Member-only story

5 Feature Selection Method from Scikit-Learn you should know

Improve your performance by selecting only the most important features

Cornellius Yudha Wijaya · Follow

Published in Towards Data Science · 8 min read · Mar 8, 2021

Photo by Clay Banks on Unsplash

Feature selection is a method to reduce the variables by using certain criteria to select variables that are most useful to predict the target by our model.

Increasing the number of features would help the model to have a good prediction power, but only until a certain point. This is what we called a **Curse of Dimensionality**, where the model's performance would increase with the higher number of features we used. Still, the performance will

deteriorate when the feature number is past the peak. That is why we need to select only the features that are effectively capable of prediction.

Feature selection is similar to the **dimensionality reduction technique,** where the aim is to reduce the number of the features, but fundamentally they are different. The difference is that feature selection selects features to keep or remove from the dataset, whereas dimensionality reduction creates a projection of the data resulting in entirely new input features. If you want to know more about dimensionality reduction, you could check other articles I have below.

5 Must-Know Dimensionality Reduction Techniques via Prince

Reducing features is good for your data science project

towardsdatascience.com

Feature Selection has many methods, but I only would show 5 feature selections present in the Scikit-Learn. I limit only the one that is present in the scikit-learn as it is the easiest yet most useful. Let's get into it.

1. Variance Threshold Feature Selection

A feature with a higher variance means that the value within that feature varies or has a high cardinality. On the other hand, lower variance means the value within the feature is similar, and zero variance means you have a feature with the same value.

Intuitively, you want to have a varied feature as we don't want our predictive model to be biased. That is why we could select the feature based on the variance we select previously.

A variance Threshold is a simple approach to eliminating features based on our expected variance within each feature. Although, there are some down-side to the Variance Threshold method. The Variance Threshold feature selection only sees the input features (X) without considering any information from the dependent variable (y). It is only useful for eliminating features for **Unsupervised Modelling** rather than Supervised Modelling.

Let's try it with an example dataset.

```
import pandas as pd
import seaborn as sns
```

```
mpg = sns.load_dataset('mpg').select_dtypes('number')
mpg.head()
```

mpg	cylinders	displacement	horsepower	weight	acceleration	model_year
18.0	8	307.0	130.0	3504	12.0	70
15.0	8	350.0	165.0	3693	11.5	70
18.0	8	318.0	150.0	3436	11.0	70
16.0	8	304.0	150.0	3433	12.0	70
17.0	8	302.0	140.0	3449	10.5	70

Image created by Author

For this example, I only use numerical features for simplicity purposes. We need to transform all of these numerical features before we use the Variance Threshold Feature Selection as the variance is affected by the numerical scale.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
mpg = pd.DataFrame(scaler.fit_transform(mpg), columns = mpg.columns)
mpg.head()
```

mpg	cylinders	displacement	horsepower	weight	acceleration	model_year
-0.706439	1.498191	1.090604	0.664133	0.630870	-1.295498	-1.627426
-1.090751	1.498191	1.503514	1.574594	0.854333	-1.477038	-1.627426
-0.706439	1.498191	1.196232	1.184397	0.550470	-1.658577	-1.627426
-0.962647	1.498191	1.061796	1.184397	0.546923	-1.295498	-1.627426
-0.834543	1.498191	1.042591	0.924265	0.565841	-1.840117	-1.627426

Image created by Author

With all the features on the same scale, let's try to select only the features we want using the Variance Threshold method. Let's say I limit my variance to one.

```
from sklearn.feature_selection import VarianceThreshold
selector = VarianceThreshold(1)
selector.fit(mpg)
mpg.columns[selector.get_support()]
```

```
Index(['weight'], dtype='object')
```

Image created by Author

It seems only the weight feature are selected based on our Variance Threshold we set.

As I said before, the Variance Threshold only useful when we consider the feature selection for Unsupervised Learning. What if we want the feature selection for Supervised Learning purposes? That is what we gonna talk about next. Most of the feature selections from the Scikit-Learn are useful for Supervised Learning, after all.

2. Univariate Feature Selection with SelectKBest

Univariate Feature Selection is a feature selection method based on the univariate statistical test, e,g: chi2, Pearson-correlation, and many more.

The premise with **SelectKBest is combining the univariate statistical test with selecting the K-number of features based on the statistical result** between the X and y.

Let's use it with an example data we have before.

```
mpg = sns.load_dataset('mpg')
mpg = mpg.select_dtypes('number').dropna()

#Divide the features into Independent and Dependent Variable
X = mpg.drop('mpg' , axis =1)
y = mpg['mpg']
```

Because the univariate feature selection method is intended for Supervised Learning, we divide the features into independent and dependent variables. Next, we would select the features using SelectKBest based on the mutual info regression. Let's say I only want the top two features.

