# Forest Fires Prediction

Data Set(s): Algerian forest fires

Author(s): Jingxiang Zhang

Email Contacts: jzhang92@usc.edu

Date: April 23, 2022

## 1. Abstract

The fire happened frequently in Algeria in summer, so it is crucial to set up a model to predict the fire by using the sensor data. To find the best model, I will use the data over the period of 3 months as a training dataset, and 1 month as a test dataset. Use cross validation method to separate the training dataset.

Before training, I will show the data, and use Nearest Mean Classifier to compare 4 methods of data preprocessing, which include PCA, standardization, delete the night data, increase new features by the average/min/max of last N days. And find out that use standardization singly will get the best performance.

In the model selection and training part, I will try to use 4 models, which are Nearest Mean Classifier, KNN, SVM, and bayes. Compare the F1 score, accuracy, precision, recall, and confusion matrix for those models, and finally get the best performance by linear kernel SVM.

Python and its corresponding libraries Numpy, sklearn, pandas, json, and scipy will be used in this problem.

**Key words:** Classification, KNN, SVM, bayes, python.

## 2. Introduction

At the very beginning, I believe it is necessary to analyze all the features of the data. This data set has 10 features, which are listed as follows:

1）Temperature
2）RH: Humidity
3）Ws: wind speed
4）Rain: rain level
5）FFMC: Fine Fuel Moisture Code (indicator of the relative ease of ignition)
6）DMC: Duff Moisture Code (same as 6)
7）DC: Drought Code (same as 6)
8）ISI: Initial Spread Index (the ability of a fire to spread)
9）BUI: Buildup Index (amount of fuel available for combustion)

From the introduction of dataset, we can draw a conclusion that nearly all features have a positive correlation with fire, except for the date, humidity, and rain level. The plot below shows whether fire happened or not corresponding to each feature. It confirms our suspect, but the data is not linear separable

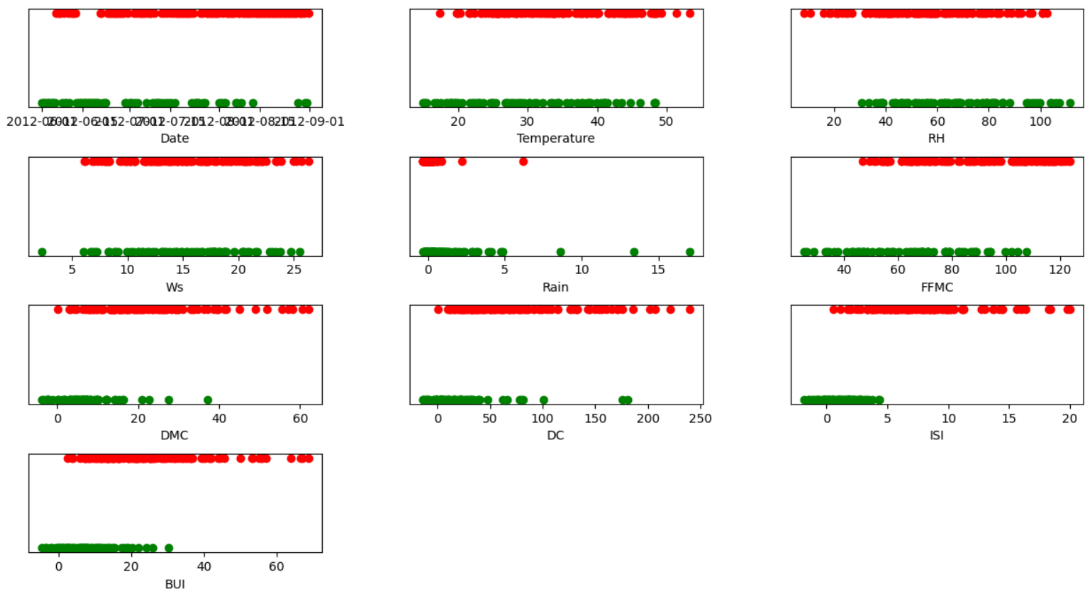

Figure 1.1: fire or no fire (red stand for fire, green stand for no fire)

Then, it is necessary to see the correlation between any two features. The scatter plot demonstrates that many features are interrelated to each other.

