# Status report.
# Feature selection package with IHT

Konstantin Pakulev, Viacheslav Pronin

April 2021

## 1 Project overview

Feature selection is of great importance for machine learning tasks especially for those that deal with high-dimensional data. There are many feature selection methods available, however, all of them are implemented in different libraries and lack a unified API. As all of the methods are different in their performance we want to select the best, however, the aforementioned problem makes it hard to perform a fair comparison of the existing methods.

Our project aims to mitigate the aforementioned issue by presenting a framework for evaluating different feature selection methods. We provide a baseline permutation importance [2] method in our pipeline (using implementation from sklearn) as well as implementations of two more recent methods: Normalised Iterative Hard Thresholding [1] and Simultaneous Feature and Feature Group Selection through Hard Thresholding [4]. The pipeline allows to perform evaluation on both synthetic and real data. The implementation is available here: https://github.com/adasegroup/FES-feature-selector

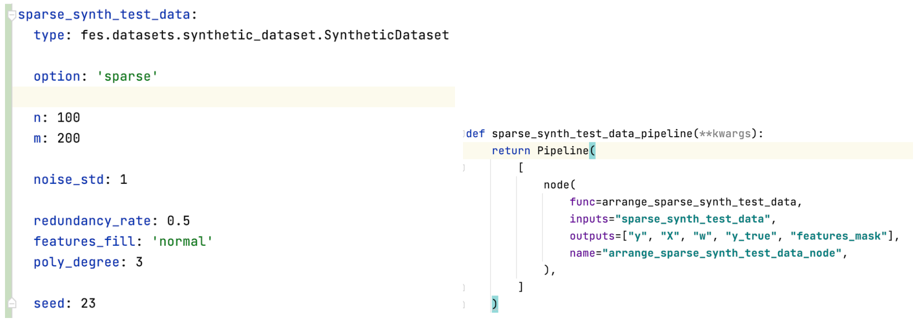
## 2 Project structure

To set up the structure of our framework as well as ensure reproducibility and the ease of maintenance of our project we use Kedro [3]. The underlying structure of this framework allows for the easy introduction of new datasets as well as methods and evaluation routines.

In order to add an existing dataset to our pipeline, it is enough to declare the dataset and its parameters inside a configuration file and create a node that takes the dataset as an input and processes it into a desirable format as illustrated in Figure 1. The usage of the configuration file allows having multiple instances of a dataset with different parameters.

The addition of a model and the evaluation for it can be done in a manner similar to the one explained before. In Figure 2 the registration of a linear regression model and the permutation importance method is shown. The inputs for those nodes are taken from the pre-processing of the dataset in a declarative manner. Any dataset, as long as it is able to provide the data in the format required by the pipeline shown in Figure 2 can be used for evaluation without any change of the code.

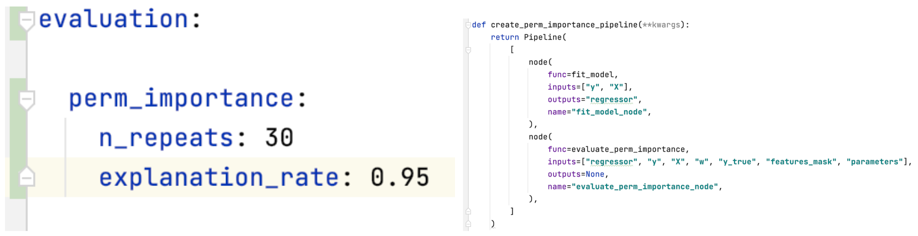
Finally, to provide the functionality of running different models on different datasets in an easy way the pipeline registry is created as shown in Figure 3.



```yaml
sparse_synth_test_data:
  type: fes.datasets.synthetic_dataset.SyntheticDataset

  option: 'sparse'

  n: 100
  m: 200

  noise_std: 1

  redundancy_rate: 0.5
  features_fill: 'normal'
  poly_degree: 3

  seed: 23
```

```python
def sparse_synth_test_data_pipeline(**kwargs):
    return Pipeline(
        [
            node(
                func=arrange_sparse_synth_test_data,
                inputs="sparse_synth_test_data",
                outputs=["y", "X", "w", "y_true", "features_mask"],
                name="arrange_sparse_synth_test_data_node",
            ),
        ]
    )
```

Figure 1: On the left: dataset declaration via config. On the right: setting up the dataset processing pipeline in the code.



