**Introduction**

In this article, we will be discussing feature selection and the techniques that are used in feature selection. Let’s see about the curse of dimensionality.

Suppose in a particular dataset if we have many features, this may increase the threshold value which in turn decreases the accuracy of the model. Whenever we give those data to train our model, the model gets confused because it is learning too much data.

To resolve this situation, what we do is that we do not select all the features from a particular dataset. Instead, we apply various techniques of feature selection. In this article, I will be discussing some of the techniques including

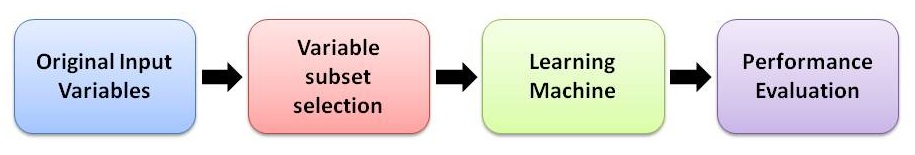
* Univariate selection
* Feature importance
* A correlation matrix with Heatmap

These techniques are very efficient and let’s see how to implement them practically.

Before we start coding, let me explain some basic techniques of feature selection.

**1. Filter method**

First, we will see about the filter method.



In the filter method, we have three sub-components. The first component is that suppose I have all the set of features I will be selecting the best subset.

**How I will be selecting the best subset?**

We can apply various techniques. Some of the techniques I would like to tell you are the ANOVA test which is a statistical method and other one is the CHI SQUARE test and one more method I would specify is correlation coefficient. These are the three techniques we use to select some important features. The important features mean that these features will be much correlated with the target output.

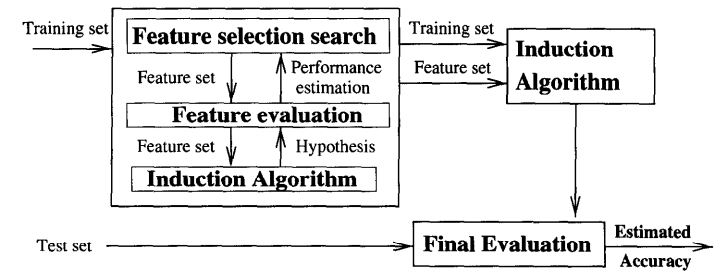
Let’s take an example. Here I am having an independent variable X and a target variable Y.

|  |  |
| --- | --- |
| **X** | **Y** |
| 1 | 10 |
| 2 | 20 |
| 3 | 30 |
| 4 | 40 |

In this scenario, you can see that as X increases, Y also increases. So, concerning the correlation coefficient, you can say that X and Y are highly correlated. We have two terms. One is covariance and the other one is a correlation. Covariance maps the value between 0 and 1. Correlation is between -1 to +1. This correlation is for the Pearson correlation coefficient.

The second technique is the wrapper method.

**2. Wrapper method**



The wrapper method is quite simple when compared to the filter method. Here, you don’t need to apply any statistical kinds of stuff. You have to apply only a simple mechanism. There are three basic mechanisms in this.

Let me explain it.

**2.1 Forward selection**

This method is used to select the best important features from the particular dataset concerning the target output. Forward selection works simply. It is an iterative method in which we start having no feature in the model. In each iteration, it will keep adding the feature.

Let me explain this with an example.

I am considering A, B, C, D, and E as my independent features. Let F be the output or target feature.

Initially, the model will train with feature A only and record the accuracy. In the next iteration, it will take A and B and train and record accuracy. If this accuracy is better than the previous accuracy, it will be considering adding B in its features set. Likewise, in each iteration, it will be adding different features until it reaches better accuracy.

This is what forward selection is.

Next, we will see about backward selection.

**2.2 Backward elimination**

This works slightly differently. Let’s discuss the same example. A, B, C, D, and E are independent features. F is the target variable. Now, I will take all the independent features and train the model. Before training the model, I will just apply a statistical test. This test will say that which feature is having the lowest impact on the target variable. This is how backward elimination is implemented.

Let me explain the recursive feature elimination.

**2.3 Recursive feature elimination**

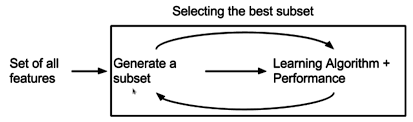
It is a greedy optimization algorithm. The main aim of this method is to select a best-performing feature subset. It will not randomly select any feature. Rather than, it will find out which is the most useful feature. And in the next iteration, it will add the next useful feature concerning the target variable. Finally, it will rank all the features and eliminate the lower ones.

Remember that the above-mentioned techniques are useful when the dataset is small.

But in reality, you will get a large dataset.

Let’s try to understand the third technique called embedded methods.

**3. Embedded methods**



[Source](https://i1.wp.com/blog.contactsunny.com/wp-content/uploads/2018/07/feature_selection_embedded_methods.png?resize=413%2C122)

Let me start with an example. I am having A, B, C, D, and E as independent variables. F is the target variable. The embedded technique creates a lot of subsets from the particular dataset. Sometimes, it may give A to the model and find the accuracy. It may give AB to the model and find the accuracy. It will try to do all the permutations and combinations. Whichever subset is having the maximum accuracy, that will be selected as a subset of features which will later be given to the dataset for training. That is how an embedded method works.

