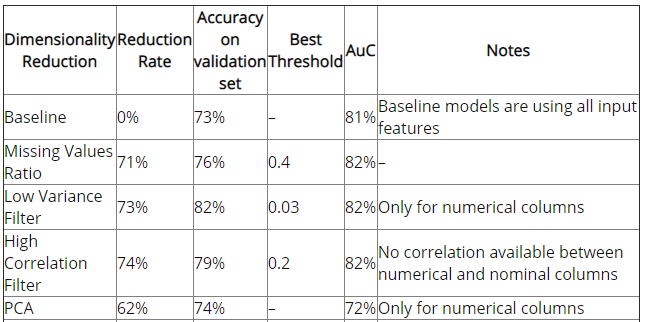
**Assignment 4 – Dimension Reduction**

As discussed during the lecture that dimension reduction is a very vital component of any machine learning pipeline. In this assignment you will practice some dimension reduction techniques on the same task you have performed in assignment number 3. Following techniques are required to be implemented (at least five) and tested by using an open source library such as Scikit-Learn Library in python. You can use the same code as was in previous assignment and optimize it with dimension reduction techniques. You are also required to make comparison tables and graphs for the results of each technique.

* **Principle component analysis (PCA)**
* **Singular value decomposition (SVD)**
* **Self-organizing mas (SOM)**
* **Non-Negative Matrix Factorization**
* **Linear Discriminant Analysis**
* **Low Variance Filter**. Similarly to the previous technique, data columns with little changes in the data carry little information. Thus all data columns with variance lower than a given threshold are removed. A word of caution: variance is range dependent; therefore normalization is required before applying this technique.
* **High Correlation Filter**. Data columns with very similar trends are also likely to carry very similar information. In this case, only one of them will suffice to feed the machine learning model. Here we calculate the correlation coefficient between numerical columns and between nominal columns as the [Pearson’s Product Moment Coefficient](http://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient) and the [Pearson’s chi square value](http://en.wikipedia.org/wiki/Pearson%27s_chi-squared_test) respectively. Pairs of columns with correlation coefficient higher than a threshold are reduced to only one. A word of caution: correlation is scale sensitive; therefore column normalization is required for a meaningful correlation comparison.
* **Backward Feature Elimination**. In this technique, at a given iteration, the selected classification algorithm is trained on n input features. Then we remove one input feature at a time and train the same model on n-1 input features n times. The input feature whose removal has produced the smallest increase in the error rate is removed, leaving us with n-1 input features. The classification is then repeated using n-2 features, and so on. Each iteration k produces a model trained on n-k features and an error rate e(k). Selecting the maximum tolerable error rate, we define the smallest number of features necessary to reach that classification performance with the selected machine learning algorithm.
* **Forward Feature Construction**. This is the inverse process to the Backward Feature Elimination. We start with 1 feature only, progressively adding 1 feature at a time, i.e. the feature that produces the highest increase in performance. Both algorithms, Backward Feature Elimination and Forward Feature Construction, are quite time and computationally expensive. They are practically only applicable to a data set with an already relatively low number of input columns.

Sample output table:



You have already done baseline in previous assignment.

**Helpful links:**

<https://machinelearningmastery.com/dimensionality-reduction-algorithms-with-python/>

<https://www.analyticsvidhya.com/blog/2018/08/dimensionality-reduction-techniques-python/>

<https://www.kdnuggets.com/2015/05/7-methods-data-dimensionality-reduction.html>