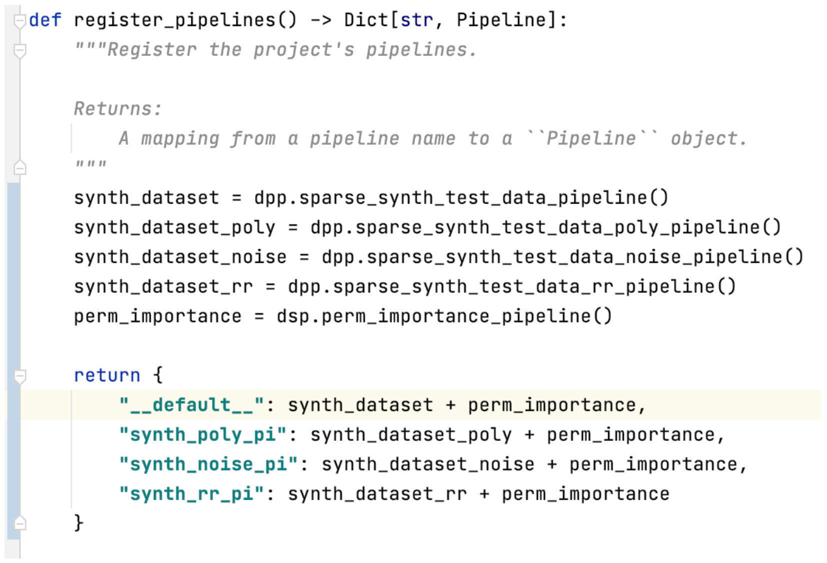
```yaml
evaluation:

  perm_importance:
    n_repeats: 30
    explanation_rate: 0.95
```

```python
def create_perm_importance_pipeline(**kwargs):
    return Pipeline(
        [
            node(
                func=fit_model,
                inputs=["y", "X"],
                outputs="regressor",
                name="fit_model_node",
            ),
            node(
                func=evaluate_perm_importance,
                inputs=["regressor", "y", "X", "w", "y_true", "features_mask", "parameters"],
                outputs=None,
                name="evaluate_perm_importance_node",
            ),
        ]
    )
```

Figure 2: On the left: evaluation parameters. On the right: linear regression fitting and evaluation with the permutation importance method.

## 3 Problem statement

The problem of feature selection can be formulated as follows. Let $A \in R^{n \times p}$ be a design matrix, $y \in R^n$ be a set of observations, $x \in R^p$ be a set of parameters and $\epsilon \in R^n$ be some noise. Assuming that observation are generated via $y = Ax + \epsilon$ we want to find such $\hat{x} \in R^p$ that satisfies optimization problem 1 where $s_1$ is some non-zero value.



```python
def register_pipelines() -> Dict[str, Pipeline]:
    """Register the project's pipelines.

    Returns:
        A mapping from a pipeline name to a ``Pipeline`` object.
    """
    synth_dataset = dpp.sparse_synth_test_data_pipeline()
    synth_dataset_poly = dpp.sparse_synth_test_data_poly_pipeline()
    synth_dataset_noise = dpp.sparse_synth_test_data_noise_pipeline()
    synth_dataset_rr = dpp.sparse_synth_test_data_rr_pipeline()
    perm_importance = dsp.perm_importance_pipeline()

    return {
        "__default__": synth_dataset + perm_importance,
        "synth_poly_pi": synth_dataset_poly + perm_importance,
        "synth_noise_pi": synth_dataset_noise + perm_importance,
        "synth_rr_pi": synth_dataset_rr + perm_importance
    }
```

Figure 3: A dictionary with different combinations of datasets and models. The selection of a particular combination can be done via command line

$$\begin{aligned} \underset{\hat{x}}{minimize} \quad & \frac{1}{2}||A\hat{x} - y||_2^2 \\ subject\ to \quad & \sum_{j=1}^{p} I(|\hat{x}_j| \neq 0) \leq s_1 \end{aligned} \tag{1}$$

Sometimes it also can be beneficial to select feature groups along with features, i.e. when the data has some grouping structures. In such cases the optimization problem is complemented by an additional constraint 2 where the entities of $x$ are separated into $|G|$ mutually exclusive groups, $G_j$ denotes indices that belong to the $j$-th group and $s_2$ is some non-zero value.

$$subject\ to \quad \sum_{j=1}^{|G|} I(||\hat{x}_{G_j}||_2 \neq 0) \leq s_2 \tag{2}$$

Solving the optimization 1 is a combinatorial problem for which no efficient algorithm exists for the general case. Instead sub-optimal algorithms are used that either convert the discrete constraints in 1 to their continuous surrogates or compute a local solution directly. In our project we will implement two methods that use the latter approach.