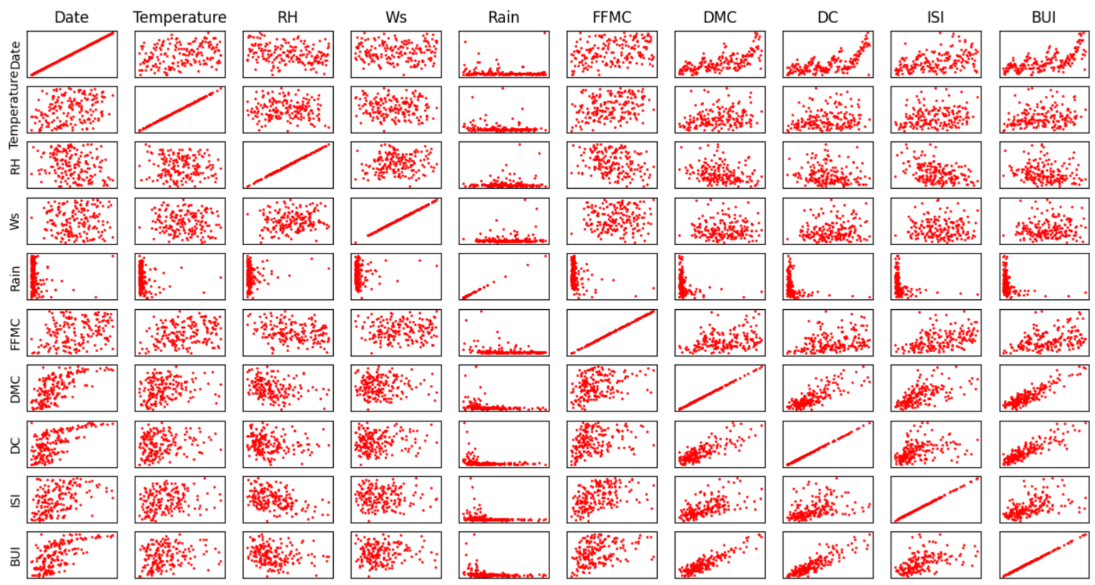

Figure 1.2: correlation between features

| | Temperature | RH | Ws | Rain | FFMC | DMC | DC | ISI | BUI |
|---|---|---|---|---|---|---|---|---|---|
| Temperature | 1.000 | -0.056 | -0.077 | -0.143 | 0.294 | 0.222 | 0.172 | 0.217 | 0.211 |
| RH | -0.056 | 1.000 | 0.133 | 0.144 | -0.253 | -0.279 | -0.140 | -0.421 | -0.200 |
| Ws | -0.077 | 0.133 | 1.000 | 0.107 | 0.042 | 0.006 | 0.129 | -0.012 | 0.060 |
| Rain | -0.143 | 0.144 | 0.107 | 1.000 | -0.293 | -0.260 | -0.270 | -0.286 | -0.290 |
| FFMC | 0.294 | -0.253 | 0.042 | -0.293 | 1.000 | 0.327 | 0.301 | 0.432 | 0.373 |
| DMC | 0.222 | -0.279 | 0.006 | -0.260 | 0.327 | 1.000 | 0.803 | 0.591 | 0.895 |
| DC | 0.172 | -0.140 | 0.129 | -0.270 | 0.301 | 0.803 | 1.000 | 0.388 | 0.876 |
| ISI | 0.217 | -0.421 | -0.012 | -0.286 | 0.432 | 0.591 | 0.388 | 1.000 | 0.520 |
| BUI | 0.211 | -0.200 | 0.060 | -0.290 | 0.373 | 0.895 | 0.876 | 0.520 | 1.000 |

correlation coefficient matrix

Here, from the dataset information, I conclude a technology roadmap of this report.

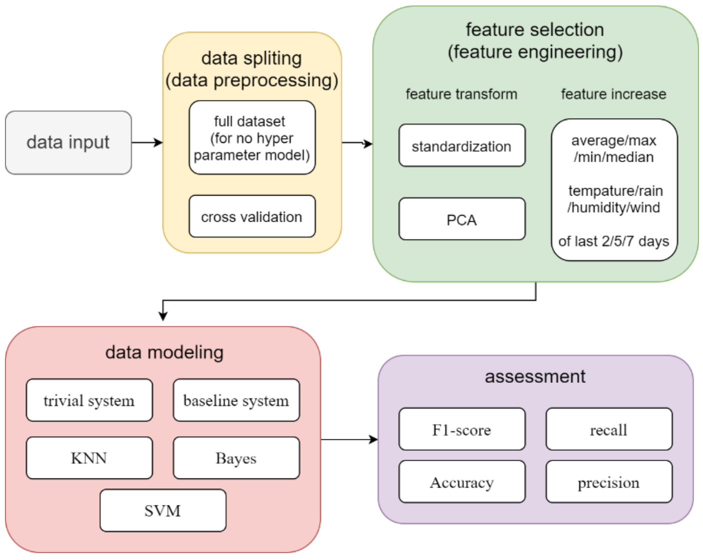

Figure1.3

Before data preprocessing, I will use the trivial model and baseline model to do the classification.

