Feature Selection and Accuracy Tradeoff

classification problem on ranking leauge of starcraft II players

Dennis Liu

Modified: August 16th, 2015 Compiled: August 16, 2015

1 Introduction

research problem

Since we have learnt many classification methods of machine learning, I want to apply them to a practical problem, classifying rank of game players based on their performance over many games.

Each palyer is placed into a league rank based on game records and the higher rank usually implies better proficiencies on the game. There are seven different leagues, each which contain a percent of the active player base (see table 1).

Table 1: Frequency Table of League Information

Rank	Bronze	Silver	Gold	Platinum	Diamond	Master	GrandMaster	Professional
Frequencies	167	347	553	811	806	621	35	55
Proportions(%)	5	10	16	24	23	18	1	2

Your rank will drop if you continue playing poor while continuing winning will promote you to a higher league. To simplify research problem, I grouped palyers whose rank is below 'Diamond' into 'low rank' class. How to judge a good player and see one's potential to be professional game player based on some features collected from game replay are of my interest in this paper.

features

There are 18 features in the originial dataset. Desciptions of them are seen in table 2.

Feature selection is helpful to screen out unimportant predictors and save computational resources. An interesting question that could be asked here is can we use less feature while preserving power to do prediction. How will training and testing accuracy change with regard to different number of features used in analysis. To evaluate these questions, widely-used classification methods we learnt throughout this course will be applied.

Table 2: Description of Features

Features	Descriptions
Age	Age of each player (integer)
HoursPerWeek	Reported hours spent playing per week (integer)
TotalHours	Reported total hours spent playing (integer)
APM	Action per minute (continuous)
SelectByHotkeys	Number of unit or building selections made using hotkeys per minutes
AssignToHotkeys	Number of units or buildings assigned to hotkeys per second
UniqueHotkeys	Number of unique hotkeys used per minutes
MinimapAttacks	Number of attack actions on minimap per second
MinimapRightClicks	number of right-clicks on minimap per second
NumberOfPACs	Number of PACs(shift of screen) per second
ActionLatency	Mean latency from the onset of a PACs to their first action in milliseconds
ActionsInPAC	Mean number of actions within each PAC
TotalMapExplored	The number of 24x24 game coordinate grids viewed by the player per minutes
WorkersMade	Number of SCVs, drones, and probes trained per second
UniqueUnitsMade	Unique unites made per second
ComplexUnitsMade	Number of ghosts, infestors, and high templars trained per second
ComplexAbilitiesUsed	Abilities requiring specific targeting instructions used per second

Methods

Data Source

The data comes from UCI machine learning repository. There are 3338 rows of information after taking off missing values and 18 different player's features.

The data were splited into three datasets, training, validating and testing data. The number of observations in each subset is shown in table 3.

Table 3: Data Used for Training, Validation, and Testing

Dataset	Observations	Class Frequencies (poor:well)	Class Proportions (%)
Training	834	458:376	55:45
Validating	835	483:352	58:42
Testing	1669	937:732	56:44

Before moving forward, I made correlation plot and make sure all features have high uniqueness and variance.

Classification Method

logistic classifier

The idea is to fit a logistic model to training set and use fitted model to predict the likelihood of class on new dataset.

$$e^{\beta_0+\beta_1 X} = \frac{p(X)}{1-p(X)}$$

support vector machine

Let us assume that we have n labeled examples $(x_1, y_1), \ldots, (x_n, y_n)$ with labels $y_i \in \{1, -1\}$. We want to find the hyperplane $< w, x > +b = 0$ to maximize the distance between the boundary and the nearest data points of each class.

I have tried fitting three different kernels (i.e., linear, radial and polynomial) as well as lambda (cost parameter).

k nearest neighbor

Given x_q, take vote among its k nearest neighbors and assign class to which most of the neighbors belong to. Noted that ss number of training examples approaches ∞ and k gets large, approaches Bayes optimal

naive bayes

Naive Bayes assumption: all features are independent

$$P(a_1, a_2 \ldots a_n | v_j) = \prod_i P(a_i | v_j)$$

which gives Naive Bayes classifier: $v_{NB} = V \; where \; P(v_j) \prod_i P(a_i|v_j)$ is largest.

linear discriminant analysis

In LDA, we assume the underlying density function is Guassian and equal variance between data in different classes. We fit each data to the following discriminant equation to see which class it is most likely belonging to.