# 4 Project objectives

In section we introduce main challenges that we're going to face during the work on our project: methods implementations and evaluation protocol.

## 4.1 Baseline method

### 4.1.1 Permutation importance

Permutation importance is an approach to compute feature importances for any black-box estimator by measuring the decrease of a score when a feature is not available. Permutation importance is calculated after a model has been fitted (i.e., the estimator is required to be a fitted estimator compatible with scorer ). This method contains the following steps:

- A baseline metric, defined by scoring, is evaluated on a data set defined by the $X$ - the data set used to train the estimator or a hold-out set;
- A feature column from the validation set is permuted and the metric is evaluated again;
- The difference between the baseline metric and metric from permutation of the feature column is the permutation importance.

The mean of feature importance shows the degree of model performance accuracy deterioration with a random shuffling, and the standard deviation shows the variation of performance from one reshuffling to the next. Cases of negative values for permutation importances are possible, especially in small datasets; however, they merely depict the insufficiency of dataset, as they point out that "noisy" data happened to be more accurate than the real data, i.e. there is some random chance distortion.

This method is most appropriate for computing feature importances with a reasonably limited number of columns (features), as it can be resource-intensive.

## 4.2 Methods to-be-implemented

### 4.2.1 Normalized Iterative Hard Thresholding

The algorithm searches for $\hat{x}^{n+1}$ starting from $\hat{x}^0 = 0$ by applying the iterative procedure 3 where $H_K()$ is a non-linear operator that sets all but the top-k largest elements by their magnitude to zero and $\mu$ is computed adaptively as described in [1].

$$\hat{x}^{n+1} = H_K(\hat{x}^n + \mu A^\top (y - A\hat{x}^n)) \quad (3)$$

Pros:

- Gives theoretical guarantees on convergence to a local minimum of the cost function

- Doesn't depend on the scaling of the design matrix $A$

Cons:

- Theoretical guarantees on convergence exist only if $A$ that satisfies restricted isometry property

#### 4.2.2 Simultaneous Feature and Feature Group Selection through Hard Thresholding

The algorithm employs the iterative procedure 4 where $f$ is the objective loss function, $L$ is found by line search and $SGHT()$ stands for Sparse Group Hard Thresholding that is solved by dynamic programming as described in [4].

$$\hat{x}^{n+1} = SGHT(\hat{x}^n - \frac{1}{L}\nabla f(\hat{x}^n)) \tag{4}$$

Pros:

- Line search on each iteration can be significantly sped up
- Gives theoretical guarantees on convergence to a local minimum of the cost function that is at least within $c||y - Ax^*||_2$ radius from globally optimal solution $x^*$ for a certain constant $c$

Cons:

- Theoretical guarantees on convergence exist only if $A$ that satisfies restricted isometry property

## 4.3 Evaluation protocol

### 4.3.1 Metrics

To evaluate performance of different methods we plan to employ the protocol from [4]. Namely, we're going to report the number of selected features, feature groups (for ISTA with SGHT [4]), mean squared error and $R^2$ score.

We also plan to study the influence of the noise on the performance of each method. Following [1] we aim to generate data with a certain signal-to-noise ratio (SNR) and compare the estimation of SNR to an oracle to which the noise values are known.

### 4.3.2 Synthetic data

As proposed in [4] for generating synthetic data for examination of methods we're going to use the linear model $y = Ax + \epsilon$, where the design matrix $A \in R^{100\times 200}$ and the noise term $\epsilon$ follows normal distribution. Depending on the method ground truth $x$ is either partitioned into groups or is sparse. The goal is to obtain an accurate estimator of $x$ that preserves the sparse structure (and/or the grouping structure), given only $A$ and $y$.

#### 4.3.3 Real data

Motivated by [4] we intend to study the algorithms on the Boston Housing data set. The original data set is used as a regression task, containing 506 samples with 13 features. Up to third-degree polynomial expansion is applied on each feature to account for the non-linear relationship between variables and response. For each variable $x$, $x^2$ and $x^3$ are recorded and gathered in a group. As a next step we split the data into the training set (approximately 50%) and testing set. The parameter settings for each method are properly scaled to fit the data set. We intend to use a linear regression model for training and testing with the evaluation protocol described in subsection 4.3.1.

