Finding the most suitable classifier for wine quality prediction

SVM for wine type and 1NN for wine quality

Abstract

The reports starts with describing the main task of this project which is to find the most suitable classifer in predicting wine quality given training and test datasets. We then describe some main methods used in this project. In experiments part, we describe the training process in more detail and because of the finding nature, some comparing discussions are included in order to give direction of the next step in the experiment. The final result is presented in the result part and then come to a conclusion that wine type use SVM without PCA and wine quality use 1NN without PCA and train 1NN depending on wine type if only prediction accuracy is concerned.

I. Introduction

The purpose of this experiment is to find most suitable classifer to predict wine type and quality given a classic dataset on Portugese 'Vinho Verde' wines which has 11 different chemical properties. The wine type includes white and red and the wine quality is classified by wine experts between 1 (good) and 7 (bad).
The given dataset is divided into 5000 training samples and 1000 test samples and stored in two *.csv* files. Each of them has 13 columns including 11 chemical measures and two columns depicting wine type and quality. Thus we can use training samples to fit models and validate those models and report their prediction perfomances with the test samples.
In this experiment, we tried different models with training dataset. In order to decide whether PCA should be used, the trainings were performed twice, one with PCA and another with original data. Once we finish training and get models, we use those obtained models to predict on the test dataset, and then compare the predictions with the provided labels to evaluate those models and choose the most suitable one. In picking models section, we use prediction accuracies as a criteria to evaluate trained models in order to reduce computation. Prediciton accuracy is percentage of the right prediction. After selecting a prefered model, we then examine the model in more detail to see if we can get better results, in this phase F-Score is employed to evaluate the model performance more precisely.
Our main task is to make predictions as accurate as posssible, thus when comparing different models, we use prediction accuracy as only criteria without considering other factors such as computation cost.

II. Methods

I. Principle Component Analysis(PCA)

The main function of PCA is to project input data of d dimemsion into a lower k ($k < d$) dimensional and linear space with minimized information loss.
For example, when projecting x to z on the direction of w: $z = w^T x$, so the task is to find

w such that $Var(z)$ is maximized.

$$Var(z) = Var(w^T x) = E[(w^T x - w^T u)^2] \\ = w^T E[(x-u)(x-u)^T]w = w^T \sum w \quad (1)$$

Where

$$\sum = Var(x) = E[(x-u)(x-u)^T] \quad (2)$$

Then maximize $Var(z)$ subject to $||w|| = 1$

$$max w_1^T \sum w_1 - a(w_1 w_1^T - 1) \quad (3)$$

$\sum w_1 = a w_1$, w_1 is an eigenvalue of $\sum$.
To make $Var(z)$ max, choose the largest eigenvalue.
Similiarly, choose the second largest eigenvalue for the second principal component.
From above, we know that columns of W are eigenvalues of $\sum$, so in order to get w, all we nee to do is calculate the variance of original variables.
Then the question comes to which k is best? Usually Proportion of Variance (PoV) is used to select a suitable k.

$$Pov = \frac{\lambda_1 + \lambda_2 + ... + \lambda_k}{\lambda_1 + \lambda_2 + ... + \lambda_k + ... + \lambda_d} \quad (4)$$

Typically, stop at PoV>0.9 **[1]**.

II. naive Bayes classifier

Assume there are two classes $C = 1$, $C = 0$ and $P(C|x)$ is known, Bayes' classifier is as following:

$$Choose \begin{cases} C = 1, \text{if P(C=1 | x) > P(C=0 | x)} \\ C = 0, \text{otherwise} \end{cases} \quad (5)$$

