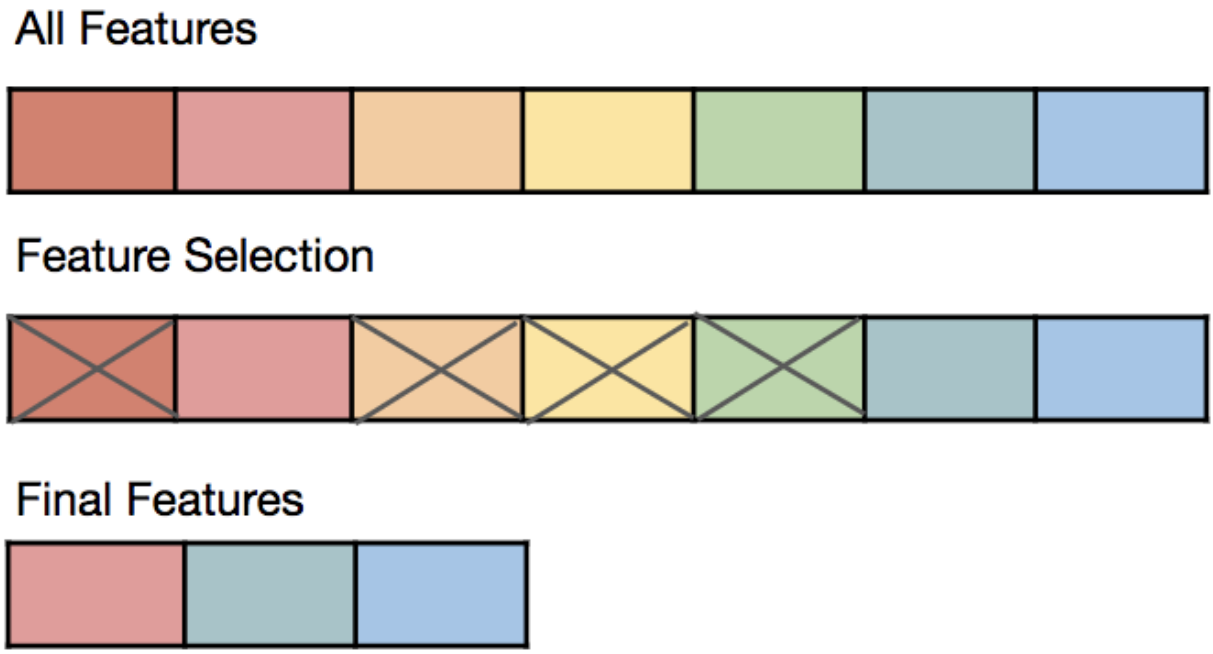
# Feature Selection for Machine Learning

python.plainenglish.io/automated-feature-selection-for-machine-learning-in-python-2ad4bcfac19a

Doruk Canga 23 August 2023

## Automated Feature Selection for Machine Learning in Python

Feature selection is the process of identifying the most important and informative features within a dataset. It is one of the most important steps of machine learning modeling pipeline, since it has significant impact on model performance and its predictive power.




Simple Visualization of Feature Selection

**Benefits of performing feature selection:**

- Improved model performance & reduced complexity ()
- Reduced training time
- Diminished risk of overfitting coming from uninformative & redundant features
- Simplified deployment processes and live data pipelines, an often underestimated advantage.

Despite its numerous benefits, feature selection can be overlooked during machine learning model development due to the tight deadlines in real-life projects and underestimation of its effects on performance. Also there are high number of feature selection methods and not knowing/deciding which one(s) to use may lead to the omission of this crucial step.

A potential solution to overcome time related challenges and to leverage the positive impact of feature selection on machine learning models is the implementation of reusable, partially or fully automated code.

## Automated Feature Selection with AutoFeatSelect

is a Python library designed to automate and accelerate feature selection processes for machine learning projects. This library automates various steps, such as calculating correlations, eliminating highly correlated features, and applying multiple feature selection methods and it presents all the outcomes.

The library offers users the flexibility to perform correlation analysis and apply feature selection methods either separately or collectively using just a few lines of code.

By doing so, helps with time and resource saving while improving model performance. The library is automated and simplified implemantation of following feature selection methods:

- Correlation analysis on continuous and categorical features
- Feature importance analysis using LightGBM, XGBoost, and Random Forest
- LassoCV Coefficients
- Permutation Importance
- Recursive Feature Elimination with Cross-Validation
- Boruta

## How to use: Applying ‘autofeatselect’ to a Real Dataset:

Before working on real datasets, just install the package with pip:

```
pip install autofeatselect
```

Let’s work on Porto Seguro’s Safe Driver Prediction dataset and implement all the features step by step. The package offers two distinct options for feature selection: applying the methods separately or utilizing the fully automated approach.