```
from sklearn.feature_selection import SelectKBest,
mutual_info_regression

#Select top 2 features based on mutual info regression
selector = SelectKBest(mutual_info_regression, k =2)
selector.fit(X, y)
X.columns[selector.get_support()]
```

```
Index(['displacement', 'weight'], dtype='object')
```

Image created by Author

Based on the mutual info regression, we only select the 'displacement,' and 'weight' features like these are the top features.

3. Recursive Feature Elimination (RFE)

Recursive Feature Elimination or RFE is a Feature Selection method **utilizing a machine learning model to selecting the features by eliminating the least important feature after recursively training.**

According to Scikit-Learn, RFE is a method 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 a `coef_` attribute or through a `feature_importances_` attribute. Then, the least important features are pruned from the current set of features. That procedure is recursively repeated on the pruned set until the desired number of features to select is eventually reached.

tl;dr RFE selects top k features based on the machine learning model that has `coef_` attribute or `feature_importances_` attribute from their model

(Almost any model). RFE would eliminate the least important features then retrain the model until it only selects the K-features you want.

This method only works if the model has `coef_` or `features_importances_` attribute, if there are models out there having these attributes, you could apply RFE on Scikit-Learn.

Let's use a dataset example. In this sample, I want to use the titanic dataset for the classification problem, where I want to predict who would survive.

```
#Load the dataset and only selecting the numerical features for
example purposes
titanic = sns.load_dataset('titanic')[['survived', 'pclass', 'age',
'parch', 'sibsp', 'fare']].dropna()
X = titanic.drop('survived', axis = 1)
y = titanic['survived']
```

I want to see which features are the best to help me predict who would survive the titanic incident using RFE. Let's use the LogisticRegression model to obtain the best features.

```
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
```

```
# #Selecting the Best important features according to Logistic
Regression

rfe_selector =
RFE(estimator=LogisticRegression(),n_features_to_select = 2, step =
1)
rfe_selector.fit(X, y)
X.columns[rfe_selector.get_support()]
```

```
Index(['pclass', 'parch'], dtype='object')
```

Image created by Author

By default, the number of features selected for RFE is the median of the total features, and the step (the number of features eliminated each iteration) is one. You could change it based on your knowledge or the metrics you used.

4. Feature Selection via SelectFromModel

Like the RFE, SelectFromModel from Scikit-Learn is based on a Machine Learning Model estimation for selecting the features. The differences are that **SelectFromModel feature selection is based on the importance attribute** (often is `coef_` or `feature_importances_` but it could be any callable) **threshold**. By default, the threshold is the mean.

Let's use a dataset example to understand the concept better. I would use the same titanic data we have before.

```
from sklearn.feature_selection import SelectFromModel

# #Selecting the Best important features according to Logistic
Regression using SelectFromModel

sfm_selector = SelectFromModel(estimator=LogisticRegression())
sfm_selector.fit(X, y)
X.columns[sfm_selector.get_support()]
```

```
Index(['pclass'], dtype='object')
```

Image created by Author

Using SelectFromModel, we found out that only one feature passed the threshold: the 'pclass' feature.

Like RFE, you could use any Machine Learning model to select the feature, as long as it was callable to estimate the attribute importance. You could try it out with the Random Forest model or XGBoost.

5. Feature Selection Sequential Feature Selection (SFS)

New in the Scikit-Learn Version 0.24, Sequential Feature Selection or SFS is a greedy algorithm to find the best features by either going forward or backward based on the **cross-validation score an estimator.**

According to Scikit-Learn, **SFS-Forward** made a feature selection by starting with zero feature and find the one feature that maximizes a cross-validated score when a machine learning model is trained on this single feature. Once that first feature is selected, the procedure is repeated by adding a new feature to selected features. The procedure is stopped when we find the desired number of features is reached.

SFS-Backward follows the same idea but works in the opposite direction: It starts with all the features and greedily removes all the features until it reached the desired number of features.

SFS differs from RFE and SelectFromModel because the machine learning model did not need the `coef_` or `feature_importances_` attribute. Although, it is considerably slower as it evaluated the result by fitting the model multiple times.

Let's try it with an example. I want to try SFS-Backward for an example.

```
from sklearn.feature_selection import SequentialFeatureSelector

#Selecting the Best important features according to Logistic
Regression

sfs_selector =
SequentialFeatureSelector(estimator=LogisticRegression(),
n_features_to_select = 3, cv =10, direction ='backward')
sfs_selector.fit(X, y)
X.columns[sfs_selector.get_support()]
```