But normally, we know the class probabilities $P(C)$ and the probability $P(x|C)$ of seeing x as the input when we know the class C. In this case, Bayes' rule is used to compute the classification probabilities. Bayes' rule is as following:

$$P(C|x) = \frac{P(x|C) \times P(C)}{P(x)} \quad (6)$$

In training process, $P(C)$ can be obtained from sample data, $P(x|C)$ can use different types of distributions to get different Bayes classifiers and we can select the one with smallest validation error.
Bayes' classifier can also be used to classify k classes. The modified Bayes' rule for k classes is as following:

$$P(C_i|x) = \frac{P(x|C_i)P(C_i)}{P(x)} = \frac{P(x|C_i)P(C_i)}{\sum_{k=1}^{K} P(x|C_k)P(C_k)} \quad (7)$$

Where

$$P(C_i) \geq 0 \text{ and } \sum_{k=1}^{K} P(C_k) = 1 \quad (8)$$

Bayes Classifier for k classes is modified to be: choose C_k where $k = argmax_k P(C_k|x)$. **[2]**

III. Supportive Vector Machines(SVM)

Supportive vector machines is a discriminative classifier defined by a separating hyperplane. It is a supervised learning model, given labeled training data, the algorithm outputs an optimal hyperplane which can then be used to categorizes new sample data. The SVM algorithm is based on finding a hyperplane that gives the largest minimum distance to the training examples**[3]**.
SVM is two-class classifier, for multiple classes such as wine quality, multisvm must be used. multisvm was trained with one-against-all approach. One-against-all approach builds as many binary classifiers as there are classes, each trained to separate one class from the rest. To predict a new instance, multisvm iterate each classifier until the first classifier who assign the new instance as its class member is found.

IV. kNN

kNN is a nonparametric method. The basic principal is: find k nearest neighbors of x and

then predict the class of x to be the majority class of the k nearest neighbors. k can be determined by validation such as k_Fold corss validation.**[4]**

V. F-Score

In this experiment, F-score is employed to measure the accuracy of the prediction. Precision p and recall r are used to compute the score. Let n_tp be the number of true positives; n_fp the number of false positives; n_tn the number of true negtive and n_fn the number of false negtive, then

$$p = \frac{n\_tp}{n\_tp + n\_fp} \text{ and } r = \frac{n\_tp}{n\_tp + n\_fn} \quad (9)$$

And F-score is calcualted as folloing **[5]**:

$$F = 2 \times \frac{precision \times recall}{precision + recall} \quad (10)$$

III. Experiments

I. Preprocess data

The first thing to do is to load data into matlab workspace from file storing data and convert categorical variables such as type colum into nominal arrays.
Before training models, it is good to examine correlations between features. To get a first impression on the relationship among different features and their types, use *gplotmatrix* to create a matrix of scatter plots. See the outcome in Figure 1.

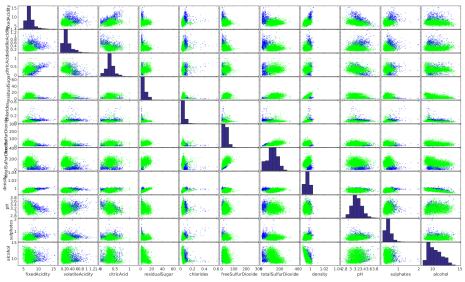

Figure 1: *Matrix of scatter plots on 11 features group by wine type*

Figure 1 shows that some of the two combinations such as alcohol and totalSulfurDioxide can divide the type quite well. So we can select those combinations to test and see if it can output good results.
Another choise would be PCA. Set $PoV = 0.98$, we got $k = 2$.
Figure 2 shows intuitively how feature selection and PCA can perform on separating wine types.

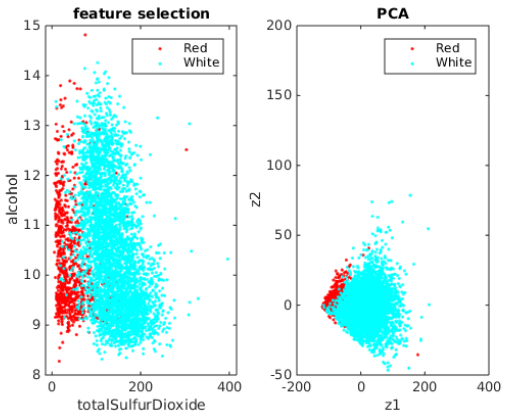

Figure 2: *Feature selection vs PCA*

From Figure 2, we can see both feature selection and PCA seems to be able to well separate wine types, thus we should compare them more precisely in the following steps.

