Feature Selection- Variance Threshold

Estimated time needed: **20** minutes

Objectives

After completing this lab you will have a good understanding in:

- Imporatance of Columns while Modelling
- Variance Value
- Why Constant Values are not important

This Notebook is created for Python Module

Feature Selection

It is almost never the case that all of the variables in a dataset can be used to develop a machine learning model in practise. Repetitive factors decrease a classifier's capacity to generalise and may also lower the classifier's overall accuracy. Additionally, a model's total complexity rises as more variables are added to it. The goal of feature selection in machine learning is to find the best set of features that allows one to build useful models of studied phenomena.

Dropping constant features

In this step we will be removing the features which have constant features which are actually not important for solving the problem statement

In [1]:
```
# Import pandas to create DataFrame
import pandas as pd

# Make DataFrame of the given data
data = pd.DataFrame({"A":[1,2,4,1,2,4],
                     "B":[4,5,6,7,8,9],
                     "C":[-0.2,0,0,0,0,0],
                     "D":[1,1,1,1,1,1]})
```

In [2]:
```
data.head()
```

Out[2]:

	A	B	C	D
0	1	4	-0.2	1
1	2	5	0.0	1
2	4	6	0.0	1
3	1	7	0.0	1
4	2	8	0.0	1

In [3]:
```
data.var()
```

Out[3]:
```
A    1.866667
B    3.500000
C    0.006667
D    0.000000
dtype: float64
```

Variance Threshold

A basic method for choosing features is the variance threshold. All features whose variance falls below a certain level are removed from data. It eliminates all zero-variance features—that is, features whose value is the same across all samples—by default. Although one of the limitations of filter methods is that they do not take into account the relationship between feature variables or between feature and target variables, we presume that features with a higher variance may contain more important information.

[Note]: Feature selector that removes all low-variance features. This feature selection algorithm looks only at the `features (X)` , not the desired `outputs (y)` and can thus be used for unsupervised learning..

In [4]:
```
### It will check for cols with var 2.0
from sklearn.feature_selection import VarianceThreshold
var_thres=VarianceThreshold(threshold=2.0)
var_thres.fit(data)
```

Out[4]:
```
VarianceThreshold(threshold=2.0)
```

[Note]: The `get_support` returns a Boolean vector where True means that the variable does not have zero variance.

In [48]:
```
data.columns, var_thres.get_support()
```

Out[48]:
```
(Index(['A', 'B', 'C', 'D'], dtype='object'),
 array([False,  True, False, False]))
```

In [49]:
```
data.columns[var_thres.get_support()]
```

Out[49]:
```
Index(['B'], dtype='object')
```

In [9]:
```
constant_columns = [column for column in data.columns
                    if column not in data.columns[var_thres.get_support()]]

print(len(constant_columns))
```

```
3
```

In [7]:
```
for feature in constant_columns:
     print(feature)
```

```
A
C
D
```

In [10]:
```
data.drop(constant_columns,axis=1, inplace=True)
```

In [11]:
```
data
```

Out[11]:

	B
0	4
1	5
2	6
3	7
4	8
5	9

Lets practise on bigger dataset: Satander

Background

At Santander our mission is to help people and businesses prosper. We are always looking for ways to help our customers understand their financial health and identify which products and services might help them achieve their monetary goals.

Our data science team is continually challenging our machine learning algorithms, working with the global data science community to make sure we can more accurately identify new ways to solve our most common challenge, binary classification problems such as: is a customer satisfied? Will a customer buy this product? Can a customer pay this loan?

In this challenge, we invite Kagglers to help us identify which customers will make a specific transaction in the future, irrespective of the amount of money transacted. The data provided for this competition has the same structure as the real data we have available to solve this problem.

In [5]:
```
import pandas as pd
from sklearn.feature_selection import VarianceThreshold
```

In [19]:
```
df = pd.read_csv('train.csv', nrows=10000) # readin only 10k row to avoid loading time
```

In [20]:
```
df.shape
```

Out[20]:
```
(10000, 371)
```

In [27]:
```
df['TARGET'].values # your end goal id to predict this target values
```

