# Feature Ranking and Selection

**August 12, 2021**

Abdullah Al Zubaer

# Contents

# 1 Introduction

To extract knowledge and patterns from data that are noisy and have features that are irrelevant or redundant is a challenging task. The source of these noisy data can be several, for example, the technologies used for collecting the data (for example sensors to collect certain data) or the source itself. The bearing of these features makes, for example, learning algorithms decrease in performance, increase computational complexity, increase memory requirements, increase learning runtime, and can significantly reduce the generalization property of the learning model.

Additionally, when the dimensionality of the features is very high it becomes a significant issue for the learning algorithm to perform optimally, in other words, the *curse of dimensionality* problem arises. When the number of features is very large the model tends to overfit, leading to a performance decline of the model regarding generalization capability.

Therefore, methods are developed to reduce the dimensionality of the features. One of the techniques is "dimensionality reduction". Dimensionality reduction (DR) can be divided into two techniques, feature extraction (FE) and feature selection (FS). Both of these techniques can help to improve the problems discussed previously. Moreover, with FS techniques features can also be ranked according to their importance/predictive power.

Both of these techniques approach the problem of DR differently. In FE, a new lower-dimensional feature space is constructed based on the original higher dimensional feature space and eventually leading to dimensionality reduction (for example by adopting Principle Component Analysis). While, in FE methods, a subset of the features that were present originally is selected, without any kind of transformations of the features and only selecting the features that are more informative compared to the others.

Based on the given goal of this project, which is a classification problem, the scope of this analysis will be narrowed to *feature selection for classification problems* only.

# 2 Feature selection

Depending on the training dataset if a label/s is present or not, feature selection algorithms are separated into three domains, supervised, unsupervised, and semi-supervised feature selection. Following the scope of this analysis, we will focus on supervised feature selection only, which can be further classified into the filter, wrapper, and embedded models.

Filter model depends on the characteristics of the dataset (and does not depend on the learning algorithm), for example, correlation, information gain, etc. The advantage of using the filter model is, we do not need to execute a learning algorithm to perform feature selection therefore it is less computationally costly compared to the other methods discussed below. Whereas, a disadvantage of using the filter approach is that this method completely overlooks how the learning algorithm is performing based on the selected features.

Whereas, the wrapper model utilizes the accuracy of the classifier based on the features and select features, iteratively, that increase the accuracy of the classifier (and consequently, given the nature of this method, it is computationally expensive). Briefly, there are two components in the wrapper model approach, a feature search, and a feature evaluation component. Both of these components interact with each other in an iterative process. First, the feature search component will select, greedily, a set of features, and the feature evaluation component, a classifier, will estimate the performance with the selected feature. This process is repeated until a certain number of features (threshold number of features, provided by the user) are selected that ensures the highest performance of the classification algorithm.

A greedy search method can be adopted for this method, compared to exhaustive search which is much more computationally expensive. The two approaches are forward selection and backward selection. In forward selection, the search starts with an empty feature set, and gradually the number of features in the feature set is increased until the maximum performance is achieved for a given number of features by the learning algorithm. Whereas, in the backward selection, a similar process is executed but the greedy search starts with all the features.

The third approach, embedded models, adopts both the filter and wrapper model approach for feature selection. There are three kinds of the embedded model method mentioned in the literature, one of them is an algorithm that has a mechanism inside it to select features, for example, ID3, C4.5, or CART (algorithms used to build decision tree which will also be the focus of our analysis).

Features can be classified into three categories, flat, structured, and streaming. In

this analysis, we will assume that the features are flat and utilize feature selection methods that are based on flat features. Feature selection methods based on flat features assume the features to be independent of each other.

In the next section, we will analyze the properties of the dataset and test the feature independent assumption.

# 3 Dataset properties

To analyze the properties of the artificially generated dataset, at first, we have examine the correlation between the features, and the labels with the features. Correlation between features, or features and targets will inform us to what extent they are correlated to each other. Moreover, highly correlated features might not be useful for executing a machine learning algorithm since they will be redundant and won't bring any contribution to improve the learned model.

