crude oil reserves' 'Crude oil and NGL production']

- ‘total_cover’: the total coverage across all splits the feature is used in.

Feature	Importance
Year	0.288777
Cononut oil ($/mt)	0.263058
Price gold	0.172569
Sugar ($/kg)	0.107836
Quantity oil embarked (million of tons)	0.051089
Us crude oil reserves	0.029105
Inflation	0.024232
GDP Growth	0.013621
World imports	0.009467
Value of Freight Transport	0.009263
Crude oil and NGL production	0.008757
Value of Nuclear Consumption (EJ)	0.007184
OPEC cuts on production	0.006684
War	0.005117
Quantity goods embarked (million of tons)	0.002386
World-oil demand	0.000799
Value of Solar Consumption (EJ)	0.000055
World exports	0.000000
Electric car registrations	0.000000
Value of Wind Consumption (EJ)	0.000000
Value of Natural Gas Consumption (BcF)	0.000000
Population	0.000000
Pandemic (covid)	0.000000

Accepted:

['Year']

Rejected:

['Sugar ($/kg)' 'Us crude oil reserves' 'Quantity oil embarked (million of tons)' 'Quantity goods embarked (million of tons)' 'Pandemic (covid)' 'War' 'Electric car registrations' 'World imports' 'World exports' 'Value of Solar Consumption (EJ)' 'Inflation' 'Value of Wind Consumption (EJ)' 'Value of Nuclear Consumption (EJ)' 'Value of Natural Gas Consumption (BcF)' 'Population' 'OPEC cuts on production' 'GDP Growth' 'Crude oil and NGL production' 'World-oil demand' 'Value of Freight Transport']

Tentative:

['Cononut oil ($/mt)' 'Price gold']

We can notice that for some of the importance types such as “weight”, and “total gain”, the selection of the feature is good. But only “total gain” provides a good ranking of the feature importance (similar as the one of Boruta).

The total gain importance type measures the total contribution of each feature to the model's performance by considering the improvement in the objective function (e.g., mean squared error, log loss) attributed to that feature. The total gain importance type takes into account both the frequency of feature use for splitting and the quality of those splits. A feature that consistently leads to significant reductions in the objective function across multiple splits and trees will have a higher total gain importance score. The total gain importance is useful for identifying features that have a strong impact on the model's performance. It can help in feature selection, identifying the most influential features, and gaining insights into the underlying patterns and relationships learned by the XGBoost model.

If we look at the time complexity of the algorithm, we can see that using the “total gain” importance type of XGboost is much faster than using the SHAP importance: less than 3 seconds against 3:34 minutes! for a single run!

By utilizing the "total gain" importance measure, you can obtain insights into the feature importance that align with the outcomes achieved through the Boruta algorithm. The advantage lies in the improved computational efficiency, allowing for faster analysis and decision-making in feature selection tasks related to the oil dataset.

Although SHAP importance offers a comprehensive perspective on feature contributions, the utilization of other importance measures can provide valuable insights while significantly reducing the computational complexity and runtime, making the Boruta algorithm more practical and feasible for feature selection tasks. In the context of my oil dataset, the "total gain" importance measure serves as a suitable alternative to SHAP in the Boruta algorithm. This importance measure yields comparable results to Boruta while significantly reducing the computational time required for execution.

6) New library

The eBoruta library is currently in the development phase, and not all model types have been thoroughly tested yet. Throughout my studies, I have encountered issues and raised concerns whenever I discovered bugs or instances where the library failed. One specific problem I encountered was with the rank function, as it did not correctly select the appropriate features or accurately rank their importance. These issues highlight areas where improvements and fixes were needed in the eBoruta library.

5- Challenges

During the course of my research, I encountered several challenges that required careful navigation and problem-solving. One significant difficulty arose from the data itself, as it was hard to acquire and contained missing values. The scarcity of available data posed a hindrance to conducting comprehensive analyses. However, I tackled this issue by employing data imputation techniques.

Another challenge I faced was the extensive runtime of the Boruta algorithm. Its lengthy execution time significantly impacted the efficiency of my experiments. To overcome this challenge, I optimized the algorithm by reducing the dimensionality of the dataset, parallelizing the computations wherever possible and used less complex importance measure functions. These adjustments helped expedite the execution time while maintaining the integrity of the results.

I also encountered problems related to errors and library imports. Resolving these issues required meticulous debugging and troubleshooting. I meticulously reviewed the error messages, researched potential solutions, and sought assistance from online forums and my referent instructor. By persistently addressing these problems, I was able to overcome the obstacles and ensure the smooth functioning of the algorithm.

Furthermore, time constraints imposed significant pressure on completing all planned experiments and analyses. To manage this challenge, I prioritized the most crucial aspects of my research and refined my experimental design to focus on key objectives. By making strategic choices and effectively allocating my time and resources, I successfully conducted the essential experiments and obtained meaningful results within the given timeframe.

One particularly demanding aspect was the interpretation of the results and understanding their implications. To address this challenge, I undertook extensive literature reviews, consulted with domain experts, and ask my referent instructor. These actions helped me gain valuable insights and perspectives that facilitated a deeper understanding of my findings and their potential implications.

Overall, by employing data imputation techniques, optimizing the Boruta algorithm, resolving errors, managing time effectively, and seeking assistance when needed, I successfully overcame the challenges encountered during my research. These experiences not only enriched my problem-solving skills but also reinforced the importance of adaptability and resourcefulness in research endeavors.