Out[27]:
```
array([0, 0, 0, ..., 0, 0, 0])
```

In [28]:
```
X=data.drop(labels=['TARGET'], axis=1)
y=data['TARGET']
```

In [29]:
```
from sklearn.model_selection import train_test_split
# separate dataset into train and test
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_state=0)

X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[29]:
```
((7000, 370), (3000, 370), (7000,), (3000,))
```

Lets apply the variance threshold

In [30]:
```
var_thres=VarianceThreshold(threshold=0)
var_thres.fit(X_train)
```

Out[30]:
```
VarianceThreshold(threshold=0)
```

In [32]:
```
var_thres.get_support()
# The get_support returns a Boolean vector where True means that the variable does not have zero variance.
```

Out[32]:
```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True, False, False,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True, False, False,  True,  True,  True,  True,  True, False,
       False,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True, False, False, False, False,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
       False, False,  True,  True,  True,  True,  True,  True,  True,
       False,  True,  True,  True, False, False,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True, False, False,  True,  True,  True,
        True,  True, False, False,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True, False, False, False,
       False,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True, False, False,  True,  True,  True,  True,  True,
        True,  True,  True, False,  True,  True,  True,  True,  True,
       False, False,  True,  True,  True,  True,  True,  True,  True,
       False,  True,  True,  True, False,  True,  True,  True,  True,
       False, False,  True,  True,  True,  True,  True, False,  True,
        True, False,  True,  True, False,  True, False, False,  True,
        True,  True,  True,  True, False, False,  True, False,  True,
       False,  True,  True,  True,  True,  True,  True, False, False,
        True, False,  True, False,  True, False, False, False, False,
        True,  True,  True,  True,  True,  True, False,  True,  True,
        True, False,  True, False, False, False, False,  True,  True,
       False,  True, False, False, False,  True, False, False, False,
        True, False,  True,  True,  True,  True,  True,  True,  True,
        True, False, False,  True,  True,  True,  True,  True,  True,
       False,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True, False,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True, False,  True,
       False, False, False, False,  True,  True,  True,  True, False,
        True, False, False, False,  True, False, False, False,  True,
        True,  True,  True,  True, False,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True, False, False,
       False, False,  True,  True,  True,  True, False, False,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True])
```

In [41]:
```
### Finding non constant features -- no variance features
sum(var_thres.get_support()), (df.shape[1]-1)-sum(var_thres.get_support())
```

Out[41]:
```
(284, 86)
```

In [38]:
```
# Lets Find non-constant features
len(X_train.columns[var_thres.get_support()])
```

Out[38]:
```
284
```

In [34]:
```
constant_columns = [column for column in X_train.columns
                    if column not in X_train.columns[var_thres.get_support()]]

print(len(constant_columns))
```

86

In [39]:
```
less_varcol = [column for column in constant_columns]
print(less_varcol[:50]) # printing 1st 50 cols
```

```
['ind_var2_0', 'ind_var2', 'ind_var13_medio_0', 'ind_var13_medio', 'ind_var18_0', 'ind_var18', 'ind_var27_0
', 'ind_var28_0', 'ind_var28', 'ind_var27', 'ind_var34_0', 'ind_var34', 'ind_var41', 'ind_var46_0', 'ind_var
46', 'num_var13_medio_0', 'num_var13_medio', 'num_var18_0', 'num_var18', 'num_var27_0', 'num_var28_0', 'num_
var28', 'num_var27', 'num_var34_0', 'num_var34', 'num_var41', 'num_var46_0', 'num_var46', 'saldo_var13_medio
', 'saldo_var18', 'saldo_var28', 'saldo_var27', 'saldo_var34', 'saldo_var41', 'saldo_var46', 'delta_imp_amor
t_var18_1y3', 'delta_imp_amort_var34_1y3', 'delta_imp_reemb_var17_1y3', 'delta_imp_reemb_var33_1y3', 'delta_
imp_trasp_var17_out_1y3', 'delta_imp_trasp_var33_out_1y3', 'delta_num_reemb_var17_1y3', 'delta_num_reemb_var
33_1y3', 'delta_num_trasp_var17_out_1y3', 'delta_num_trasp_var33_out_1y3', 'imp_amort_var18_hace3', 'imp_amo
rt_var18_ult1', 'imp_amort_var34_hace3', 'imp_amort_var34_ult1', 'imp_var7_emit_ult1']
```

Now Let's Remove these no variance cols

In [42]:
```
novar = X_train.drop(constant_columns,axis=1)
novar.head()
```

Out[42]:

	ID	var3	var15	imp_ent_var16_ult1	imp_op_var39_comer_ult1	imp_op_var39_comer_ult3	imp_op_var40_comer_ult1	imp_
7681	15431	2	42	840.0	4477.02	4989.54	0.0	
9031	18181	2	31	0.0	52.32	52.32	0.0	
3691	7411	2	51	0.0	0.00	0.00	0.0	
202	407	2	36	0.0	0.00	0.00	0.0	
5625	11280	2	23	0.0	0.00	0.00	0.0	

5 rows × 284 columns

We will export this dataframe for Correltion Test. Don't forget to remove index while exporting.

```
novar.to_csv('', index = False)
```

In [43]:
```
novar.to_csv('var-pass.csv', index=False)
```

Practice Session

Find out the irrevelant coluumns on the dataset https://www.kaggle.com/datasets/iabhishekofficial/mobile-price-classification?select=train.csv

```
df=pd.read_csv('train.csv',nrows=10000)
```

Great Job!

That's all we need to know for now! Congratulations, you have learnt one more topic and hands-on with Python.This Notebook is prepared by Sumit Kumar Shukla IBM ICE.

About Author

Mr. Sumit is a Subject Matter Expert at IBM, and a data Scientist with more than five years of experience tutoring students from IITs, NITs, IISc, IIMs, and other prestigious institutions. Google Data Studio certified and IBM certified data analyst Data Science, Machine Learning Models, Graph Databases, and Data Mining techniques for Predictive Modeling and Analytics, as well as data integration, require expertise in Machine Learning and programming languages such as Python, R, and Tableau.