Report: Smart Steel feature ranking
Author: Tung Dinh
Stuttgart 01.06.2021

Introduction

Why do we need to rank features?

In the ML system design, pre-processing the dataset is one of the most important steps, in which we try to grasp as many as possible insights from the dataset. Understanding the problem statement of the pipeline, we can get our hands on the feature ranking. Feature ranking could be seen as a part of feature selection. The fact is that not all features are useful for the final task, and adding or removing one feature could change the accuracy of the final model. Choosing the correct features based on their ranks could bring some advantages:

- Faster training model
- Improve accuracy
- Save time and space
- Reduce overfitting

Although tree-based model is the most common among all, we will go through some methods to rank features:

- Filter method
- Tree-based or intrinsic method
- Additional ML method
- Wrapper method

The main problem with the dataset is testing the ranked features.
In order to understand the ranking models and to pick the best performing one, one final (neutral) model should try to predict the classes of sensors based only on the ranked sensors.
Therefore, once we have the ranked sensors, it is very difficult to argue that the ranking is totally correct.

Components

- task_data.csv
- task.txt
- README.md
- svmranker.py: a class built from soft SVM, including other functions for the ease of computation.
- treeranker.py: a class built from tree-model, including with other functions for the ease of computation.
- my_notebook.ipynb: includes the pipeline of the process
- tree_ranking.txt: ranked features from tree model
- svm_ranking.txt: ranked features from soft SVM

- ReportSmartSteel.pdf: the final report.

In each ranker, we have:

- dataLoader: create features and labels, split train-test sets
- Train: perform fit model
- Permutation: verify the ranked feature with permutation importance, it can also be used alone, by adjusting the train-test split to 1
- Report: plot the ranking and returns ranking as list

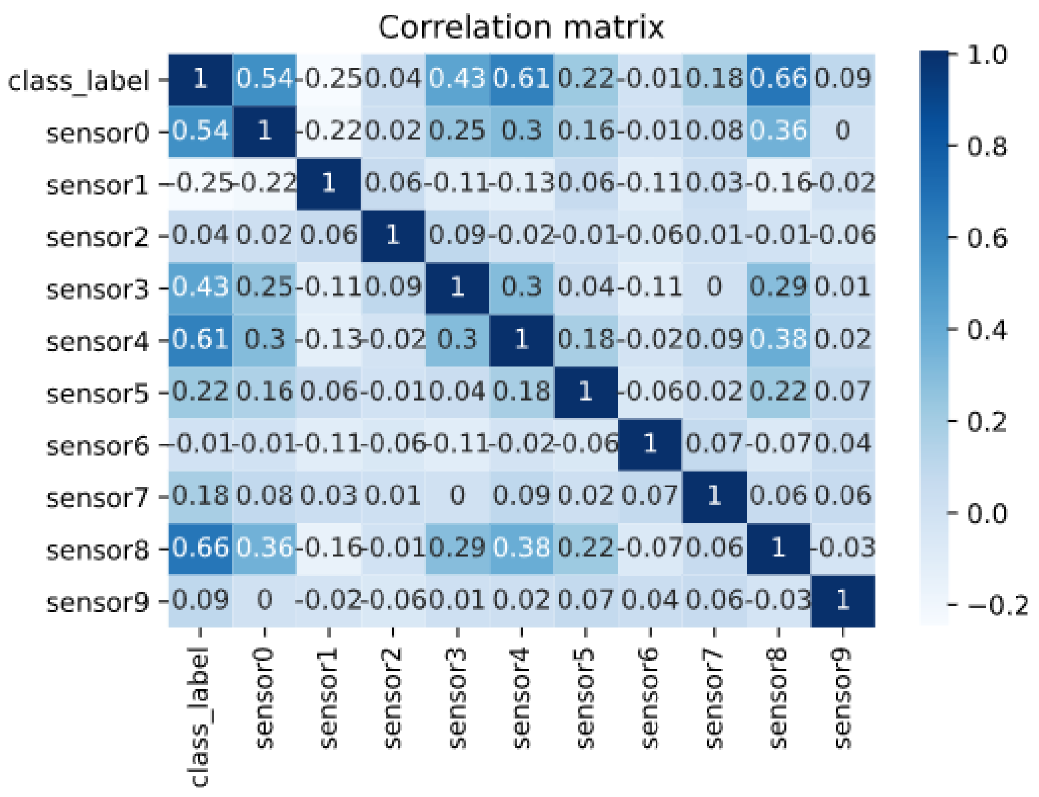
Experiments

We attempt to pick one model for each method. For the filter method, the very basic one is Correlation matrix, XGBClassifier for tree-based, SVM for ML model and Permutation Importance for wrapper method. Also, to understand if the model performs well under the ranking task, we are tempted to use a wrapper method to verify it. To be specific, we will try to use 80% of the dataset for ranking with tree-based and ML models, then we will use 20% test set to verify if the ranking brings us some insights or not. Since the wrapper method depends solely on the target model (tree-based and ML), it might not be a good idea to compare them unless we have a standard model for the classification task.