| model | precision | recall | accuracy | F1_Score | TP | TN | FP | FN |
|---|---|---|---|---|---|---|---|---|
| trivial | 0. 415 | 0. 739 | 0. 500 | 0. 177 | 17 | 13 | 24 | 6 |
| baseline | 0. 917 | 0. 478 | 0. 783 | 0. 306 | 11 | 36 | 1 | 12 |

## 3. Data Prepressing, Cross Validation and Complexity Analysis

### 3.1. Cross Validation

I will use cross validation. Split the dataset into 3 groups, corresponding to 3 mouths, and train the model 3 times and pick the best performance hyper parameter for each model.

### 3.2. Data Prepressing

I will use four methods of data prepressing, which are standardization, PCA, adding new features, and separate the data by day and night. In PCA, I will keep the dimensions that can add up to more than 90% of contribution rate. And in adding new features method, I will create new features based on the original feature, such as Temperature or Humidity. Use the average/minimal/maximal of the last N days separate the daytime and night. By using this method, I will try to drop the original features and replace it by the new feature, or keep the original features. When using the last N days information, I will drop the last N days in the training data set, because this information will be used in test data set.

First, there are two data in each day, one has a relative low temperature, and another has a relative high temperature. I suppose the first one is measured in the midnight, and the second is measured at noon, and the original data are listed as follow (first 5 lines):

|  | Date | Temperature | RH | Ws | Rain | FFMC | DMC | DC | ISI | BUI |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2012-06-01 00:00:00 | 18. 95 | 43. 86 | 12. 29 | -0. 34 | 73. 06 | -2. 37 | 28. 54 | 0. 49 | 6. 23 |
| 1 | 2012-06-01 12:00:00 | 34. 50 | 46. 23 | 14. 84 | 0. 94 | 56. 03 | 0. 44 | -10. 01 | -1. 47 | 2. 27 |
| 2 | 2012-06-02 00:00:00 | 15. 26 | 57. 08 | 10. 97 | 1. 40 | 33. 11 | 3. 39 | -13. 77 | 1. 34 | -3. 33 |
| 3 | 2012-06-02 12:00:00 | 24. 85 | 99. 91 | 17. 92 | 3. 96 | 26. 15 | 5. 18 | -1. 39 | -0. 76 | 0. 13 |
| 4 | 2012-06-03 00:00:00 | 19. 38 | 104. 40 | 23. 74 | 13. 39 | 28. 66 | -2. 25 | -3. 43 | 0. 56 | -1. 93 |

This is the data after standardization.

|  | Date | Temperature(st) | RH(st) | Ws(st) | Rain(st) | FFMC(st) | DMC(st) | DC(st) | ISI(st) | BUI(st) |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2012-06-01 00:00:00 | -1. 53 | -0. 76 | -0. 62 | -0. 50 | -0. 25 | -1. 40 | -0. 50 | -1. 04 | -0. 82 |
| 1 | 2012-06-01 12:00:00 | 0. 16 | -0. 65 | -0. 12 | 0. 16 | -0. 95 | -1. 19 | -1. 22 | -1. 48 | -1. 08 |
| 2 | 2012-06-02 00:00:00 | -1. 94 | -0. 15 | -0. 89 | 0. 40 | -1. 88 | -0. 97 | -1. 29 | -0. 86 | -1. 45 |
| 3 | 2012-06-02 12:00:00 | -0. 89 | 1. 83 | 0. 50 | 1. 73 | -2. 16 | -0. 84 | -1. 06 | -1. 32 | -1. 22 |
| 4 | 2012-06-03 00:00:00 | -1. 49 | 2. 04 | 1. 65 | 6. 64 | -2. 06 | -1. 39 | -1. 10 | -1. 03 | -1. 36 |