## Import Libraries

```
import pandas as pd
import numpy as np

 autofeatselect  CorrelationCalculator, FeatureSelector, AutoFeatureSelect
```

## Simple Data Preparation

```
#Preparing data
df = pd.read_csv("train.csv")
df.drop('id', axis=1, inplace=True)

#Setting numeric & categorical features for further analysis
response = 'target'
cat_feats = [c for c in df.columns if'_cat'in c]
bin_feats = [c for c in df.columns if'_bin'in c]
cat_feats = cat_feats + bin_feats
num_feats = [c for c in df.columns if c notin cat_feats+[response]]



df[num_feats] = df[num_feats].astype('float')
df[cat_feats] = df[cat_feats].astype('object')

df.replace(-1, np.nan, inplace=True)

X_train, X_test, y_train, y_test = train_test_split(df[num_feats+cat_feats],
df[response],                                                    test_size=,
random_state=)
```

## Approach 1: Applying Correlation Analysis and Feature Selection Methods Separately

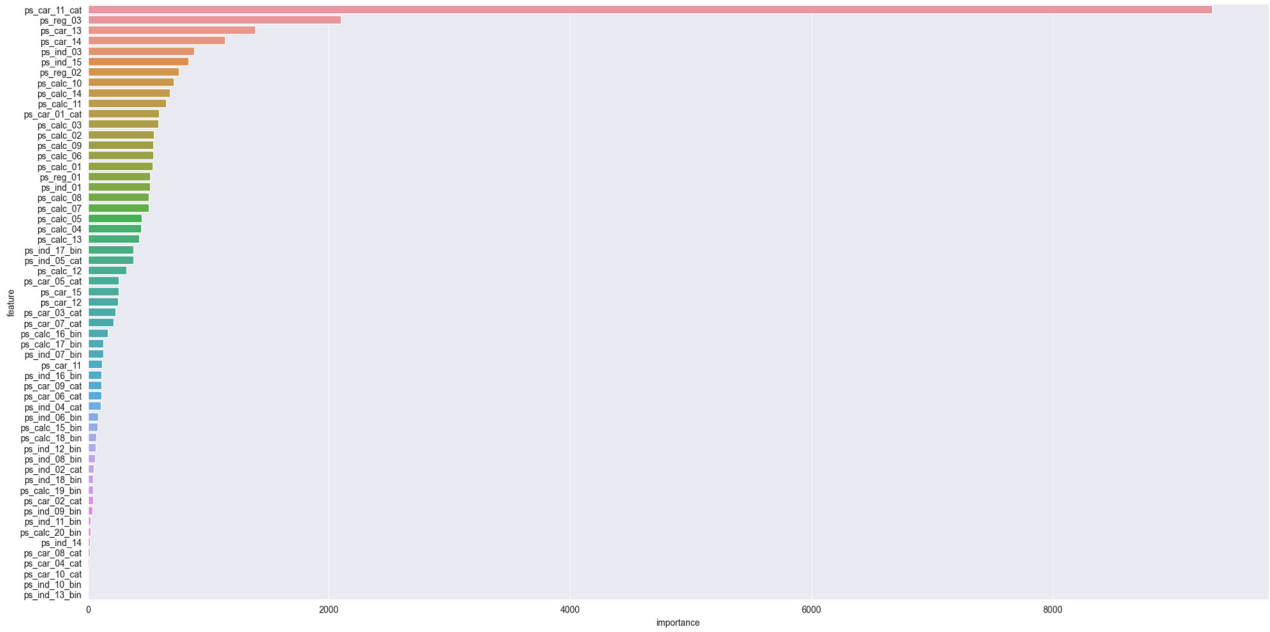
Since correlated features can have negative effects on feature importance and selection outcomes (*even though predictive performance of tree based ensemble models is not affected by correlations.*), high correlated features must be removed initially. They can be easily detected with and methods of class:

```
#Static features will not be removed even if they are correlated with other
features.
static_features = ['ps_ind_01', 'ps_ind_03','ps_ind_14']

#Detect correlated features
corr_df_num, num_remove_list = CorrelationCalculator.numeric_correlations(X_train,

features=num_feats,

static_features=static_features,

threshold=0.9)

corr_df_cat, cat_remove_list =
CorrelationCalculator.categorical_correlations(X_train,

features=cat_feats,

static_features=None,

threshold=0.9)

num_feats = [c  c  num_feats  c   num_remove_list]cat_feats = [c  c  cat_feats  c
cat_remove_list]
```

