Week 4 (08.11 - 17.11.2019)

**Part 1**

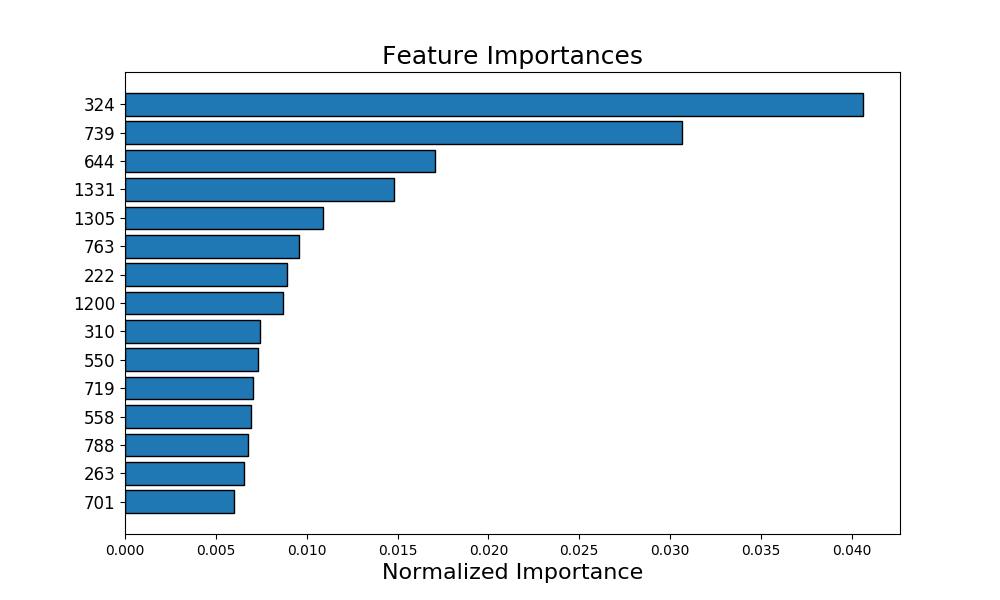
**Feature Selection (cont’d):**

Other features selection techniques I added:

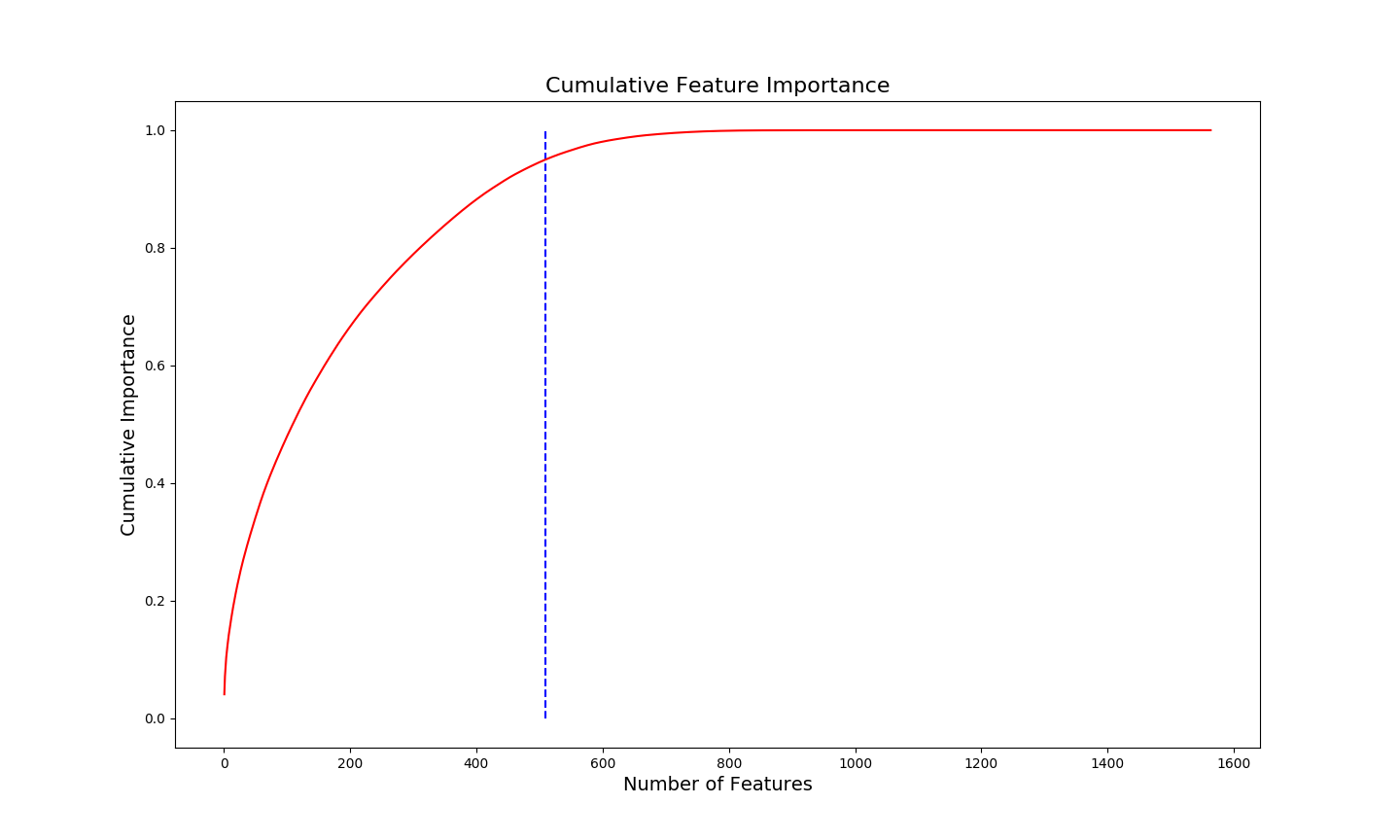
1. **Percentage of NaN values:** If the ratio of the NaN values to the total values in the column is above 0.6, the column will be discarded.
2. **Single unique value:** Get all the columns with only a single unique value and discard them.
3. **Standard deviation:** Add all columns with no standard deviation in the values to the list of columns to be deleted.

Train the resulting data on a GB classifier and extract feature importance, which is an inbuilt function.