Let’s go and find out how univariate selection is done.

**4. Univariate selection**

Univariate selection is a statistical test and it can be used to select those features that have the strongest relationship with the target variable.

Here, I am using the SelectKBest library. Suppose if you give K value as 5. It will find out the best 5 attributes concerning the target variable.

import pandas as pd

import numpy as np

from sklearn.feature\_selection import SelectKBest

from sklearn.feature\_selection import chi2

data = pd.read\_csv("train.csv")

X = data.iloc[:,0:20]

y = data.iloc[:,-1]

The dataset has many features. We have to select the best one. Because as you know in the curse of dimension, if I increase the number of features after a particular threshold value, the accuracy of the model will decrease.  
For that, I am using univariate selection and the SelectKBest.

bestfeatures = SelectKBest(score\_func=chi2, k=10)

fit = bestfeatures.fit(X,y)

After fitting, I will get two different parameters. One is fit.scores which will calculate the score with respect to the chi-square test value.

dfscores = pd.DataFrame(fit.scores\_)

dfcolumns = pd.DataFrame(X.columns)

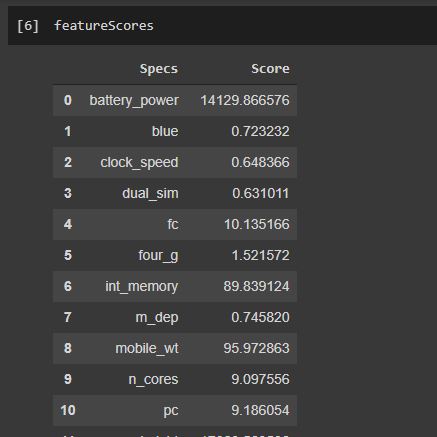
I am concatenating in the next statement for better visualization and I am renaming the column as Specs and Score.

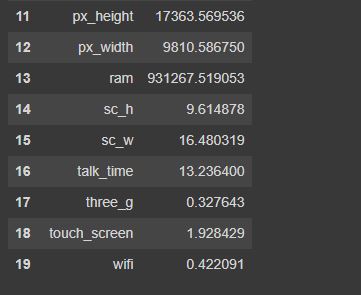
featureScores = pd.concat([dfcolumns,dfscores],axis=1)

featureScores.columns = ['Specs','Score']

Here, you can see all the features. The higher the score, the more important the feature is. here, the ram has the highest score.

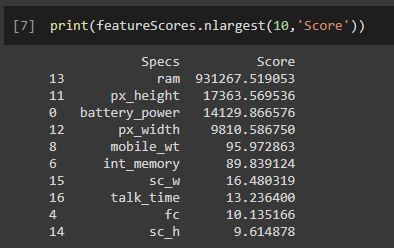
featureScores





I am printing the top 10 features.

print(featureScores.nlargest(10,'Score'))



These 10 best features can be used to train the model.

Let’s look into the next technique called feature importance.

## 5. Feature importance

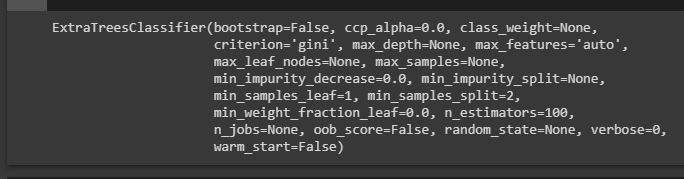
Here, you can get the feature importance of every feature. The higher the score, the more important the feature is. An inbuilt classifier called Extra Tree Classifier is used here to extract the best 10 features.

from sklearn.ensemble import ExtraTreesClassifier

import matplotlib.pyplot as plt

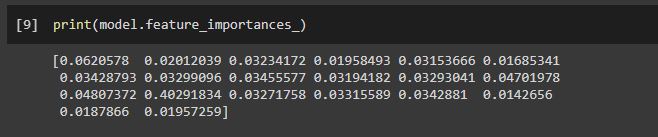
model = ExtraTreesClassifier()

model.fit(X,y)



After fitting, you can see the scores of the features.

print(model.feature\_importances\_)

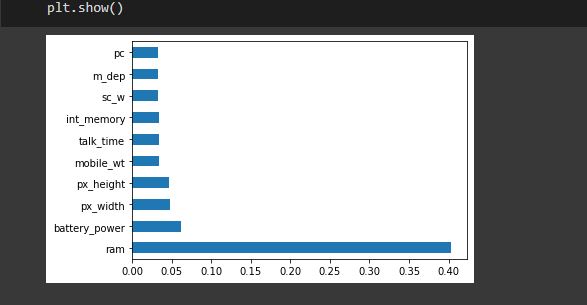


The best 10 features can be seen like this.

feat\_importances = pd.Series(model.feature\_importances\_, index=X.columns)

feat\_importances.nlargest(10).plot(kind='barh')

plt.show()



Let me explain the last technique.

## 6. Correlation matrix with Heatmap

Here, we are checking each and every feature. The correlation can be plotted like this.

import seaborn as sns

corrmat = data.corr()

top\_corr\_features = corrmat.index

plt.figure(figsize=(20,20))

g=sns.heatmap(data[top\_corr\_features].corr(),annot=True,cmap="RdYlGn")

