Table of Contents:

for more info: https://en.wikipedia.org/wiki/Feature_selection#Main_principles (https://en.wikipedia.org/wiki/Feature_selection#Main_principles)

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

Importance of Feature Selection in Machine Learning

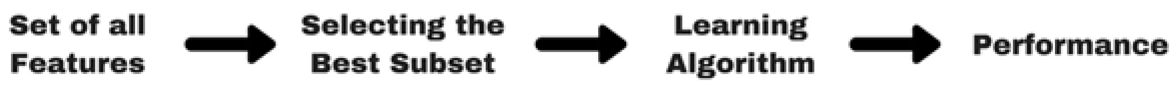
• It enables the machine learning algorithm to train faster.

• It reduces the complexity of a model and makes it easier to interpret.

• It improves the accuracy of a model if the right subset is chosen.

• It reduces overfitting.

1. Filter Methods



Filter methods are generally used as a preprocessing step. The selection of features is independent of any machine learning algorithms. Instead, features are selected on the basis of their scores in various statistical tests for their correlation with the outcome variable.

Filter methods can be broadly categorized into two categories:

1.1 Univariate Filter Methods and ¶

1.2 Multivariate filter methods.

1.1 Univariate Methods

The univariate filter methods are the type of methods where individual features are ranked according to specific criteria. The top N features are then selected.

Univariate filter methods are ideal for removing constant and quasi-constant features from the data.

One of the major disadvantage of univariate filter methods is that they may select redundant features because the relationship between individual features is not taken into account while making decisions.

Removing Constants

In [1]:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import VarianceThreshold
```

Dataset link:

https://www.kaggle.com/c/santander-customer-satisfaction/data (https://www.kaggle.com/c/santander-customer-satisfaction/data)

In [2]:

```
santandar_data = pd.read_csv("train.csv",nrows = 40000)
```

In [4]:

```
santandar_data.shape
```

Out[4]:

```
(40000, 371)
```

In [16]:

```
train_features , test_features , train_labels , test_labels = train_test_split(
santandar_data.drop(labels=['TARGET'] , axis =1),
santandar_data['TARGET'],
test_size = 0.2,
random_state = 41)
train_features.shape , test_features.shape
```

Out[16]:

```
((32000, 370), (8000, 370))
```

To remove constants we will use VarianceThreshold function that we imported earlier. The function requires a value for its threshold parameter. Passing a value of zero for the parameter will filter all the features with zero variance.

Because constant features have values with zero variance since all the values are the same.

In [8]:

```
constant_filter = VarianceThreshold(threshold = 0)
```

In [9]:

```
constant_filter.fit(train_features)
```

Out[9]:

```
VarianceThreshold(threshold=0)
```

In [13]:

```
# no:of features that are not constants
len(train_features.columns[constant_filter.get_support()])
```

Out[13]:

```
320
```

In [14]:

```
constant_columns = [column for column in train_features.columns
                    if column not in train_features.columns[constant_filter.get_support(
print(len(constant_columns))
```

```
50
```

Finally to remove constant features , we can use the transform() method of the constant_filter.

In [15]:

```
train_features = constant_filter.transform(train_features)
test_features = constant_filter.transform(test_features)

train_features.shape , test_features.shape
```

Out[15]:

```
((32000, 320), (8000, 320))
```

Removing Quasi Constants

Instead of passing 0 as the value for the threshold parameter, we will pass 0.01, which means that if the variance of the values in a column is less than 0.01, remove that column. In other words, remove feature column where approximately 99% of the values are similar

In [1]:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import VarianceThreshold

santandar_data = pd.read_csv("train.csv", nrows=40000)
santandar_data.shape
```

Out[1]:

```
(40000, 371)
```

In [2]:

```
train_features, test_features, train_labels, test_labels = train_test_split(
    santandar_data.drop(labels=['TARGET'], axis=1),
    santandar_data['TARGET'],
    test_size=0.2,
    random_state=41)
```

In [3]:

```
const_filter = VarianceThreshold(threshold = 0)
const_filter.fit(train_features)
len(train_features.columns[const_filter.get_support()])
```

Out[3]:

```
320
```

In [4]:

```
const_colomns = [column for column in train_features.columns
                if column not in train_features.columns[const_filter.get_support()]
                ]
train_features.drop(labels = const_colomns , axis =1 , inplace = True)
test_features.drop(labels = const_colomns, axis =1 , inplace = True)
```

In [5]:

```
qconstant_filter = VarianceThreshold(threshold=0.01)
```

In [6]:

```
qconstant_filter.fit(train_features)
```

Out[6]:

```
VarianceThreshold(threshold=0.01)
```

In [7]:

```
len(train_features.columns[qconstant_filter.get_support()])
```

Out[7]:

```
265
```

In [8]:

```
qconstant_columns = [column for column in train_features.columns
                    if column not in train_features.columns[qconstant_filter.get_suppor

print(len(qconstant_columns))
```

```
55
```

In [9]:

```
train_features = qconstant_filter.transform(train_features)
test_features = qconstant_filter.transform(test_features)

train_features.shape, test_features.shape
```

Out[9]:

```
((32000, 265), (8000, 265))
```

In [ ]:

```

```

LINKS FOR THE BUILT-IN METHODS DOCCUMENTATIONS USED IN ABOVE PROGRAM

read_csv() : [https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_csv.html (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_csv.html)](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_csv.html)

train_test_split() : [https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html (https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html)](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html)

VarianceThreshold() : [https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.VarianceThreshold.html (https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.VarianceThreshold.html)](https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.VarianceThreshold.html)

transform() : [https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.transform.html (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.transform.html)](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.transform.html)

fit() : [https://scikit-learn.org/stable/modules/generated/sklearn.svm.libsvm.fit.html (https://scikit-learn.org/stable/modules/generated/sklearn.svm.libsvm.fit.html)](https://scikit-learn.org/stable/modules/generated/sklearn.svm.libsvm.fit.html)