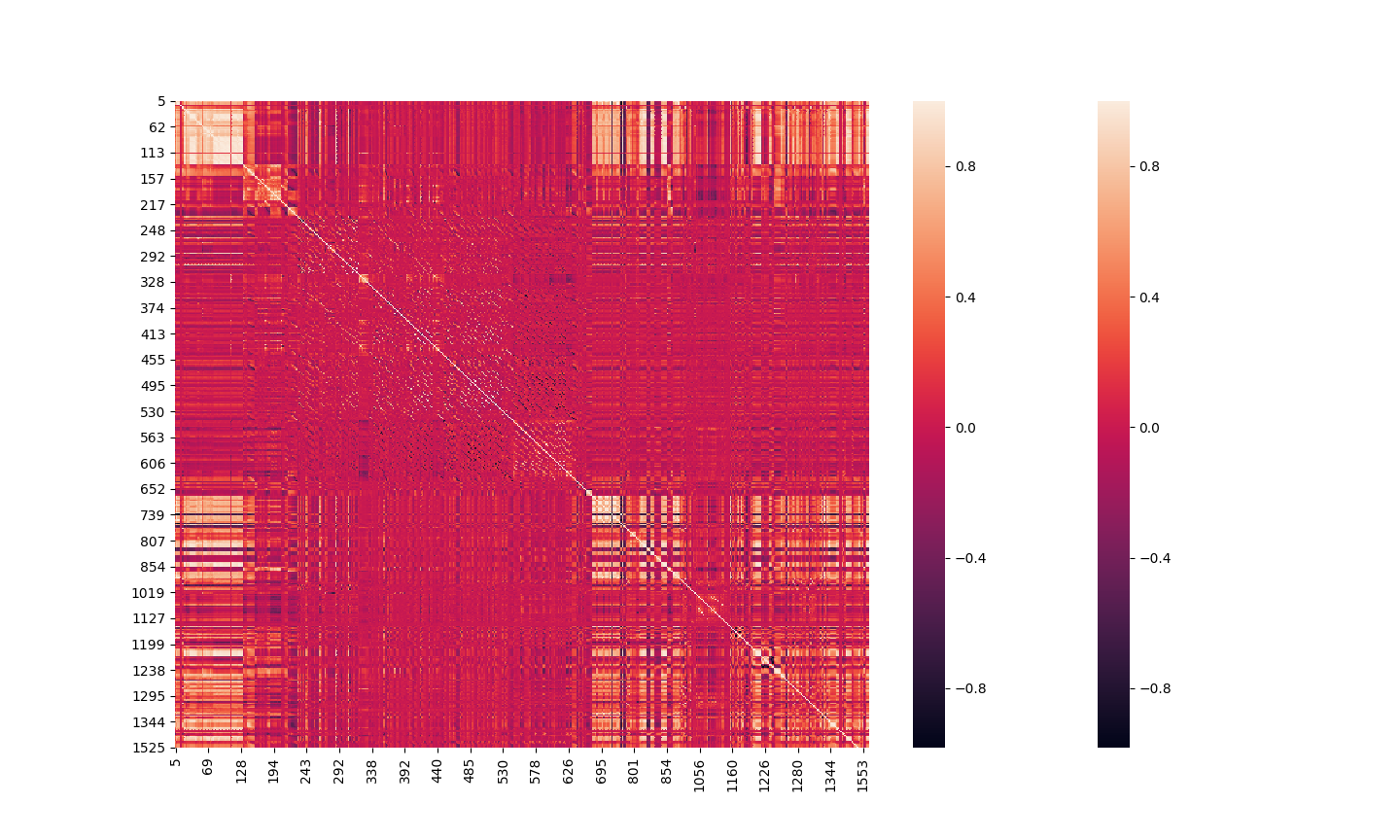
1. **Zero importance features:** Below given is the plot of the feature importance.



1. **Low importance features:** Discard features that don’t contribute to the cumulative threshold, a value that will be set by the user.



1. **Collinearity Matrix:** Get the collinearity matrix from the Pandas Dataframe API, and compare each feature with all the other features. If the collinearity coefficient between them is too high (in the code, it is set to 0.9) then one of these features will be discarded. Below given is the plot of the collinearity matrix.



This analysis was carried out on the **Mordred descriptor dataset**. All the methods used here don’t take into account interaction between the features. The original number of features were **1613**. After column statistics, the number was reduced to **1463**. After the collinearity test, the number of features reduced to **509**. After performing feature selection via importance, **353** features were left.

Training on the reduced feature dataset on RandomForest, GradientBoosting and SVM classification methods did not give much difference in performance, which was around 0.9. This value also did not differ from the performance without any feature selection.

**Part 2**

**Missing Value Imputation:**

I tried 3 kinds of methods, the SimpleImputer and IterativeImputer provided by sklearn and the fillna() API provided by Pandas.

1. **SimpleImputer():** You can set the missing values to the mean, median, most frequenty occuring value in the column, or simply a constant that the user can choose. This works individually on features.
2. **IterativeImputer():** It is a multivariate imputer that imputes missing values by modelling each feature with missing values as a function of other features in a round-robin fashion. But it takes a long time to converge and did not happen on my system.
3. **fillna()**: Built-in method provided by Pandas to impute values. Can use methods like bfill, ffill, which stands for forward fill, takes the next valid value in the column and replaces the missing value.