```
Index(['pclass', 'age', 'parch'], dtype='object')
```

Image created by Author

Using the SFS-Backward with three features to select and ten cross-validations, we end up with ‘pclass’, ‘age’, and ‘parch’ features.

You could experiment with this feature selection and see how your model performance is, but remember that with a higher number of features and data, your selection time would getting higher as well.

Conclusion

Feature Selection is an important aspect of the machine learning model, as we did not want to have many features that did not affect our predictions model at all.

In this article, I have shown you 5 Scikit-Learn Feature Selection methods you could use, they are:

1. Variance Threshold Feature Selection
2. Univariate Selection using SelectKBest
3. Recursive Feature Elimination or RFE
4. SelectFromModel
5. Sequential Feature Selection or SFS

I hope it helps!

Visit me on my **Social Media**.

If you are not subscribed as a Medium Member, please consider subscribing through my referral.

Data Science | Education | Artificial Intelligence | Technology | Towards Data Science

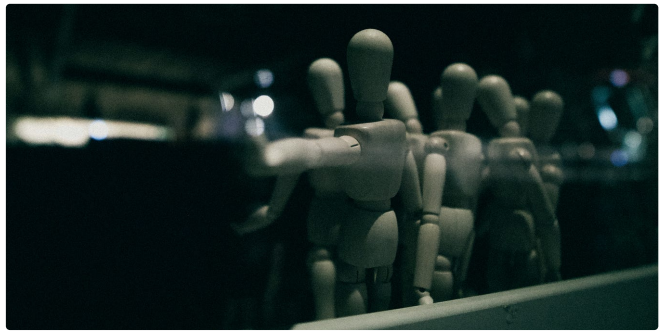
Written by Cornellius Yudha Wijaya

3.9K Followers · Writer for Towards Data Science

2.1M+ Views |Top 1000 Writer | LinkedIn: Cornellius Yudha Wijaya | Twitter:@CorneliusYW

Follow

More from Cornellius Yudha Wijaya and Towards Data Science



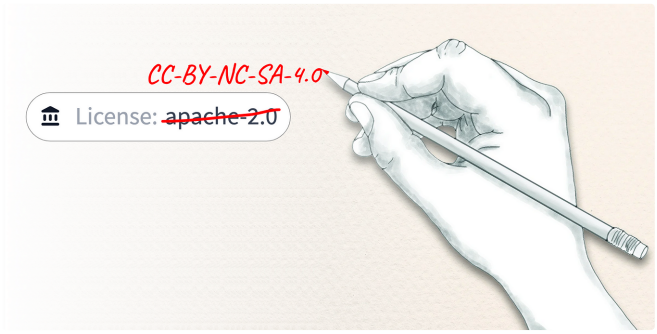
Cornellius Yudha Wijaya in DataDrivenInvestor

Measurement of Social Bias Fairness Metrics in NLP Models

Understand the metrics to mitigate bias in text models.

✨ · 14 min read · Jun 7

204 2



Clemens Mewald in Towards Data Science

The Golden Age of Open Source in AI Is Coming to an End

NC, SA, GPL, and other acronyms you don't want to see in the open source license of the...

7 min read · Jun 7

919 14

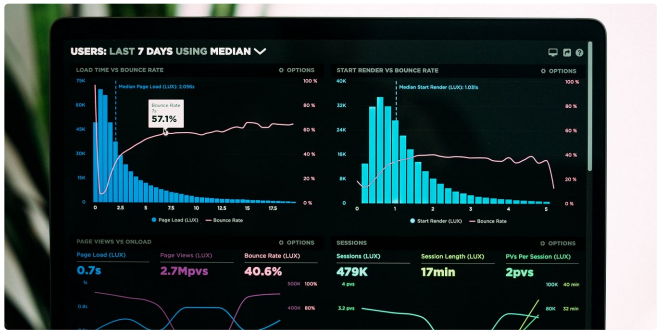
Luís Roque in Towards Data Science

Harnessing the Falcon 40B Model, the Most Powerful Open-Source...

Mastering open-source language models: diving into Falcon-40B

✦ · 12 min read · Jun 9

243 4



Cornellius Yudha Wijaya in Towards Data Science

4 Python Packages to Create Interactive Dashboards

Use these packages to improve your data science project

✦ · 7 min read · May 27, 2022

399 5

See all from Cornellius Yudha Wijaya

See all from Towards Data Science

Recommended from Medium

Andras Gefferth in Towards Data Science

One Hot Encoding scikit vs pandas

You can safely use pandas.get_dummies for machine learning applications, just need to ...

8 min read · Mar 13

13

RITHP

Logistic Regression for Feature Selection: Selecting the Right...

Logistic regression is a popular classification algorithm that is commonly used for feature...

4 min read · Jan 1

16

Lists

ChatGPT prompts

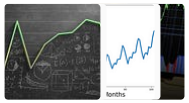
17 stories · 6 saves

ChatGPT

18 stories · 3 saves

AI Regulation

6 stories · 1 save



Predictive Modeling w/ Python

18 stories · 12 saves

Sean Knight

Intro to Data Scaling and Normalization in Python

If you’re looking to dive into the world of data analysis and machine learning, you’ve...

✨ · 3 min read · Feb 28

3

Xinqian Zhai in MLearning.ai

Create ColumnTransformer & FeatureUnion in Pipelines with...

In this post, we’ll show how ColumnTransformer & FeatureUnion differ,...

5 min read · Jan 30

105

Rajneesh Tiwari in CueNex

Advanced Evaluation Metrics for Imbalanced Classification Models

Measure Imbalanced Models the right way!

8 min read · Feb 9

109 2

Serafeim Loukas, PhD in Towards AI

How To Use The LazyPredict Python Library To Select The Best...

In this article, I introduce a nice python library, Lazy Predict. Build basic models without...

· 7 min read · Feb 15

129 3

See more recommendations