Figure 1 provides Pearson correlation between features, and between features and target class. From Figure 1, we can conclude that the features are not highly correlated, and we can assume that the features are independent. As mentioned before, correlated features are not useful for the learning process, since features that are highly correlated will not provide new information for the learning algorithm to utilize. Furthermore, we can observe that the correlation between sensor0, sensor4, and sensor8 are the highest w.r.t to the target label. This observation intuitively leads to the fact that these three sensors' values are most responsible for classifying the label of the class. Additionally, from the correlation heatmap, we can observe that sensor2, sensor6, and sensor9 have the least correlation with the class label. This also intuitively tells us that these three sensors values are least responsible for classifying the label of the class. But, this requires further study which we will be continuing later in this analysis.

From Figure 2, we can observe that the dataset is balanced. The total number of instances for the -1.0 class and 1.0 class is equal. An imbalanced dataset will make the learning algorithm bias, since the class labels are equally distributed in the dataset we can assume that the learning algorithm will not be biased to any particular class. Therefore, we do not need to perform any other techniques to mitigate class imbalance issues (for example, oversampling or undersampling).

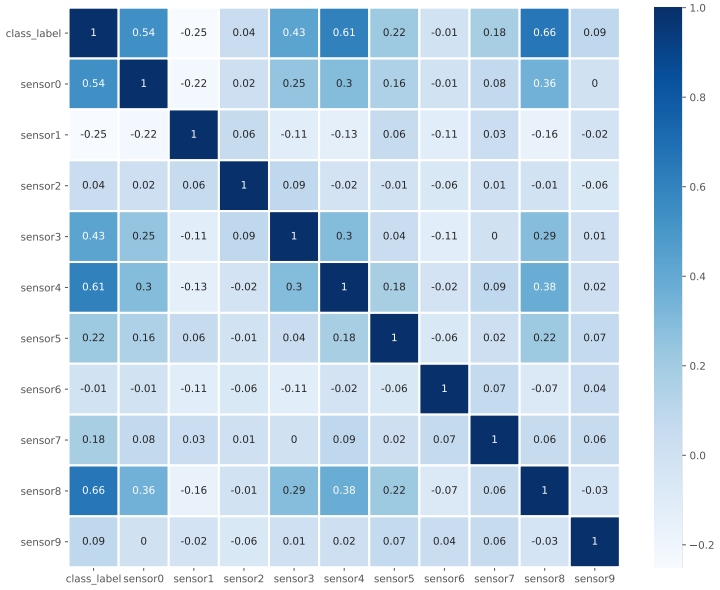


Figure 1: Correlation heatmap for the dataset.

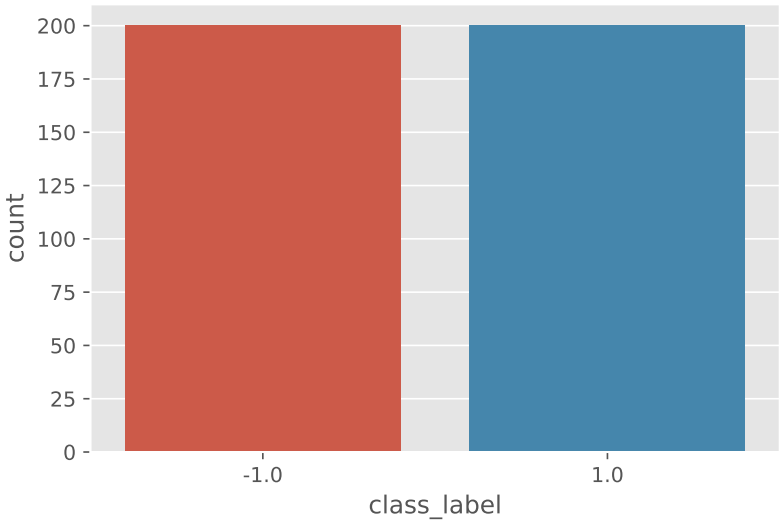


Figure 2: Class distribution.

