Feature Selection With Filtering Method |Constant , Quasi constant and Duplicate Feature Removal

- Machine learning and deep learning is learning from data which consist of diffrent types of features.
- lets say if we have huge datasets it could have huge featurespace as well.if we have huge feature space then we want some types of algorithms which can select the highest performing features for apping machine learning algorithms otherwise u will endup your machine with overfitting which will deterioate the performace of your model.
- Training time and Performace of our machine learning depends havily on the features in the datasets.
- If we having the large datasets then it will definately take long traning time.
- Its is garanteed that huge feature space will result the better performace.
- So idealy we have to select the those feature which are actually help in machine learning model learn something usefull.

- Unnecessary and redudant features not only slow down the traing time of our algorithms but they are also affect the overall performace of the algorithm.
- The process of selecting the most suitable feature for training the machine learning model is called a feature selection.
- In that we are going to learn about the feature selection with the filtering method.In that we will be do the constant,quasi constant and duplicate feature removal.

There is several advantages of performing the feature selection before taring our machine learning model.

- Models with the less numbers of featutres have higher explainability.
- Its very easier to implement the machine lerning models with the reduced features.
- Fewer feature leads to enhance the generalization which in turns reduces overfitting.
- Feature selection reduces data redudancy.
- Training time for a model with fewer features is significantly lower.
- Models with the fewer features are less prone to errors.

What is Filter Technique ?

- Filter method is belonging to the feature selection method that select features independently of machine learning models.
- It doesn't involve any machine learning alorithm while selecting a features.
- This is one of the biggest advantage of this method since is doesnt inovolve any machine learning algorithm that's it is a pretty fast.
- Feature selected by using with the filter method can be use as an input to any machine learning model.
- And another advantages of the fiter method is they are very fast.
- Filter method is a generally first step in feature selection pipeline so this kind of best screening method to get subset of the features.

There are two kind of feature seection method in filtering method those are the

- Univariate
- MultiVariant

Univariant

- In the univariate we are selecting the feature without taking consideration of other features.Thats it kind of independed.
- Lets consider we having the 10 features then we can select the some subset of the features without taking the information of nine features for particular feature.
- In the univariant feature selection we can select the features based on the Fisher score,Mutual Information Gain etc.

- The univariate filter methods are types of the method where individual features are ranked according to the specific criteria.The top N features are then selected.Diffrent type of the ranking criteria are used for the univariat filter methods.For an example:- Fisher Score,mutual information and variance of the feature.

Multi-Variate

- In in multi-variate feature selection method we take information of feature to feature.Then we will try to find out the best feature out of those subset features and method from which we select known as "pearson correletion".
- Multivariant filter methods are capable of removing redundant features from the data since they mutual relationship between the features into account.

Univariant Filtering Methods.

- Constant Removal
- Quasi Constant Removal
- Duplicate Feature Removal

Lets Strat With ML.

In [18]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings('ignore')
```

In [19]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import VarianceThreshold
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report,accuracy_score,mean_squared_error
```

In [20]:
```
data = pd.read_csv('Churn_Modelling.csv')
```

In [21]:
```
data.head()
```

Out[21]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1

In [22]:
```
data.tail()
```

Out[22]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMen
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	2	1	
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	1	1	
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	1	0	
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	1	
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79	1	1	