Result

<u>Correlation matrix:</u>

They the the general methods, which we expect to pick a subset of features. Using correlation matrix is the most basic filter method. We can see that all 10 sensors show the independence to each other with small correlation values. On the class column, we can see the top three sensors are sensor 8, 4, and 0. Naively, this means the value of the class column depends on 3 sensors more than others.



	class_label	sensor0	sensor1	sensor2	sensor3	sensor4	sensor5	sensor6	sensor7	sensor8	sensor9
class_label	1	0.54	-0.25	0.04	0.43	0.61	0.22	-0.01	0.18	0.66	0.09
sensor0	0.54	1	-0.22	0.02	0.25	0.3	0.16	-0.01	0.08	0.36	0
sensor1	-0.25	-0.22	1	0.06	-0.11	-0.13	0.06	-0.11	0.03	-0.16	-0.02
sensor2	0.04	0.02	0.06	1	0.09	-0.02	-0.01	-0.06	0.01	-0.01	-0.06
sensor3	0.43	0.25	-0.11	0.09	1	0.3	0.04	-0.11	0	0.29	0.01
sensor4	0.61	0.3	-0.13	-0.02	0.3	1	0.18	-0.02	0.09	0.38	0.02
sensor5	0.22	0.16	0.06	-0.01	0.04	0.18	1	-0.06	0.02	0.22	0.07
sensor6	-0.01	-0.01	-0.11	-0.06	-0.11	-0.02	-0.06	1	0.07	-0.07	0.04
sensor7	0.18	0.08	0.03	0.01	0	0.09	0.02	0.07	1	0.06	0.06
sensor8	0.66	0.36	-0.16	-0.01	0.29	0.38	0.22	-0.07	0.06	1	-0.03
sensor9	0.09	0	-0.02	-0.06	0.01	0.02	0.07	0.04	0.06	-0.03	1

Advantages and disadvantages of this method

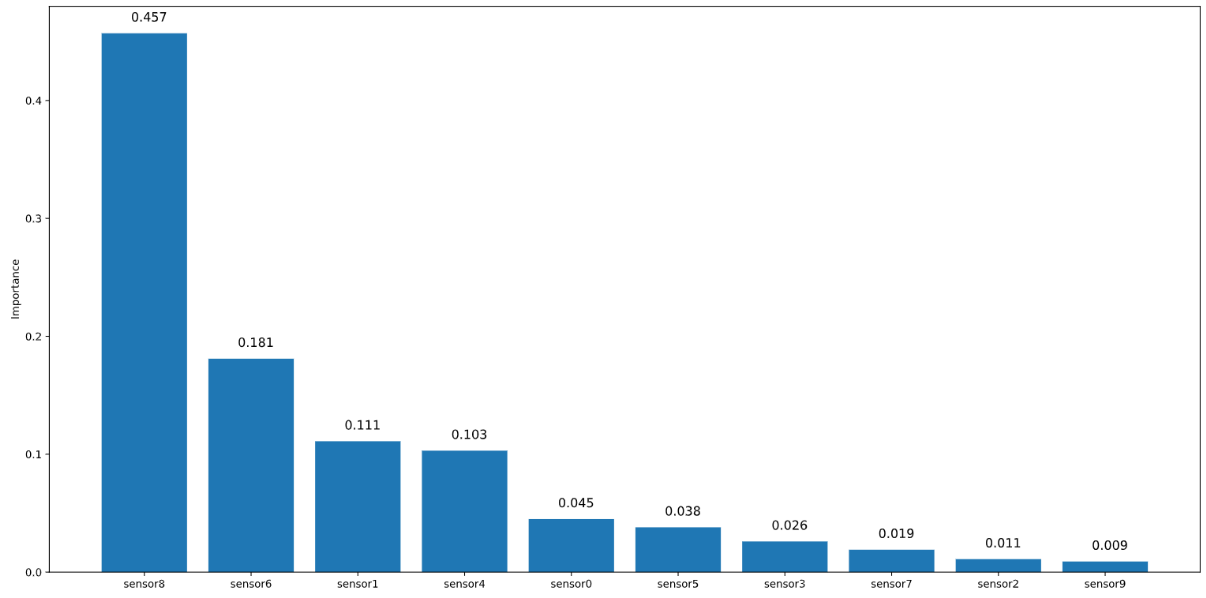
- Correlation method is simple, fast to calculate, and shows the positive/ negative/ neutral relationship between features.
- It can also be applicable for larger dataset, where we might have many dependent features (collinearity), and these dependent features could be removed to make the dataset compact while maintaining (or even improving) the performance of the model.
- Its downside is: we can incorrectly assume the relationship between features since the dataset is too small.