# 4 Evaluation and Results

In this section, we will present the experiments and their results along with a discussion.

## 4.1 Filter approach

In this approach we will consider uni-variate feature evaluation instead of multi-variate feature evaluation, i.e. each feature will be ranked individually.

**Experiment 1: Using information gain**

### Experiment setup

For this experiment we have calculated the information gain of the features w.r.t the class label.

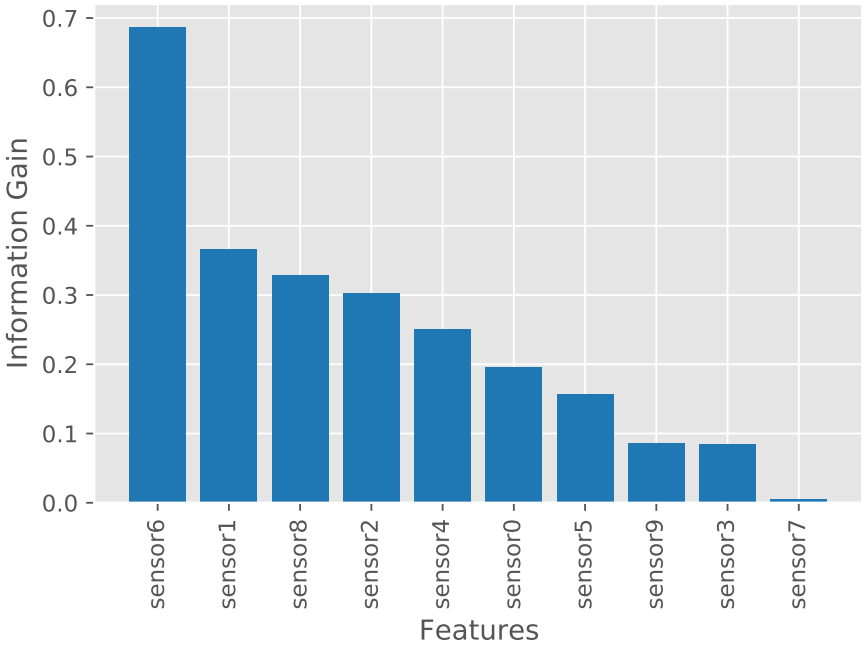


Figure 3: Information Gain of each features (descending order).

### Result and discussion

In Table 1 and in Figure 3 we can observe the ranking of the sensors w.r.t to the information gain value. Usage of information gain is efficient and is simple to

Table 1: Ranked (descending order) sensor.

| **Sensors** | **Information Gain** |
|---|---|
| sensor6 | 0.686 |
| sensor1 | 0.366 |
| sensor8 | 0.328 |
| sensor2 | 0.303 |
| sensor4 | 0.251 |
| sensor0 | 0.196 |
| sensor5 | 0.156 |
| sensor9 | 0.086 |
| sensor3 | 0.084 |
| sensor7 | 0.005 |

interpret regarding the importance of the features w.r.t the class labels. It gives us a scalar measure that represents to what extend the features and the labels are dependent on each other. In another word, which features has more information to determine the class label. A feature with high information gain is more relevant than a feature with low information gain. We can conclude that sensor6 is the most important feature in our feature space.

Given the nature of this method, with no involvement of any learning algorithm, it is scalable. However, due to the absence of a learning algorithm involved in this process of ranking the features, the ranking might not reflect the true performance of the algorithm with this features. This can be easily mitigated by training the dataset with the specific number of features with the highest information gain and testing the algorithm with completely unseen data i.e. test data. If the algorithm performs "well" with the test data then we can safely assume that the feature selected based on this method is reasonable.