II. Fit models for wine type

To find the most suitable model for wine type, Naive Bayesian, SVM, kNN and desicion tree are tried and implemented with both feature selection and PCA. Firgure 3 shows their prediction accuracy.

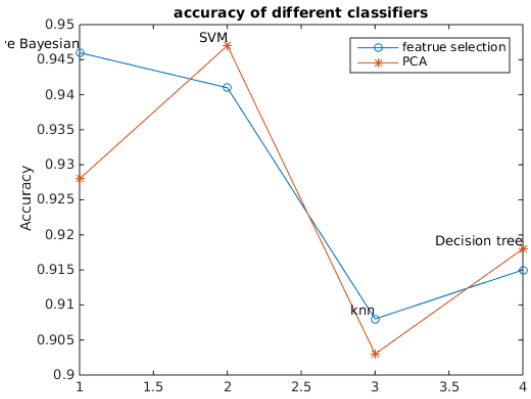

Figure 3: *Accuracies of different models with feature selection and PCA*

Figure 3 shows that Naive Bayesian and SVM classifier is obviously better than kNN and desicion tree. Thus kNN and decision tree can be skiped now and Naive Bayes and SVM should be studied in more detail. And from now on, F-Scores are employed to evaluate model performance. The matlab implementation of F-Score can be found in Appendix A.
Moreover, Naive Bayesian with feature selection is better than with PCA, but SVM performs better with PCA than with feature selection, so lets compare Naive Bayes with feature selection and SVM with PCA. Table 1 shows the comparision.

Table 1: *NB-feature selection vs SVM-PCA*

model	**White**	**Red**
NB - feature selection	0.96683	85484
SVM - PCA	0.96758	0.85479

From Table 1, we know that SVM PCA is a little better than Naive Bayes with feature selection.
So considering to best estimate wine type, SVM with PCA is the most suitable classifier and thus it is selected as our final model to predict the wine type. The next step is to examine SVM in more detail.
PCA has many advantages such as reducing computation cost, but if our goal is to improve accuracy as much as possible, then the performance of SVM without PCA should also be tried. Besides 11 dimemsion is acceptable due to the strong computation ability of our computer.
Table 2 shows the comparision result of training SVM model with original 11 features and that of PCA. From Table 2, we know that original 11 features has much higher F-Scores especially for red type than that of PCA. Thus SVM with original features datasets is selected if only prediction accuracy is concerned.

Table 2: *SVM F-Scores with PCA vs SVM F-Scores with original featrues*

SVM	White	Red
PCA	0.9676	0.8548
Original	0.9963	0.9848

Now SVM with original features is selected. The above training process use default parameter values. But different parameter values of $fitcsvm$ should also be tried to see if better performance can be achieved.
First set *Standardize* flag to be true and then train the data to get svm model, F-Scores shows that there is no difference between Standardize *true* or Standardize $false$(default).
Then different *KernelFunction* such as *polynomial* and *rbf* were also tried. Table 4 shows their F-Scores.
For *linear* SVM with original features, F-Scores are shown in Table 4 and 12 lines of related matlab code can be found in Appendix B.

III. Fit models for wine quality

In order to find a suitable classifer for wine quality, first train different models with and without PCA and compare them to get the best one.
Logistic regression with gerneralized linear model was first tried using matlab function *fitglm*. Different distributions such as *normal*, *poisson*, *gamma*, *inverse gaussian* were tested, and the result shows that *normal* distribution yields the highest prediction accuracy.
Naive Bayesian and mutisnm was then trained.
To train kNN classifer, we should find optimal k. K-Fold cross validation was employed to select the optimal k. Divide training dataset into 100 equally sized datasets. Tried k from 1 to 15, we get Figure 4 which shows that $k = 1$ is the best. Thus we choose 1 as k to train kNN model. Matlab implementation of K-Fold cross validation is in Appendix C.