**Part 3**

**Paper: A Survey on Evolutionary Computation Approaches to Feature Selection**

1.1 Why do we need feature selection?

Feature selection is a difficult task due mainly to a large search space, where the total number of possible solutions is 2n for a dataset with n features. The task is becoming more challenging as n is increasing in many areas with the advances in the data collection techniques and the increased complexity of those problems.

However, most existing feature selection methods still suffer from stagnation in local optima and/or high computational cost. Therefore, we need a global search technique to better solve feature selection problems.

But the problem of FS is hard not only because of the large state space, but also because it is very difficult to determine the interactions between features that could make a positive impact on the performance.

FS involves two main objectives, to maximize classification accuracy and reduce the number of features. These can sometimes be conflicting goals and hence FS can be treated as a multi-objective problem.

1.2 Types of feature selection algorithms:

FS algorithms are divided into 2 categories based on their evaluation methods:

1. **Filter Method**: Features are selected based on some statistical tests.

Advantages: Less expensive and more general than wrapper method algorithms.

Disadvantages: Lower performance than wrapper methods.

1. **Wrapper Method:** They include a classification or learning algorithm in the feature subset evaluation step.

1.3 EC techniques from three aspects:

1. **Search Techniques**: Used only when the search space is small, maybe 50 features. EC techniques do not need domain knowledge and do not make any assumption about the search space, such as whether it is linearly or nonlinearly separable, and differentiable. Another significant advantage of EC techniques is that their population-based mechanism can produce multiple solutions in a single run. However, EC techniques have a major limitation of requiring a high computational cost since they usually involve a large number of evaluations. Another issue with EC techniques is their stability since the algorithms often select different features from different runs, which may require a further selection process for real-world users.
2. **Evaluation Criteria:** For wrapper feature selection approaches, the classification performance of the selected features is used as the evaluation criterion. Most of the popular classification algorithms, such as decision tree (DT), sup- port vector machines (SVMs), Naïve Bayes (NB), K-nearest neighbour (KNN), artificial neural networks (ANNs), and linear discriminant analysis (LDA), have been applied to wrap- pers for feature selection. For filter approaches, measures from different disciplines have been applied, including information theory-based measures, correlation measures, distance measures, and consistency measures.
3. **Number of Objectives:** All the multi-objective feature selection algorithms to date are based on EC techniques since their population-based mechanism producing multiple solutions in a single run is particularly suitable for multi-objective optimization.

1.4: Genetic Algorithms (GA’S) for FS:

1. For wrapper approaches, different classification algorithms have been used to evaluate the goodness of the selected features, e.g., SVMs, KNN, ANNs, NB, multiple linear regression for classification, extreme learning machines (ELMs) and discriminant analysis. SVMs and KNN are the most popular classification algorithms due to their promising classification performance and simplicity, respectively.
2. For filter approaches, different measures have been applied to GAs for feature selection, e.g., information theory, consistency measures, rough set theory, and fuzzy set theory.

GAs address combinatorial optimization problems by identifying good building blocks of information, combining complementary blocks via crossover, and adjustment via mutation. Thus, GAs are likely to be suited to domains in which there are groups of interacting features, potentially with multiple good subsets, to consider.

1.5 Genetic Programming for FS:

GP is used more often in feature construction than feature selection because of its flexible representation. In feature selection, most GP works use a tree-based representation, where the features used as the leaf nodes of a tree are the selected features. GP can be used as a search algorithm and also as a classification algorithm.

In filter approaches, GP is mainly used as the search algorithm. In most wrapper approaches, GP is used as both the search method and the classification algorithm. In a very few cases, GP was used as a classification algorithm only in a feature selection approach.

1.6 Particle Swarm Optimization for FS:

Both continuous PSO and binary PSO have been used for both filter and wrapper, single objective and multi-objective feature selection. The representation of each particle in PSO for feature selection is typically a bit-string, whereby the dimensionality is equal to the total number of features in the dataset.