PCA, keep the 90% of contribution rate, there will be 3 components.

|  | Date | com(0) | com(1) | com(2) |
|---|---|---|---|---|
| 0 | 2012-06-01 00:00:00 | -100. 62 | -15. 66 | 110. 54 |
| 1 | 2012-06-01 12:00:00 | -139. 48 | -21. 34 | 116. 27 |
| 2 | 2012-06-02 00:00:00 | -148. 97 | -46. 15 | 122. 11 |
| 3 | 2012-06-02 12:00:00 | -140. 31 | -77. 26 | 90. 30 |
| 4 | 2012-06-03 00:00:00 | -144. 26 | -78. 33 | 84. 88 |

Here is an example of using the data of last 5 days to create new features. There are 3 newly created features, by the original temperature, wind speed, and rain fall. The date begins at 5th in July, because the days before it cannot create such a feature. What's more, all the new feature is calculated daytime and night data separately, i.e., when calculate the temperature at 5th in the night, I will use all the data in the night.

| | Date | Temperature_AVE_5 | Ws_MAX_5 | Rain_MIN_5 |
|---|---|---|---|---|
| 0 | 2012-06-05 00:00:00 | 19.87 | 23.74 | -0.34 |
| 1 | 2012-06-05 12:00:00 | 36.03 | 20.32 | -0.22 |
| 2 | 2012-06-06 00:00:00 | 20.86 | 23.74 | -0.34 |
| 3 | 2012-06-06 12:00:00 | 37.37 | 20.32 | -0.22 |
| 4 | 2012-06-07 00:00:00 | 22.52 | 23.74 | -0.25 |

The final data preprocessing method, I delete the night data. In this problem, we want to figure out whether it will have fire or not, I suppose that the fire is more likely to happened at daytime. Therefore, I delete the entire night data, only keep the daytime data.

| | Date | Temperature_AVE_5 | Ws_MAX_5 | Rain_MIN_5 |
|---|---|---|---|---|
| 0 | 2012-06-05 12:00:00 | 36.03 | 20.32 | -0.22 |
| 1 | 2012-06-06 12:00:00 | 37.37 | 20.32 | -0.22 |
| 2 | 2012-06-07 12:00:00 | 36.29 | 20.32 | -0.22 |
| 3 | 2012-06-08 12:00:00 | 37.87 | 20.59 | -0.24 |
| 4 | 2012-06-09 12:00:00 | 34.75 | 20.59 | -0.24 |

### 3.3. Complexity Analysis

Due to different data preprocessing strategy, the totally possible combination of training models would grow exponentially. Whether to standardization or not are two possible choices. Whether to use cross validation or not are two possible choices. Whether to drop the night data or not are two choices. Whether use PCA or not, and use what contribution rate, there will be A possible choices. Adding what type of features and how many types of features, there will be B possible choices, using MAX/MIN/AVE will be 3 choices, and used last N days, N is another variable. Therefore, regardless of hyperparameter in the learning process, there will be 2*2*A*B*3*N possible choices, that's tons of choices. To minimize the workload, I will assume that in each learning model, those variables are all convex. That is, if I use SVM model, and the accuracy of using cross validation is better than not, then I will assume that whatever strategy I use, I will keep cross validation when splitting the data.

## 4. Model Selection

In this part, I will use linear classification, KNN, SVM, Bayes and ANN five models, to find the models that have the relatively best performance.

## 4.1. Nearest Mean Classification

The difference between linear models and baseline models is that, in the linear models, I will use all the data preprocessing strategy. First, I will use cross validation to find the best models (use F1 score as the only criteria). Then, I will use the best models to train the whole dataset, and check the result on the test set.

Here is the result by cross validation (not using test data currently). In the form below, we can find that to get the best.