After removing correlated features from initial feature set, we can apply feature selection methods. Let's calculate LightGBM Feature Importance Scores with class:

```
#Create Feature Selector Object
feat_selector = FeatureSelector(modeling_type='classification',
                                X_train=X_train, y_train=y_train,
                                X_test=X_test, y_test=y_test,
                                numeric_columns=num_feats,
                                categorical_columns=cat_feats,
                                seed=24)

lgbm_importance_df = feat_selector.lgbm_importance(hyperparam_dict=,
objective=,                                                    return_plot=)
```

Resulting feature importance graph:




LightGBM Feature Importance Results

We can apply more selection methods separately with just calling their methods. Application of Recursive Feature Elimination after LightGBM:

```
rfecv_importance_df = feat_selector.rfecv_importance(lgbm_hyperparams=,
rfecv_hyperparams=,
return_plot=)
```

| | feature | importance |
|---|---|---|
| 0 | ps_ind_03 | 1 |
| 1 | ps_car_11_cat | 1 |
| 2 | ps_ind_15 | 1 |
| 3 | ps_car_01_cat | 1 |
| 4 | ps_reg_02 | 1 |
| ... | ... | ... |
| 52 | ps_ind_11_bin | 41 |
| 53 | ps_ind_14 | 42 |
| 54 | ps_car_10_cat | 43 |
| 55 | ps_ind_13_bin | 44 |
| 56 | ps_ind_10_bin | 45 |

RFECV Rankings Table

## Approach 2: Fully Automated Approach

Instead of removing correlated features and applying feature selection methods one by one, the entire process can be significantly automated through the functionality which offers more efficient and comprehensive way to handle feature selection.

```
#Create AutoFeatureSelect class
feat_selector = AutoFeatureSelect(modeling_type='classification',
                                  X_train=X_train,
                                  y_train=y_train,
                                  X_test=X_test,
                                  y_test=y_test,
                                  numeric_columns=num_feats,
                                  categorical_columns=cat_feats,
                                  seed=24)

#Detect Correlated Features
corr_features = feat_selector.calculate_correlated_features(static_features=None,
                                                            num_threshold=0.9,
                                                            cat_threshold=0.9)
#Drop Correlated Features
feat_selector.drop_correlated_features()

#Determine Selection Methods to Apply
#Options: 'lgbm', 'xgb', 'rf','perimp', 'rfecv', 'boruta', 'lassocv'
#Note: Hyperparameters of all methods can be changed
selection_methods = ['lgbm', 'xgb', 'rf','perimp', 'rfecv', 'boruta']
final_importance_df =
feat_selector.apply_feature_selection(selection_methods=selection_methods,
                                                            lgbm_hyperparams=None,
                                                            xgb_hyperparams=None,
                                                            rf_hyperparams=None,


lassocv_hyperparams=None,

perimp_hyperparams=None,

rfecv_hyperparams=None,

boruta_hyperparams=None)

final_importance_df.head()
```

With calling just 4 functions and writing few lines of code, it is possible to apply several feature selection methods. Resulting dataframe will contain all the results and look like below:

| | feature | lgbm_importance | xgb_importance | rf_importance | permutation_importance | rfecv_rankings | boruta_support | boruta_support_weak | boruta_ranking |
|---|---|---|---|---|---|---|---|---|---|
| 0 | ps_car_11_cat | 9325.593919 | 0.070897 | 0.090008 | 0.000134 | 1 | 1 | 0 | 1 |
| 1 | ps_reg_03 | 2098.823395 | 0.103508 | 0.055653 | 0.000000 | 1 | 0 | 1 | 2 |
| 2 | ps_car_13 | 1387.502321 | 0.091939 | 0.082182 | 0.000101 | 1 | 1 | 0 | 1 |
| 3 | ps_car_14 | 1137.281807 | 0.067428 | 0.028448 | 0.000034 | 1 | 0 | 0 | 5 |
| 4 | ps_ind_03 | 882.397635 | 0.034633 | 0.023691 | -0.000118 | 1 | 0 | 0 | 12 |

After having these results, it is possible to select final feature subset by using results of one or more method. Also, by using combination set or intersection set of top N features of several methods, better feature set can be achieved.

## Conclusion

Feature selection is one of the most important steps of machine learning modeling, but it can be underestimated or overlooked for various reasons. To avoid this and to ensure more efficient feature selection, automated or semi-automated solutions can be used. package enables the application of multiple methods with just a few lines of code and allows us to work more efficiently on real-life data science projects.

Anyone can contribute to the project on GitHub and all feedbacks and criticism are welcomed on LinkedIn.

## In Plain English

*Thank you for being a part of our community! Before you go:*