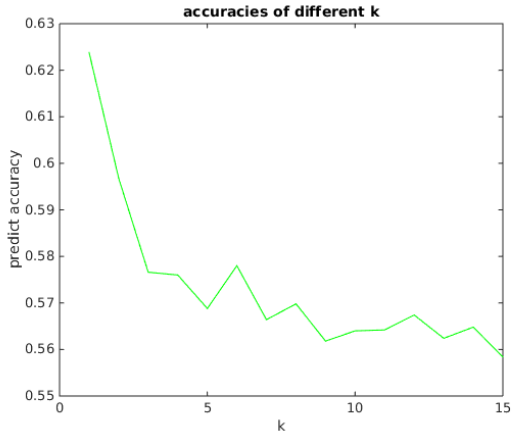

Figure 4: *Accuracies of different k*

The predict accuracies of the above trained classifiers are shown in Figure 5.

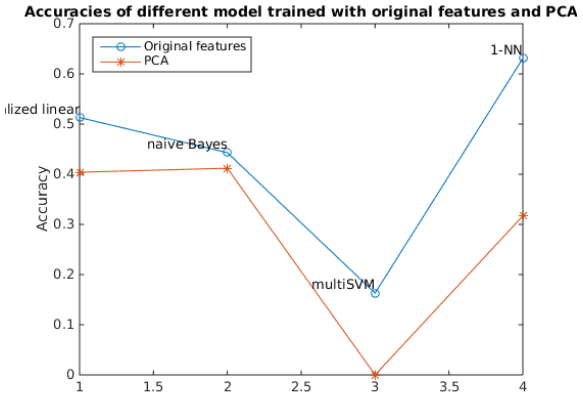

Figure 5: *Accuracies of different classifiers*

From Figure 5, we know that for wine quality, $1NN$ is the most suitable and perfoming PCA on the original features would deacrease the accuracy a lot, thus $1NN$ without PCA should be used.
Matlab implementation of $1NN$ without PCA is in Appendix D and its F-Scores are shown in Table 5.
Continue to if we can get better predict results. The relationship between 11 chemical features and wine quality may depend on wine type. And as we can see from wine type prediction section, the accuracy of predicting wine type with svm classifier is quite high that it is reasonable to train models for quality based on wine type, because we can predict wine type with high accuracy.
So we can train $1NN$ models for white wine and red wine separately without PCA.
To do this, first we divide trainng sample into white subset and red subset according to their wine type, and use those two subsets to train white and red $1NN$ models.
Next step is to divide the test sample into white and red test subsets according to their predicted wine type obtained from wine type predict section. And then predict quality of white wine with white $1NN$ model, and quality of red wine with red $1NN$ model.

Calculate F-Scores of white $1NN$ classifier and that of red $1NN$ classifier and compare them to that of training the whole data without division, we got Table 6. Matlab implementation is in Appendix E.

IV. Results

I. Fit models for wine type

Table 3: *SVM F-Scores with different KernelFunctions*

KernelFunction	White	Red
rbf	0.9034	0.2182
polynomial	0.0202	0.2096
linear	0.9963	0.9848

Table 4: *Final result F-Scores of SVM with original variables*

Type	White	Red
F-Score	0.9963	0.9848

II. Fit models for wine quality

Table 5: *Quality F-Scores of 1NN*

Quality	F-Score
1	0
2	0.4643
3	0.6295
4	0.6406
5	0.6686
6	0.3125
7	0

Table 6: *Quality F-Scores of white and red wine*

Quality	whole	white 1NN	red 1NN
1	0	0	0
2	0.46	0.50	0
3	0.63	0.64	0.64
4	0.64	0.67	0.59
5	0.67	0.66	0.72
6	0.31	0.34	0.15
7	0	0	0