The bit-string can be binary numbers in binary PSO or real-value numbers in continuous PSO. When using binary representation, 1 means the corresponding feature is selected and 0 means it is not selected. When using the continuous representation, a threshold θ is usually used to determine the selection of a particular feature, i.e., if the value is larger than θ, the corresponding feature is selected. Otherwise, it is not selected.

PSO should suit domains in which there is a structure in how features interact, i.e., low sensitivity to the inclusion of each feature in a solution, and where fast convergence does not lead to local optima. PSO has an advantage over GAs and GP of being easy to implement.

However, representation of PSO might not scale well on problems with thousands or tens of thousands of features, since it forms a huge search space.

1.7 Ant Colony Optimization for FS:

In ACO for FS, the proportion of filter approaches is much higher than that in GAs, GP, and PSO for feature selection. The graph representation in ACO is more flexible than the representation in GAs and PSO, but the order of encoding the features as nodes may influence the performance. Building feature subsets through ants traversing nodes is similar to many traditional ways of gradually adding or removing features to a subset, which makes it easy to adopt existing filter measures in ACO for feature selection.

However, the graph representation may not scale well to problems with thousands of features, which might be the reason why current ACO approaches focus mainly on relatively small-scale problems.

1.8 Applications:

1. Image and signal processing, including image analysis, face recognition, human action recognition, EEG brain–computer-interface, speaker recognition, hand- written digit recognition, personal identification, and musical instrument recognition.
2. Biological and biomedical tasks, including gene analysis, biomarker detection, and disease diagnosis.
3. Business and financial problems, including financial cri- sis, credit card issuing in bank systems, and customer churn prediction.
4. Network/Web service, including text mining, Web ser- vice, network security, and email spam detection.
5. Others, such as power system optimization, weed recognition in agriculture, melting point prediction in chemistry, and weather prediction.

1.9 Issues and Challenges:

1. **Scalability:** Most of the existing EC-based large-scale feature selection approaches employ a two-stage approach, where in the first stage, a measure is used to evaluate the relevance of individual features, then ranks them according to the relevance value. Only the top-ranked (better) features are used as inputs to the second stage to further select features from them. However, the first stage removes lowly-ranked features without considering their interaction with other features.
2. **Computational Cost:** To reduce the computational cost, two main factors, an efficient search technique and a fast evaluation measure, need to be considered.
3. **Search Mechanisms:** EC algorithms are stochastic approaches, which may produce different solutions when using different starting points. Even when the fitness values of the solutions are the same, they may select different individual features. Therefore, the stability of the algorithms not only involves the difference of the fitness values, but also involves the consistency of the selected features. Therefore, to propose new search algorithms with high stability is also an important task.
4. **Measures:** Discovering complex feature interaction is very challenging, and only a few works have been conducted on this direction. There are some measures that can evaluate groups of features, but they are usually computationally expensive, such as rough set-based measures. Furthermore, many studies show that filter methods do not scale well above tens of thousands of features. Therefore, new measures still need to be developed for feature selection, especially when dealing with large-scale problems.
5. **Representation**: A good representation scheme can help to reduce the search space size. It in turn helps to design new search mechanisms to improve the search ability. Furthermore, the interpretation of the solution is also an important issue closely related to the representation.
6. **Multi-Objective FS:** Developing new evaluation metrics and further selection methods to choose a single solution from a set of trade-off solutions is also a challenging topic. Finally, besides the two main objectives, other objectives, such as the complexity, the computational time, and the solution size (e.g., tree size in GP and number of rules in LCSs), could also be considered in multi-objective feature selection.
7. **Feature Construction**: If the original features are not informative enough to achieve promising performance, feature selection may not work well, yet feature construction may work well. One of the challenges for feature construction is to decide when feature construction is needed.
8. **Number of Instances:** If there are tens of thousands of features, but the number of instances can be smaller than one hundred because of the high cost of collecting such instances. It is difficult to split the data into a training set and a test set to represent the actual problem. On the other hand, when the number of instances is too big, one major problem is the computational cost.