$$\delta_k(x) = \frac{\mu_k}{\sigma^2} x - \frac{\mu_k^2}{2\sigma^2} + \log \pi_k$$

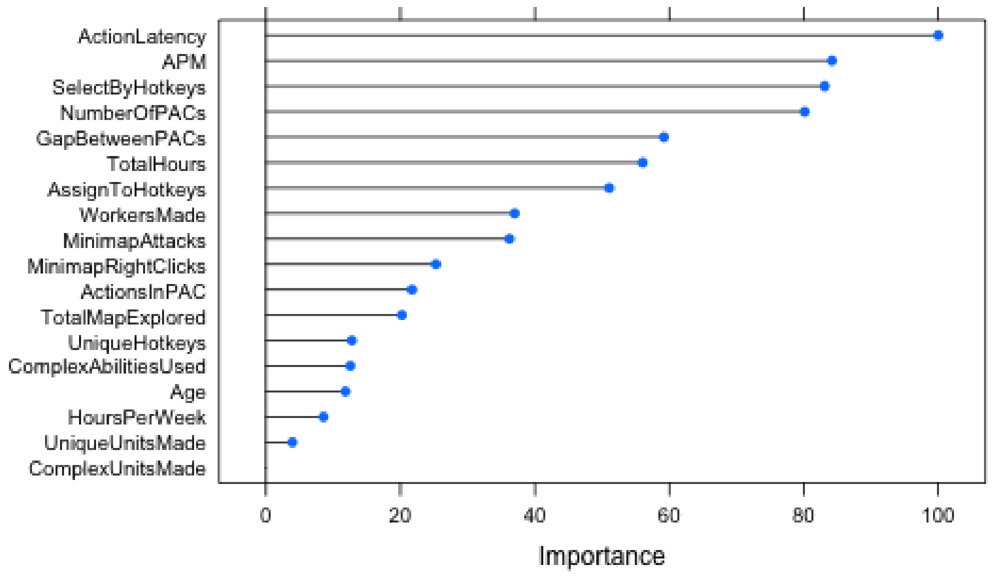

Figure 1: Variable Importance by Random Forest

quadratic discriminant analysis

Similar to LDA, except no assumptions on equal variances. We want to find such a k for which the following is largest.

$$\log \pi_k - (1/2\sigma_k^2)(x-\mu_k)^2 = -\frac{1}{2\sigma_k^2}x^2 + \frac{\mu_k}{\sigma_k^2}x - \frac{\mu_k^2}{2\sigma_k^2} - \log \sigma_k + \log \pi_k$$

Dimension Reduction with Random Forest

I am using an add-in feature of random forest function in R to select features by their importance. The idea is that if all features are irrelevant with response, then rearranging these features won't degrade prediction accuracy. However it does not taking into account high correlation between features.

The dataset has 18 features before running into analysis. I set 5 different number of features selected based on variable importance (see below).

Results

Variable Accuracy

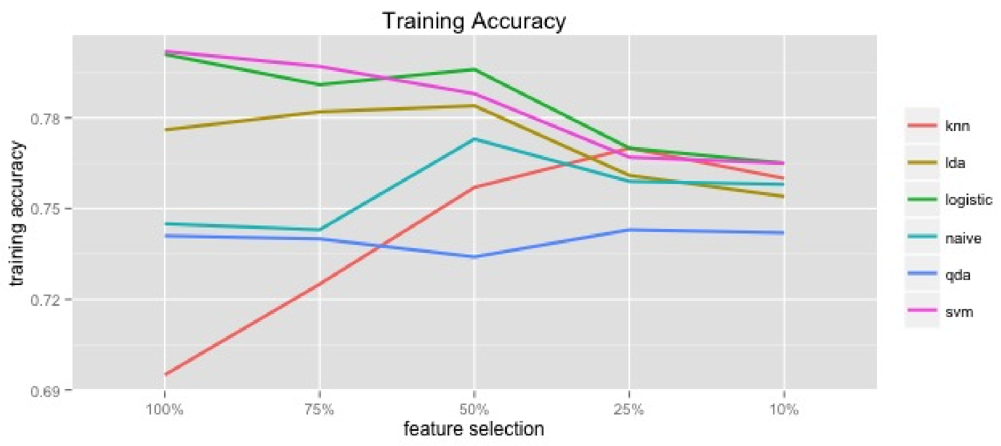

Figure 2: Training Accuracy with Different Model Complexity

training accuracy

The above graph shows how training accuracy respond to models with different amount of features and complexity. Features are selected by their importance. After trying different truncated dimensions of the original data, the training accuracies seem to converge between different methods as we have less and less features to fit.