V. Discussion

I. Fit models for wine type

During the experiment section, we compared different classifiers and came to a conclusion that SVM without PCA would be most suitable and Table 3 was obtained by trying different $KernelFunction$. Table 3 shows that the default KernelFunction *linear* is best. Thus we end up with using *linear* Kernel SVM without PCA whose F-Scores are shown in Table 4.
From Table 4, we can see that F-Scores of *linear* kernel SVM with original variables is quite high and we can say that our goal of finding a suitable classifier for wine type has been achieved.

II. Fit models for wine quality

We have already narrowed our choice to $1NN$ in the experiment section, now we want to compare more precisely $1NN$ trained on the whole dataset and $1NN$ trained depending on wine type.
From Table 6 we can see for white wine, F-Scores of white $1NN$ classifier is higher than that of whole training, as for red wine, red $1NN$ classifier has higher F-Scores where quality is 5 and 3, but lower in quality 6, 4 and 2. So it is hard to said if training classcifiers depending on wine type is better than training whole data or not. Also the low F-Scores of red $1NN$ wine classifer may due to the small number of red wine samples. But the differences is not that high, and if we want to make the training process simple, training one $1NN$ classifier on the whole data once would be most suitable.

References

[1] http://www.cmpe.boun.edu.tr/ ethem/i2ml2e/2e_v1-0/i2ml2e-chap6-v1-0.pdf
[2] https://noppa.aalto.fi/noppa/kurssi/t-61.3050/luennot/T-61_3050_lecture_4__bayesian_decision_theory.pdf
[3] http://web.mit.edu/zoya/www/SVM.pdf
[4] https://noppa.aalto.fi/noppa/kurssi/t-61.3050/luennot/T-61_3050_lecture_10__nonparametric_methods_and_decis
[5] http://wiki.eyewire.org/en/F-Scores_and_Accuracy

Appendix A

```
% calculates F-score given the labels, predictions and positive value
function [fscore] = fScore(labels, predictions, positive)
  n_tp = 0;
  n_tn = 0;
  n_fp = 0;
  n_fn = 0;
  for i=1:length(predictions)
    if (predictions(i) == positive)
        if (labels(i) == positive)
            n_tp = n_tp + 1;
        else
            n_fp = n_fp + 1;
        end
    else
        if(labels(i) == positive)
            n_fn = n_fn + 1;
        else
            n_tn = n_tn + 1;
        end
    end
  end
  fscore = computeFScore(n_tp, n_fp, n_fn);
end
```

Appendix B

```
 clear, clc, close all;
% Load training data
wine     = readtable('trainingdataset.csv');
wine     = table2dataset(wine);

% Load test data
testwine  = readtable('testdataset.csv');
testwine  = table2dataset(testwine);

% Convert categorical variables such as type colum into nominal arrays
wine      = ConvertCate(wine);
testwine = ConvertCate(testwine);

% fit svm model
CVSVMModel = fitcsvm(double(wine(:, 1:end-2)), wine.type);
Type_svm    = predict(CVSVMModel, double(testwine(:, 1:end-2)));
accuracy    = mean((double(Type_svm == testwine.type)));

wsc  = fScore(testwine.type, Type_svm, 'White');
rsc  = fScore(testwine.type, Type_svm, 'Red');
```