| model | preprocess | precision | recall | accuracy | F1_Score | TP | TN | FP | FN |
|---|---|---|---|---|---|---|---|---|---|
| baseline | | 0.917 | 0.478 | 0.783 | 0.306 | 11 | 36 | 1 | 12 |
| linear | | 0.902 | 0.920 | 0.852 | 0.756 | 46 | 6 | 5 | 4 |
| linear | 'PCA' | 0.902 | 0.920 | 0.852 | 0.756 | 46 | 6 | 5 | 4 |
| linear | 'STD' | 0.906 | 0.960 | 0.885 | 0.811 | 48 | 6 | 5 | 2 |
| linear | 'DelNight' | 0.880 | 0.917 | 0.833 | 0.725 | 22 | 3 | 3 | 2 |
| linear | 'PCA', 'STD' | 0.902 | 0.920 | 0.852 | 0.756 | 46 | 6 | 5 | 4 |
| linear | 'PCA', 'DelNight' | 0.880 | 0.917 | 0.833 | 0.725 | 22 | 3 | 3 | 2 |
| linear | 'STD', 'DelNight' | 0.920 | 0.958 | 0.900 | 0.828 | 23 | 4 | 2 | 1 |
| linear | 'PCA', 'STD', 'DelNight' | 0.913 | 0.875 | 0.833 | 0.714 | 21 | 4 | 2 | 3 |

**Note that** in baseline system, TP and TN are completely upside down compare to other result. According to cross check, it is not a fault, because baseline system used the test set, while all the other result used the validation set.

But there is another tricky question. Because there are lots of fire in August, the data in August will be intolerable biased. And the F1_Score will be surpassing all the other data. Therefore, I will drop August data as validation set from now on.

| model | preprocess | precision | recall | accuracy | F1_Score | TP | TN | FP | FN |
|---|---|---|---|---|---|---|---|---|---|
| baseline | | 0.917 | 0.478 | 0.783 | 0.306 | 11 | 36 | 1 | 12 |
| linear | | 0.840 | 0.553 | 0.656 | 0.323 | 21 | 19 | 4 | 17 |
| linear | 'PCA' | 0.840 | 0.553 | 0.656 | 0.323 | 21 | 19 | 4 | 17 |
| **linear** | **'STD'** | **0.941** | **0.842** | **0.869** | **0.707** | **32** | **21** | **2** | **6** |
| linear | 'DelNight' | 0.750 | 0.500 | 0.600 | 0.234 | 9 | 9 | 3 | 9 |
| linear | 'PCA', 'STD' | 0.867 | 0.684 | 0.738 | 0.460 | 26 | 19 | 4 | 12 |
| linear | 'PCA', 'DelNight' | 0.750 | 0.500 | 0.600 | 0.234 | 9 | 9 | 3 | 9 |
| linear | 'STD', 'DelNight' | 0.929 | 0.722 | 0.800 | 0.554 | 13 | 11 | 1 | 5 |
| linear | 'PCA', 'STD', 'DelNight' | 0.857 | 0.667 | 0.733 | 0.435 | 12 | 10 | 2 | 6 |

The result show that only doing standardization as preprocessing method can get the best performance. And henceforth, **all the data input will be standardized in all the models after**.

Adding one new feature of the data after standardization. There are 9 original features, each new feature I will try maximal/minimal/average 3 methods, and in each method, I will use the last 2/3/4 days. There will be 9*3*3=81 total possible combination. Part of the result below show that this modify are so slight that can hardly change anything. I think the reason of it is that because there are 9 features previously, so modifying one feature a bit will do nothing important to the result.

| model | preprocess | precision | recall | accuracy | F1_Score | TP | TN | FP | FN |
|---|---|---|---|---|---|---|---|---|---|
| linear | Temperature(st)_AVE_2 | 0.941 | 0.842 | 0.869 | 0.707 | 32 | 21 | 2 | 6 |
| linear | Temperature(st)_AVE_3 | 0.941 | 0.842 | 0.869 | 0.707 | 32 | 21 | 2 | 6 |
| linear | Temperature(st)_AVE_4 | 0.941 | 0.842 | 0.869 | 0.707 | 32 | 21 | 2 | 6 |
| linear | Temperature(st)_MAX_2 | 0.941 | 0.842 | 0.869 | 0.707 | 32 | 21 | 2 | 6 |
| linear | Temperature(st)_MAX_3 | 0.941 | 0.842 | 0.869 | 0.707 | 32 | 21 | 2 | 6 |
| linear | Temperature(st)_MAX_4 | 0.941 | 0.842 | 0.869 | 0.707 | 32 | 21 | 2 | 6 |

Finally, let's check the data preprocessing method on the test set. It improves a little from the baseline.

| model | preprocess | precision | recall | accuracy | F1_Score | TP | TN | FP | FN |
|---|---|---|---|---|---|---|---|---|---|
| baseline | None | 0.917 | 0.478 | 0.783 | 0.306 | 11 | 36 | 1 | 12 |
| linear | std | 0.929 | 0.565 | 0.817 | 0.392 | 13 | 36 | 1 | 10 |