## 5 Evaluation results

We've conducted several tests to examine the properties of each method. We study how each method behaves when: there are non-linear dependencies between observations and features, high SNR and small number of actual informative features. The data generation procedure follows the one described in section 4.3.2.

In table 1 evaluation results for linear regression model with permutation importance (PI) and IHT feature selection are presented. The results of Test 1 (see 1) show that with high SNR PI gives good results both in MSE and $R^2$ as well as selects a number of features that is close to the ground-truth. IHT requires the true number of informative features to be provided so we do not report it. It can be seen that IHT gives significantly better results but outperforms oracle which means that it has also overfitted to the noise. If features are polynomial with the degree of 3 then MSE significantly increases as shown in Test 2 for PI, but for IHT there is no change at all. If the noise is too high and SNR is lower then 0 then although MSE and $R^2$ are both low the model is apparently over-fitting to the data as it performs significantly better than the oracle. The same holds true for IHT. In the case when the actual number of features is four times lower than the dimensionality of the feature space permutation importance fails to recover the number of informative features. IHT performs decently in this scenario.

To conclude, permutation importance works well when SNR is high, there is no non-linear dependencies between features and the number of informative features is not lower than 50%. When working with IHT it is better for the data to be noise-free or to employ some sort of regularization. Also, IHT needs the desired number of features to be provided.

## 6 Team members roles

In this research, we split responsibilities as follows:

- Viacheslav Pronin will implement the Normalized Iterative Hard Thresholding method and study the algorithms on the real data;

| | PI | IHT | PI | IHT | PI | IHT | PI | IHT |
|---|---|---|---|---|---|---|---|---|
| | Test 1 | | Test 2 | | Test 3 | | Test 4 | |
| SNR (dB) | 15.475 | | 21.145 | | -0.088 | | 13.303 | |
| Inform. feat. | 105 | | 105 | | 105 | | 56 | |
| Poly deg. | 1 | | 3 | | 1 | | 1 | |
| Oracle MSE | 1.055 | | 0.987 | | 37.992 | | 1.073 | |
| Oracle $R^2$ | 0.992 | | 0.998 | | 0.700 | | 0.984 | |
| Sel. feat. | 107 | - | 102 | - | 111 | - | 92 | - |
| MSE | 3.440 | 0.029 | 13.272 | 0.029 | 5.855 | 0.128 | 1.707 | 8.308 |
| $R^2$ | 0.972 | 1.000 | 0.968 | 1.000 | 0.960 | 0.992 | 0.976 | 0.882 |

Table 1: Evaluation on synthetic dataset with different SNR, number of informative features and polynomial degree of dependencies between features. For evaluation the number of selected features, MSE and $R^2$ are reported along with oracle's estimates of errors.

- Konstantin Pakulev will apply the Simultaneous Feature and Feature Group Selection through Hard Thresholding method and focus on the examination of algorithms on the synthetic data.

Both researchers will work on the metrics section and provide the evaluation of methods' accuracy, as well as conclusions and recommendations for further study.

## References

[1] T. Blumensath and M. E. Davies. Normalized iterative hard thresholding: Guaranteed stability and performance. *IEEE Journal of Selected Topics in Signal Processing*, 4(2):298–309, 2010.

[2] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[3] Lorena Bălan, Kiyohito Kunii (Kiyo), Dmitrii Deriabin, Lim Hoang, Andrii Ivaniuk, Yetunde Dada, Deepyaman Datta, Zain Patel, Gordon Wrigley, Ivan Danov, Jo Stichbury, Nasef Khan, Nikolaos Tsaousis, Merel Theisen, Waylon Walker, Tam Nguyen, Richard Westenra, Lais Carvalho, Marcelo Duarte Trevisani, Sebastian Bertoli, Shahil Mawjee, sasaki takeru, Bas Nijholt, Dmitry Vukolov, Kody Fischer, Vijaykumar, Yusuke Minami, bru5, and dr3s. quantumblacklabs/kedro: 0.17.0, December 2020.

[4] Shuo Xiang, Tao Yang, and Jieping Ye. Simultaneous feature and feature group selection through hard thresholding. In *Proceedings of the 20th ACM*

*SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, page 532–541, New York, NY, USA, 2014. Association for Computing Machinery.