Appendix C

```
 function final_K = kcrossvalidation (wine, fold_num, k_num)

fold_size       = floor(size(wine, 1) / fold_num);
k_accuracies    = zeros(k_num, 1);

wine          = wine(randperm(size(wine,1)), :);
wine_quality  = double(wine(:, end-1));
wine_features = double(wine(:, 1:end-2));

for k=1:k_num
    accuracies = zeros(fold_num, 1);
    for f=1:fold_num
        validation_index = [1:fold_size] + fold_size*(f-1);
        training_index   = find(~ismember([1:size(wine_features,1)], validation_index))

        X_train = wine_features(training_index, :);
        y_train = wine_quality(training_index, :);

        X_vali = wine_features(validation_index, :);
        y_vali = wine_quality(validation_index, :);

        model = fitcknn(X_train, y_train, 'NumNeighbors', k, 'NSMethod','kdtree','Distan

        y_predict = predict(model, double(X_vali));
        accuracy = sum(y_predict == y_vali)/length(validation_index);
        accuracies(f) = accuracy;
    end
    k_accuracies(k) = mean(accuracies);
end

plot(1:k_num, k_accuracies,'g');
xlabel('k');
ylabel('predict accuracy');
title('accuracies of different k');

final_K = find(k_accuracies==max(k_accuracies));
```

Appendix D

```
 % must be run at matlab R2014b
% Clean window
clear ; close all; clc;

% load training data
wine = readtable('trainingdataset.csv');
wine = table2dataset(wine);

wine_quality  = double(wine(:, end-1));
wine_features = double(wine(:, 1:end-2));

% Load test data
testwine  = readtable('testdataset.csv');
testwine  = table2dataset(testwine);

test_features = double(testwine(:, 1:end-2));
test_quality  = double(testwine(:, end-1));

% k-cross validation to get a suitable k
fold_num = 100;
k_num    = 15;
K = kcrossvalidation(wine, fold_num, k_num);

% Predict with the best k
knn_model       = fitcknn(wine_features, wine_quality, 'NumNeighbors', K, 'NSMethod','k
predict_quality = predict(knn_model, test_features);

% Calculate fscores for each quality value
knn_fscores = qualityScores(test_quality, predict_quality);
```

Appendix E

```
% Clean window
clear ; close all; clc

% Load training data
wine = readtable('trainingdataset.csv');
wine = table2dataset(wine);

% Load test data
testwine  = readtable('testdataset.csv');
testwine  = table2dataset(testwine);

% Convert all the categorical variables such as type colum into nominal arrays
wine     = ConvertCate(wine);
testwine = ConvertCate(testwine);

% fit svm model
CVSVMModel = fitcsvm(double(wine(:, 1:end-2)), wine.type);
Type_svm = predict(CVSVMModel, double(testwine(:, 1:end-2)));

% Construct training data to be trained and tested
train_white_index = find(wine.type == 'White');
train_red_index   = find(wine.type == 'Red');

train_white_features = double(wine(train_white_index,1:end-2));
train_white_quality  = double(wine(train_white_index,end-1));

train_red_features = double(wine(train_red_index,1:end-2));
train_red_quality  = double(wine(train_red_index,end-1));

% predict the class and construct
test_white_index = find(Type_svm == 'White');
test_red_index   = find(Type_svm == 'Red');

test_white_features = double(testwine(test_white_index,1:end-2));
test_white_quality  = double(testwine(test_white_index,end-1));

test_red_features = double(testwine(test_red_index,1:end-2));
test_red_quality  = double(testwine(test_red_index,end-1));

% train with kNN for white wine
knn_white_model       = fitcknn(train_white_features, train_white_quality,'NumNeighbors
predict_white_quality = predict(knn_white_model, test_white_features);
```

```
white_knn_fscores     = qualityScores(test_white_quality, predict_white_quality);

% train with kNN for red wine
knn_red_model       = fitcknn(train_red_features, train_red_quality,'NumNeighbors', 1,
predict_red_quality = predict(knn_red_model, test_red_features);
red_knn_fscores     = qualityScores(test_red_quality, predict_red_quality);

% reconstruct the whole test data
testwinequality = [test_white_quality; test_red_quality];
testwinepredict = [predict_white_quality; predict_red_quality];

redwhite_knn_fscores     = qualityScores(testwinequality, testwinepredict);
```