## 4.2. KNN (K-Nearest Neighbor)

From 3.1, we know that the best data preprocess method is standardization, and by test, it is also applied to this model. There, to choose the best numbers of neighbor, I standardize the data, and test the number of N of validation set, and get the result below. From the result, n = 13 will have the best performance.

| model | N | precision | recall | accuracy | F1_Score | TP | TN | FP | FN |
|---|---|---|---|---|---|---|---|---|---|
| KNN | 3 | 0.868 | 0.868 | 0.836 | 0.655 | 33 | 18 | 5 | 5 |
| KNN | 4 | 0.889 | 0.842 | 0.836 | 0.648 | 32 | 19 | 4 | 6 |
| KNN | 5 | 0.846 | 0.868 | 0.820 | 0.630 | 33 | 17 | 6 | 5 |
| KNN | 6 | 0.865 | 0.842 | 0.820 | 0.622 | 32 | 18 | 5 | 6 |
| KNN | 7 | 0.842 | 0.842 | 0.803 | 0.597 | 32 | 17 | 6 | 6 |
| KNN | 8 | 0.889 | 0.842 | 0.836 | 0.648 | 32 | 19 | 4 | 6 |
| KNN | 9 | 0.821 | 0.842 | 0.787 | 0.574 | 32 | 16 | 7 | 6 |
| KNN | 10 | 0.889 | 0.842 | 0.836 | 0.648 | 32 | 19 | 4 | 6 |
| KNN | 11 | 0.805 | 0.868 | 0.787 | 0.585 | 33 | 15 | 8 | 5 |
| KNN | 12 | 0.870 | 0.833 | 0.881 | 0.617 | 20 | 32 | 3 | 4 |
| **KNN** | **13** | **0.875** | **0.875** | **0.898** | **0.670** | **21** | **32** | **3** | **3** |
| KNN | 14 | 0.889 | 0.842 | 0.836 | 0.648 | 32 | 19 | 4 | 6 |
| KNN | 15 | 0.815 | 0.917 | 0.881 | 0.647 | 22 | 30 | 5 | 2 |
| KNN | 16 | 0.846 | 0.868 | 0.820 | 0.630 | 33 | 17 | 6 | 5 |

Using K=13 on the test set, compare to baseline model, we can analyze that although the accuracy of KNN is not much greater than baseline, it is more balance. Therefore, KNN get a better F1 score.

| model | preprocess | N | precision | recall | accuracy | F1_Score | TP | TN | FP | FN |
|---|---|---|---|---|---|---|---|---|---|---|
| baseline | None | None | 0.917 | 0.478 | 0.783 | 0.306 | 11 | 36 | 1 | 12 |
| KNN | std | 13 | 0.792 | 0.826 | 0.850 | 0.529 | 19 | 32 | 5 | 4 |

## 4.3. SVM

There are 4 types of kernels: Linear, RBF, Polynomial and Sigmoid, and 3 hyper parameters, corresponding to different kernels:

a) gamma: used in 'poly', 'RBF' and 'sigmoid', big gamma imply more support vector and over fitting;
b) degree: only used in 'poly', typically less or equal than 3.
c) coef0: used in 'poly' and 'sigmoid', lower coef0 will cause under fitting, and vice versa.

### 4.3.1.Polynomial Kernel

To find the relative best hyper parameter (use F1 score as the only criteria), I use the validation data set, and iterate gamma in the list 0.02, 0.03, 0.05, 0.07, 0.1, 0.15, 0.2, 0.3, 0.5, and coefficient in the list -0.5, 0, 0.5, and degree is 2 or 3. Here are the result, I only keep 8 group of the result with the best performance, and 3 with the worst performance.