Since this method is applicable for larger dataset, it can be seen as a scalable method.
Other methods, which could be considered here are T-test, ANOVA and ppscore because of independent features. One more thing to note here, is that the dataset should follow normal distribution here. And it is difficult to verify since we do not know the distribution of the dataset, we only have a subset.
https://github.com/8080labs/ppscore

<u>XGBClassifier:</u>

The model shows the importance of features: the most important ones are sensors 8, 6 and 4.
Advantages and disadvantages of XGBClassifier

- It is immune to multicollinearity of the dataset
- It is the perfect algo for this situation, since we only have 2 classes and small dataset of 400 samples, it also calculates fast
- Its downside is when we have undefined values, missing values or more complicated dataset, with more classes, more features.

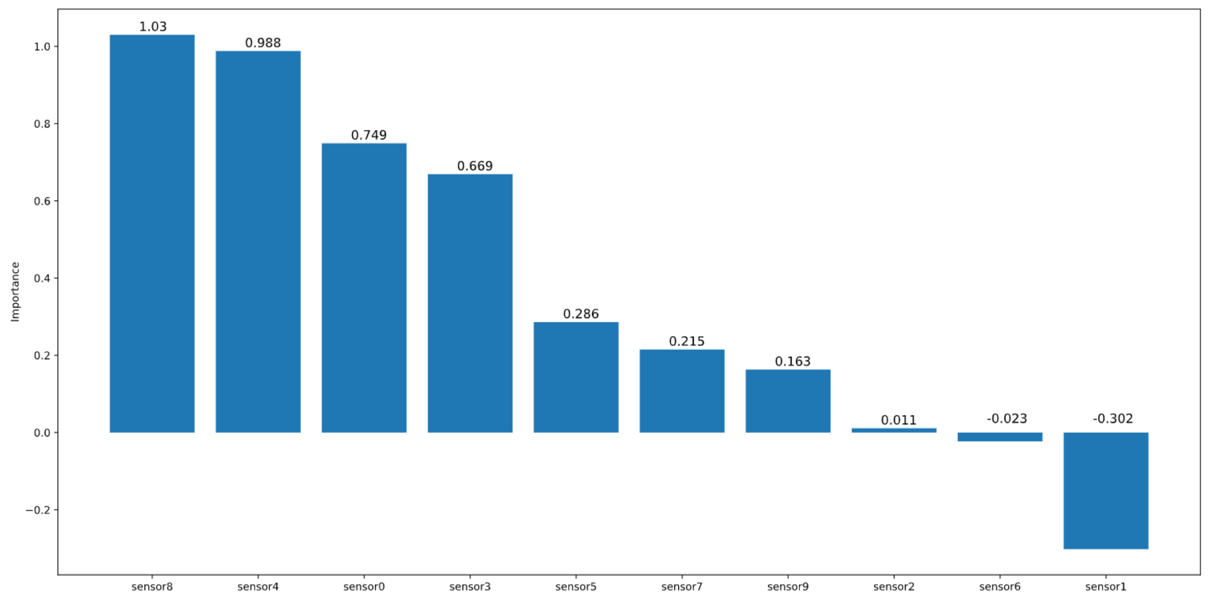


Verification with wrapper method: there is a small conflict in ranking with the dataset, in which the most important ones are sensors 6, 8 and 4. We can see that, with XGBClassifier, we can tell that sensors 8, 6 and 4 are more important than other in classifying the class using XGBClassifier.

Another method to considered here: Random forest tree, it is also included in the code, however, during research, XGBClassifier is shown to be better.

<u>Other ML method</u>

SVM is powerful in classification, and using soft SVM, we try to classify our dataset, based on the 10 sensors. The sensors are ranked on their coefficients.



Verification with wrapper method: there is a small conflict in ranking with the dataset, in which the most important ones are sensors 8, 4 and 0. We can see that, with soft SVM, we can tell that sensors 8, 0 and 4 are more important than other in classifying the class using soft SVM.

Advantages and disadvantages of SVM

- It is relatively powerful in classification tasks, even in case of not-so-clear margin, we can use soft SVM.
- It can avoid overfitting.
- It works well with high dimensional data, we have here 10 features.
- It is shown to be memory efficient also.
- Luckily enough, with our small dataset, it should have no problems. However, with a larger dataset, we might have a time problem.
- Also, the output of SVM depends on our choice of kernels.

In this case, we use linear kernel, it can work on large (only to consider the fitting time) and high dimensional dataset, it can also be seen as a scalable model.

Advantages and disadvantages of the wrapper method

- This method works well in destroying each feature and discovering the accuracy of the model in each turn.
- It could be ideal to verify the ranked feature regarding to the host model.

- Since it is a repeated process, time is a factor to be considered, and it might not be a good idea to apply for a dataset with many features.

Another wrapper method is Recursive Feature Elimination. This method returns the accuracy of the corresponding models without each feature. Other ML method: Lasso, with L1, it can punish features, therefore, it is well known for the ability to select good features.

Problems

During working, one mistake encountered is: data is not shuffled. The output of the host model and wrapper method got a huge conflict.

Conclusion

We have been through some methods in all the groups. It is difficult to conclude which ranking is better than others, however, from all the ranking results, we can tell that sensor 8 is important to classify classes.