## Experiment 2: ANOVA test

### Experiment setup

For this experiment we have performed, analysis of variance, ANOVA, F test for selecting the features. This test will be performed pairwise ANOVA test based on the attributes and the class label.

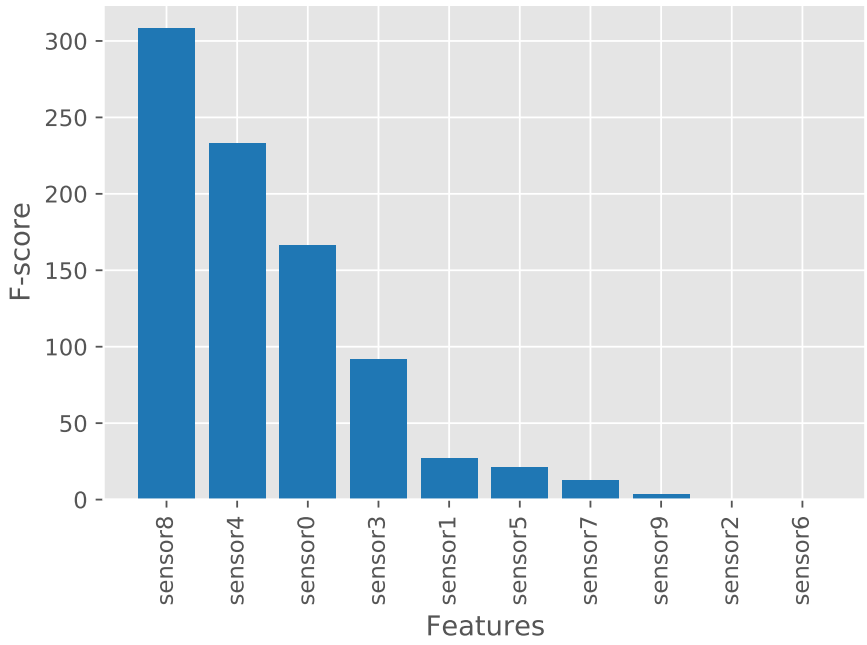


Figure 4: F-score of each features (descending order).

**Result and discussion**

In Figure 4 and Table 2 we can observe the ranking of the features w.r.t to ANOVA F-value, which represents the degree of dependency between the features and labels. A higher score represents a higher dependency between the feature and the label. Therefore, we can conclude that sensor8 has the highest dependency with the class label present, indicating the highest importance of this feature. This is because this score represents which feature has the relevant characteristic to classify the target class

As far as we are aware, one of the assumptions that the dataset must hold to perform the ANOVA test is that the data are independent and identically distributed (i.i.d.), since we have not confirmed if the given dataset holds this assumption, the ranking might not be reasonable.

However, this method is scalable given that no learning algorithm is involved during this process, which eventually eliminates the runtime involved in executing the learning algorithm to determine if the chosen feature is representative of the dataset or not. This also brings a weakness of this method similar to what was discussed in Section 4.1 in Experiment 1.

Table 2: Ranked (descending order) sensor.

| Sensors | F-Score |
|---|---|
| sensor8 | 308.195 |
| sensor4 | 232.95 |
| sensor0 | 166.675 |
| sensor3 | 92.172 |
| sensor1 | 26.99 |
| sensor5 | 21.054 |
| sensor7 | 12.665 |
| sensor9 | 3.618 |
| sensor2 | 0.599 |
| sensor6 | 0.022 |

## 4.2 Wrapper approach

## Experiment 1: Random Forest

### Experiment setup

For this experiment, we have taken the wrapper method approach for feature selection. As mentioned in Section 2, a greedy search strategy is adopted for this method, with forwarding and backward selection techniques. Following the limitation of the machine learning library we are utilizing for this approach, it is not possible to rank the features using forward selection techniques. Since this is not the goal of this analysis (i.e. to select features only without ranking), we will perform only backward selection (recursive) techniques along with Random Forest as our classification algorithm for the evaluation component. This approach allows us to rank and select the features simultaneously.