In [23]:
```
data.shape
```

Out[23]:
```
(10000, 14)
```

In [24]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
RowNumber          10000 non-null int64
CustomerId         10000 non-null int64
Surname            10000 non-null object
CreditScore        10000 non-null int64
Geography          10000 non-null object
Gender             10000 non-null object
Age                10000 non-null int64
Tenure             10000 non-null int64
Balance            10000 non-null float64
NumOfProducts      10000 non-null int64
HasCrCard          10000 non-null int64
IsActiveMember     10000 non-null int64
EstimatedSalary    10000 non-null float64
Exited             10000 non-null int64
dtypes: float64(2), int64(9), object(3)
memory usage: 976.6+ KB
```

In [25]:
```
data.describe()
```

Out[25]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	Est
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	1.530200	0.70550	0.515100	1
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	1
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	1
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	1

In [26]:
```
data.columns
```

Out[26]:
```
Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography',
       'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Exited'],
      dtype='object')
```

In [27]:
```
data.nunique()
```

Out[27]:
```
RowNumber          10000
CustomerId         10000
Surname             2932
CreditScore          460
Geography              3
Gender                 2
Age                   70
Tenure                11
Balance             6382
NumOfProducts          4
HasCrCard              2
IsActiveMember         2
EstimatedSalary     9999
Exited                 2
dtype: int64
```

In [28]:
```
from sklearn.preprocessing import LabelEncoder
lable = LabelEncoder()
data['Geography']= lable.fit_transform(data['Geography'])
data['Gender']=lable.fit_transform(data['Gender'])
```

In [29]:
```
data.head()
```

Out[29]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember
0	1	15634602	Hargrave	619	0	0	42	2	0.00	1	1	1
1	2	15647311	Hill	608	2	0	41	1	83807.86	1	0	1
2	3	15619304	Onio	502	0	0	42	8	159660.80	3	1	0
3	4	15701354	Boni	699	0	0	39	1	0.00	2	0	0
4	5	15737888	Mitchell	850	2	0	43	2	125510.82	1	1	1

In [30]:
```
data.drop(columns=['Surname'],inplace=True)
```

In [31]:
```
data.head()
```

Out[31]:

	RowNumber	CustomerId	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	Estimate
0	1	15634602	619	0	0	42	2	0.00	1	1	1	10
1	2	15647311	608	2	0	41	1	83807.86	1	0	1	11
2	3	15619304	502	0	0	42	8	159660.80	3	1	0	11
3	4	15701354	699	0	0	39	1	0.00	2	0	0	9
4	5	15737888	850	2	0	43	2	125510.82	1	1	1	7

In [32]:
```
data.drop(columns=['CustomerId'],inplace=True)
```

In [33]:
```
data.drop(columns=['RowNumber'],inplace=True)
```

In [34]:
```
data.head()
```

Out[34]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	619	0	0	42	2	0.00	1	1	1	101348.88	1
1	608	2	0	41	1	83807.86	1	0	1	112542.58	0
2	502	0	0	42	8	159660.80	3	1	0	113931.57	1
3	699	0	0	39	1	0.00	2	0	0	93826.63	0
4	850	2	0	43	2	125510.82	1	1	1	79084.10	0

In [35]:
```
x= data.drop(columns=['Exited'])
y = data['Exited']
```

In [36]:
```
x.shape,y.shape
```

Out[36]:
```
((10000, 10), (10000,))
```

In [37]:
```
from sklearn.model_selection import train_test_split
```

In [38]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,train_size=0.80,random_state=42,stratify=y)
```

In [39]:
```
x_train.shape,x_test.shape
```

Out[39]:
```
((8000, 10), (2000, 10))
```

Help us to remove the duplicate features inside the data.

In [65]:
```
var = VarianceThreshold(threshold=0.25)
x_train_unique = var.fit_transform(x_train)
x_test_unique = var.transform(x_test)
```

In [66]:
```
x_train_unique.shape,x_test_unique.shape
```

Out[66]:
```
((8000, 7), (2000, 7))
```

In [67]:
```
#to remove the duplicated features
```

In [68]:
```
x_train_T = x_train_unique.T
x_test_T = x_test_unique.T
```

In [69]:
```
x_train_T = pd.DataFrame(x_train_T)
x_test_T =pd.DataFrame(x_test_T)
```

In [70]:
```
duplicated = x_train_T.duplicated()
duplicated
```

Out[70]:
```
0    False
1    False
2    False
3    False
4    False
5    False
6    False
dtype: bool
```

In [71]:
```
keep_them = [not i for i in duplicated]
```

In [72]:
```
x_train_unique = x_train_T[keep_them].T
x_test_unique  = x_test_T[keep_them].T
```

In [73]:
```
x_train_unique.shape,x_test_unique.shape
```

Out[73]:
```
((8000, 7), (2000, 7))
```

In [78]:
```
def RanndomForest(x_train,x_test,y_train,y_test):
    clf = RandomForestClassifier(random_state=42,n_estimators=100)
    clf.fit(x_train,y_train)
    y_pred = clf.predict(x_test)
    print('Accuarcy :-',accuracy_score(y_test,y_pred))
```

In [79]:
```
RanndomForest(x_train_unique,x_test_unique,y_train,y_test)
```

```
Accuarcy :- 0.855
```

In [ ]:
```
R
```