| model | hyper parameter | precision | recall | accuracy | F1_Score | TP | TN | FP | FN |
|---|---|---|---|---|---|---|---|---|---|
| SVM | coef: 0.5, gamma: 0.02, degree: 2 | 0.919 | 0.895 | 0.885 | 0.746 | 34 | 20 | 3 | 4 |
| SVM | coef: 0.5, gamma: 0.02, degree: 3 | 0.919 | 0.895 | 0.885 | 0.746 | 34 | 20 | 3 | 4 |
| SVM | coef: 0.5, gamma: 0.03, degree: 2 | 0.919 | 0.895 | 0.885 | 0.746 | 34 | 20 | 3 | 4 |
| SVM | coef: 0.5, gamma: 0.03, degree: 3 | 0.919 | 0.895 | 0.885 | 0.746 | 34 | 20 | 3 | 4 |
| SVM | coef: 0.5, gamma: 0.05, degree: 2 | 0.919 | 0.895 | 0.885 | 0.746 | 34 | 20 | 3 | 4 |
| SVM | coef: -0.5, gamma: 0.02, degree: 3 | 0.943 | 0.868 | 0.885 | 0.742 | 33 | 21 | 2 | 5 |
| SVM | coef: -0.5, gamma: 0.03, degree: 3 | 0.943 | 0.868 | 0.885 | 0.742 | 33 | 21 | 2 | 5 |
| SVM | coef: 0.5, gamma: 0.05, degree: 3 | 0.880 | 0.917 | 0.915 | 0.725 | 22 | 32 | 3 | 2 |
| | | … | | | | | | | |
| SVM | coef: -0.5, gamma: 0.2, degree: 2 | 0.386 | 0.917 | 0.373 | 0.230 | 22 | 0 | 35 | 2 |
| SVM | coef: -0.5, gamma: 0.3, degree: 2 | 0.375 | 0.875 | 0.356 | 0.205 | 21 | 0 | 35 | 3 |
| SVM | coef: -0.5, gamma: 0.5, degree: 2 | 0.364 | 0.833 | 0.339 | 0.181 | 20 | 0 | 35 | 4 |

Among all the 9*3*2 = 54 groups of hyper parameters, I find that the degree of polynomial kernel is not important, and gamma from 0.01 to 0.05 will have the best performance (I use gamma equal to 0.01, and find that it has exact same result). And my conclusion is in polynomial kernel, to get the best performance, coefficient=0.5, gamma=0.03, and degree=3

### 4.3.2.RBF Kernel

The only hyper parameter in RBF kernel is gamma, and when the range of gamma from 0.007 to 0.03, the validation set will get the best result. Here I will use gamma equal to 0.02

| model | gamma | precision | recall | accuracy | F1_Score | TP | TN | FP | FN |
|---|---|---|---|---|---|---|---|---|---|
| SVM | 0.007 | 0.919 | 0.895 | 0.885 | 0.746 | 34 | 20 | 3 | 4 |
| SVM | 0.01 | 0.919 | 0.895 | 0.885 | 0.746 | 34 | 20 | 3 | 4 |
| SVM | 0.02 | 0.919 | 0.895 | 0.885 | 0.746 | 34 | 20 | 3 | 4 |
| SVM | 0.03 | 0.919 | 0.895 | 0.885 | 0.746 | 34 | 20 | 3 | 4 |
| SVM | 0.05 | 0.895 | 0.895 | 0.869 | 0.716 | 34 | 19 | 4 | 4 |
| SVM | 0.07 | 0.872 | 0.895 | 0.852 | 0.689 | 34 | 18 | 5 | 4 |
| SVM | 0.1 | 0.895 | 0.895 | 0.869 | 0.716 | 34 | 19 | 4 | 4 |
| SVM | 0.15 | 0.892 | 0.868 | 0.852 | 0.682 | 33 | 19 | 4 | 5 |

### 4.3.3.Sigmoid Kernel

Gamma and coefficient are two hyper parameters in Sigmoid kernel. The result on validation data set is:

| model | hyper parameter | precision | recall | accuracy | F1_Score | TP | TN | FP | FN |
|---|---|---|---|---|---|---|---|---|---|
| SVM | coef: 0, gamma: 0.03 | 0.944 | 0.895 | 0.902 | 0.777 | 34 | 21 | 2 | 4 |
| SVM | coef: 0, gamma: 0.05 | 0.944 | 0.895 | 0.902 | 0.777 | 34 | 21 | 2 | 4 |
| SVM | coef: 0, gamma: 0.07 | 0.944 | 0.895 | 0.902 | 0.777 | 34 | 21 | 2 | 4 |
| SVM | coef: 0, gamma: 0.1 | 0.944 | 0.895 | 0.902 | 0.777 | 34 | 21 | 2 | 4 |
| SVM | coef: 0, gamma: 0.15 | 0.944 | 0.895 | 0.902 | 0.777 | 34 | 21 | 2 | 4 |
|  |  | … |  |  |  |  |  |  |  |
| SVM | coef: 0.5, gamma: 0.5 | 0.816 | 0.816 | 0.770 | 0.543 | 31 | 16 | 7 | 7 |
| SVM | coef: -0.5, gamma: 0.005 | 0.714 | 0.921 | 0.721 | 0.538 | 35 | 9 | 14 | 3 |
| SVM | coef: 0.5, gamma: 0.005 | 0.700 | 0.921 | 0.705 | 0.523 | 35 | 8 | 15 | 3 |

Here, the best gamma I will use 0.1, and coefficient is 0.