6- Background material

Before delving into the project of extending the Boruta feature selection algorithm, it was necessary to first acquire a strong understanding of machine learning and feature selection techniques. As someone who had not previously worked with machine learning algorithms, the initial learning curve was steep. However, with some background in statistics and programming, I was able to quickly grasp the foundational concepts of machine learning, including supervised and unsupervised learning, regression analysis, decision trees, and ensemble methods.

To gain more practical experience with machine learning, I started working through some online tutorials and courses, in parallel with my CSE204 Introduction to Machine Learning course. Through these courses, I was able to develop a stronger understanding of the theoretical foundations of machine learning as well as practical experience working with real-world data sets.

One of the key concepts in machine learning that I needed to become familiar with for this project was feature selection. I read several research papers and textbooks on the topic, including "Feature Selection for Knowledge Discovery and Data Mining" by Liu and Motoda, and "Applied Predictive Modeling" by Kuhn and Johnson. Through these resources, I was able to learn about various feature selection techniques, including filter methods, wrapper methods, and embedded methods.

Finally, before embarking on the project of extending the Boruta feature selection algorithm, I spent some time researching the existing literature on Boruta and related algorithms. This included reading the original research paper on Boruta by Kursa and Rudnicki, as well as several follow-up papers and articles on the algorithm. Through this research, I gained a better understanding of the strengths and weaknesses of Boruta, and identified areas where the algorithm could potentially be improved or extended.

Conclusion

This study confirmed the usefulness of the eBoruta algorithm, an extension of the Boruta algorithm that incorporates the built-in SHAP importance measure. This novel feature selection method offers unbiased and consistent identification of significant and non-significant attributes within an information system. It introduces compelling parameters that assist in guiding the selection of features based on the specific dataset characteristics.

Summary

During my research, I explored eBoruta, an enhanced version of the Boruta algorithm that integrates the SHAP importance measure. To begin, I created an appropriate dataset for investigating oil prices. Subsequently, I examined the various hyperparameters provided by eBoruta, including the p-value and percentile. I explored different feature selection techniques, encompassing embedded, wrapper, and filter methods, and compared them with the eBoruta selection. Lastly, I investigated alternative importance measures that were similar to SHAP importance but boasted quicker execution times.

Outlook

The eBoruta feature selection algorithm has demonstrated its effectiveness in providing unbiased and stable selection of important attributes in various information systems. As the field of data analysis and machine learning continues to evolve, there are several potential avenues for future development and enhancement of eBoruta.

- Integration with Deep Learning: Deep learning models have gained significant attention and achieved remarkable success in various domains. An exciting direction for eBoruta's future development could involve investigating methods to incorporate and leverage the interpretability aspects of SHAP importance within the context of deep learning models. This would enable researchers to gain insights into the feature importance of complex deep learning architectures.
- Automation and Efficiency: Efforts can be directed towards further automating the parameter selection process in eBoruta. Developing intelligent techniques or algorithms to automatically determine optimal values for parameters such as p-value, percentile, and test size would simplify the feature selection process and enhance its efficiency.
- Visualization and Interpretability: Visualizing and interpreting the results of eBoruta can aid users in understanding and explaining the selected features. Future developments could focus on creating intuitive visualizations and summary reports that facilitate the interpretation of feature importance and contribute to better decision-making processes.

By embracing these future directions, eBoruta has the potential to continue empowering researchers and practitioners with a robust and interpretable feature selection method, enabling them to make informed decisions and uncover valuable insights from complex data.

Acknowledgments

I would like to express my sincere gratitude to my referent instructor Ivan Reveguk, whose guidance and support were invaluable throughout this project. I am very grateful for the time and effort that he devoted to helping me, which greatly contributed to the success of this project. I am glad to have had the opportunity to work under his supervision, and I will remember the lessons and skills that I have learned from him.

References

1) Textbooks and research papers:

Miron B. Kursa and Witold R. Rudnicki. “Feature Selection with the Boruta Package”. In: Journal of Statistical Software 36.11 (2010), pp. 1–13.

Leo Breiman. “Random Forests”. In: Machine Learning 45.1 (Oct. 2001), pp. 5–32. issn: 1573-0565.

Vittorio Fortino et al. “A Robust and Accurate Method for Feature Selection and Prioritization from Multi-Class OMICs Data”. In: PLoS ONE 9.9 (Sept. 2014)

Yuliang Pan et al. “Computational identification of binding energy hot spots in protein-RNA complexes using an ensemble approach.” In: Bioinformatics (Oxford, England) 34.9 (July 2017), pp. 1473–1480. issn: 1367-4803. doi: 10.1093/bioinformatics/btx822

Liu and Motoda, "Feature Selection for Knowledge Discovery and Data Mining" in Part of the book series “The Springer International Series in Engineering and Computer Science” (SECS, volume 454)

Kuhn and Johnson, "Applied Predictive Modeling"

2) Websites

https://towardsdatascience.com/boruta-shap-an-amazing-tool-for-feature-selection-every-data-scientist-should-know-33a5f01285c0

https://towardsdatascience.com/using-shap-values-to-explain-how-your-machine-learning-model-works-732b3f40e137

https://towardsdatascience.com/shap-explained-the-way-i-wish-someone-explained-it-to-me-ab81cc69ef30

https://towardsdatascience.com/boruta-explained-the-way-i-wish-someone-explained-it-to-me-4489d70e154a