The hyperparameter for the random forest is kept as provided by default in the scikit-learn library and the trees are built using the CART algorithm.

### Result and discussion

In Table 3 we can observe the ranking of the features in descending order. The features are ranked based on recursive feature elimination (RFE) techniques. In RFE features are selected in a recursive manner. Briefly, the estimator is trained using

Table 3: Ranked (descending order) sensor.

| **Sensors** |
|---|
| sensor6 |
| sensor8 |
| sensor4 |
| sensor0 |
| sensor2 |
| sensor1 |
| sensor3 |
| sensor9 |
| sensor5 |
| sensor7 |

complete feature space, then recursively the number of features are reduced to the desired number (provided by the user). During this process, the least important features (according to the estimator) are removed from the initial complete feature space. As for the parameter of this method, we have removed 1 feature in each iteration. Also, other classification algorithms can be utilized in a similar way to rank the features.

The feature importance is calculated based on Gini importance which indicates the total decrease in node impurity and the final value is the average overall trees in the ensembles for that feature. Gini is a metric to measure the impurity of the node that is created in the tree. If the leaf contains an equal number of two classes, it is the least pure and has a value of 0.5. Similarly, a pure leaf will have a Gini value of 0. In traditional decision tree algorithm, to determine how suitable and effective a feature is to split a node, a measure is calculated which is called "gain". Gain is calculated by the difference between the impurity of the parent and the child node. The higher the gain value the better the feature is to split that particular node.

Briefly, random forest is an ensemble learning method, where multiple decision trees are created in order to perform classification (or regression). A decision tree, if allowed to grow all the way to the end of pure leaves, will be overfitting to the training example. Therefore decision tree has low bias and high variance. On the other hand, random forest overcomes this issue of overfitting by creating several decision trees and training with sample (with replacement) from

the data set, including selecting a subset of features. During testing, test data are fed to all the decision trees and by majority vote, the instance is classified.

The strength of this method is that we are directly tracking the performance of the learning algorithm with the features, and then we can select the features that are more important with respect to the class label for classification. However, the weakness of this method is that it is computationally expensive since we are training the model every time in each iteration, based on the size of the dataset this method will take computational time. Nevertheless, this approach gives a reasonable result that can be utilized to perform feature selection or ranking.

Since the wrapper approach allows us to use any kind of classification algorithms (that has an internal mechanism to select features) for feature selection, as an alternative approach, we can use an advanced classification algorithm, for example, XGBoost for a larger dataset.

## 4.3 Embedded approach

## Experiment 1: Random Forest

### Experiment setup

For this experiment, we have also adopted random forest for ranking the features. The random forest has an internal criterion based on which features are selected. Even though in random forest several decision trees are created, the final metric for feature importance is normalized.

The hyperparameter for the random forest is kept as provided by default in the scikit-learn library and the trees are built using the CART algorithm.

### Result and discussion

In Table 4 and Figure 5 we can observe the features are ranked in descending order w.r.t to Gini importance metrics (normalized). This metric is used by a random forest algorithm to split the nodes. As discussed in detail in Section 4.2, the higher the Gini importance value, the more important the feature is. Therefore according to the Gini importance metric sensor8 is the most important feature.

The Embedded model approach has the advantage of both filter and wrapper methods. It is less computationally expensive (in contrast with the wrapper method) and allows the feature to interact with the learning algorithm and therefore allowing consideration of the specific learning algorithm (in contrast with the filter method).