### 4.3.4.SVM conclusions

By using the best hyper parameter in each kernel method, I get the result below. The linear kernel has the best performance.

| model | preprocess | kernel | hyper parameter | precision | recall | accuracy | F1_Score | TP | TN | FP | FN |
|---|---|---|---|---|---|---|---|---|---|---|---|
| baseline | None |  |  | 0.92 | 0.48 | 0.78 | 0.31 | 11 | 36 | 1 | 12 |
| SVM | std | linear |  | 0.91 | 0.87 | 0.92 | 0.70 | 20 | 35 | 2 | 3 |
| SVM | std | polynomial | coef:0.5, gamma:0.03, degree:3 | 0.86 | 0.83 | 0.88 | 0.60 | 19 | 34 | 3 | 4 |
| SVM | std | RBF | gamma:0.02 | 0.86 | 0.83 | 0.88 | 0.60 | 19 | 34 | 3 | 4 |
| SVM | std | sigmoid | gamma:0.1 | 0.90 | 0.78 | 0.88 | 0.59 | 18 | 35 | 2 | 5 |

Some experience shows that, if there are lots of features, then linear kernel would be a good choice. If the number of features is small, then radio base function would be a good choice. If the number of features is small, will in the same time, the size of dataset is huge, then you'd better increase some more features.

## 4.4. Bayes

In this problem, I assume each feature in the data satisfy Gaussian distribution. Therefore, Gaussian distribution can be used to estimate the probability density of each feature. I will use Gaussian naive Bayes to do this classification.

| model | precision | recall | accuracy | F1_Score | TP | TN | FP | FN |
|---|---|---|---|---|---|---|---|---|
| baseline | 0.917 | 0.478 | 0.783 | 0.306 | 11 | 36 | 1 | 12 |
| Bayes | 0.895 | 0.739 | 0.867 | 0.540 | 17 | 35 | 2 | 6 |

## 5. Libraries used and coded introduction

Libraries used: Numpy, sklearn, pandas, json, and scipy.

There is total 6 parts (folders) in my code, which are data preprocessing, classification, models, assessment, show, and root dictionary.

1) In data preprocessing folder, I have two python files: read.py for reading the data, preprocess.py for data preprocessing. Data preprocess part include standardization function, PCA decomposition, delete night data function, automatic yield the combination of all the data preprocessing function, and increase new features, cross validation function.
2) In the classification folder, there is a base class, which include some common function like automatic save the learning result and load result. All the classification models need to derive from it. SVM, bayes, KNN, trivial_and_base classification models are included in the folder.
3) In the models folder, there are some json files. It is created by classification methods and used to save the parameters.
4) In the show folder, it provides some plot function and export to excel function, to show the result of each process.
5) Finally in the root dictionary, there are main function of testing data preprocessing methods, and train each model.

## 6. Contributions of each team member

No team members, I do this project myself.

## 7. Summary and conclusions

Among all the data preprocessing methods, all the models, and all the hyper parameter I used in the problem, the best choice is standardizing the data with no extra feature, and use soft margin linear kernel SVM to classify the data. And here is the result of trivial model, baseline model, and linear kernel SVM model running on the test set.

| model | preprocess | precision | recall | accuracy | F1_Score | TP | TN | FP | FN |
|---|---|---|---|---|---|---|---|---|---|
| trivial | None | 0.415 | 0.739 | 0.500 | 0.177 | 17 | 13 | 24 | 6 |
| baseline | None | 0.917 | 0.478 | 0.783 | 0.306 | 11 | 36 | 1 | 12 |
| SVM(linear) | std | 0.900 | 0.780 | 0.880 | 0.590 | 18 | 35 | 2 | 5 |

Those three models do not have any hyperparameter, therefore it is meaningless to use cross validation to adjust the model.