Table 4: Ranked (descending order) sensor.

| **Sensors** | **Gini importance** |
|---|---|
| sensor8 | 0.2575 |
| sensor4 | 0.2254 |
| sensor6 | 0.2049 |
| sensor0 | 0.1063 |
| sensor2 | 0.0638 |
| sensor3 | 0.0546 |
| sensor1 | 0.0334 |
| sensor5 | 0.0210 |
| sensor9 | 0.0198 |
| sensor7 | 0.0133 |

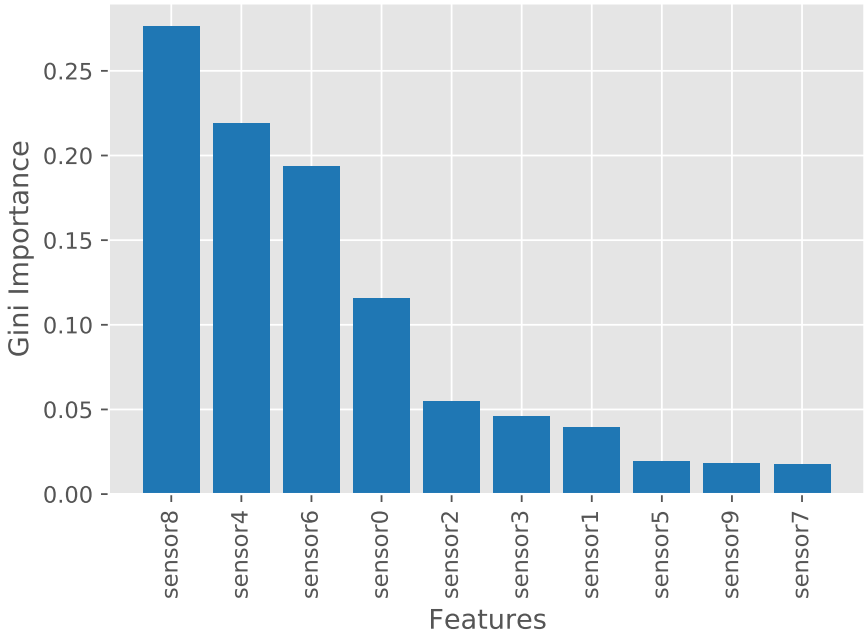


Figure 5: Gini importance of each feature (descending order).

# 5 Conclusion

In this analysis, we have presented three approaches for feature selection/ranking, filter model, wrapper model, and embedded models. Each approach has its own strength and weakness. For example, the filter model approach is computationally inexpensive, whereas the wrapper model is computationally expensive and embedded models take the best of both worlds.

Based on the experiments performed both wrapper and embedded model approach has ranked sensor8, sensor4, sensor6, sensor0, and sensor2 as top five importance features. Whereas, according to the filter model approach, using information gain, sensor6, sensor1, sensor8, sensor2, and sensor 4 as the top five important features. And with the ANOVA test, sensor8, sensor4, sensor0, sensor3 and sensor 1 are the top five important features. Given the assumption that should hold in the dataset to perform the ANOVA test, we think this ranking can be misleading.

One limitation in our experiments was, we did not split the data into train, test, and validation sets and performed the experiments with the complete dataset. Even though the experiments we have performed can be a baseline for working with a larger dataset, but in principle, we should at least separate our dataset into train and test sets, and test our model with the top-k ranked feature to verify the performance of our model. This would allow us to determine if our model would perform better when top-k ranked features were used instead of all the features. Given the small number of the dataset, we have intentionally used all the data for feature ranking but in the future (in the real-world dataset), we would like to test our model with a test dataset containing only the top-k ranked features.

# References

[TSK06] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introducing to Data Mining*. 2006.

[Ped+11] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[Bui+13] Lars Buitinck et al. "API design for machine learning software: experiences from the scikit-learn project". In: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 2013, pp. 108–122.

[TAL14] Jiliang Tang, Salem Alelyani, and Huan Liu. "Feature selection for classification: A review". In: *Data classification: Algorithms and applications* (2014), p. 37.

[GMS17] Baptiste Gregorutti, Bertrand Michel, and Philippe Saint-Pierre. "Correlation and variable importance in random forests". In: *Statistics and Computing* 27.3 (2017), pp. 659–678.

[BMM18] Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. "A systematic study of the class imbalance problem in convolutional neural networks". In: *Neural Networks* 106 (2018), pp. 249–259.