The training performance does not necessarily get improved as we fit more features (i.e., in linear model, more features always implies higher training accuracy). To the opposite, the K-nearest method actually benefits from fitting less features.

Support vector machine and logistic classifier perform equally well regardless of how many features to fit although their accuracy decreased. K nearest neighbor outperforms quadratic discriminant analysis and naive bayes classifier when we use less than half of features.

testing accuracy

K nearst neighbor (KNN) has a bell-curve on its testing accuracy. All methods outperform KNN and their testing accuracy converge when we use fewer features.

Support vector machine and logistic classifier outperform other methods when we use full features and this difference is getting smaller. Since our training and testing accuracies are simlar, overfitting the data won't be an issue here.

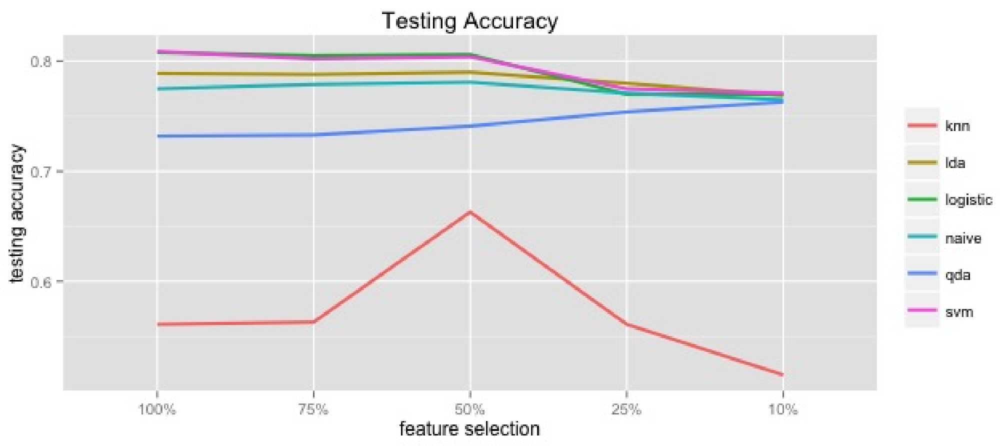

Figure 3: Testing Accuracy with Different Model Complexity

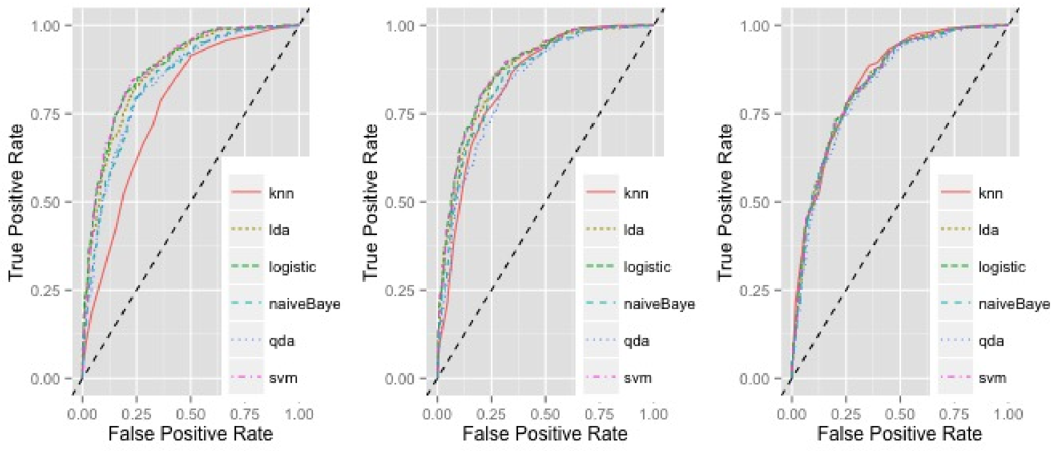

Figure 4: Training ROC Analysis (from left to right, less complex)

ROC analysis

training roc

Consistent with what we found, difference of perforance is decreasing as model becomes simple. Reducing half features (from plot1 to plot2) does not affect training accuracy significantly. With less computation, we can still gain high training accuracies.

All methods perform well when the false positive rate is under 25% but once it passes this threshold there is not too much additional gain in the true positive rate.

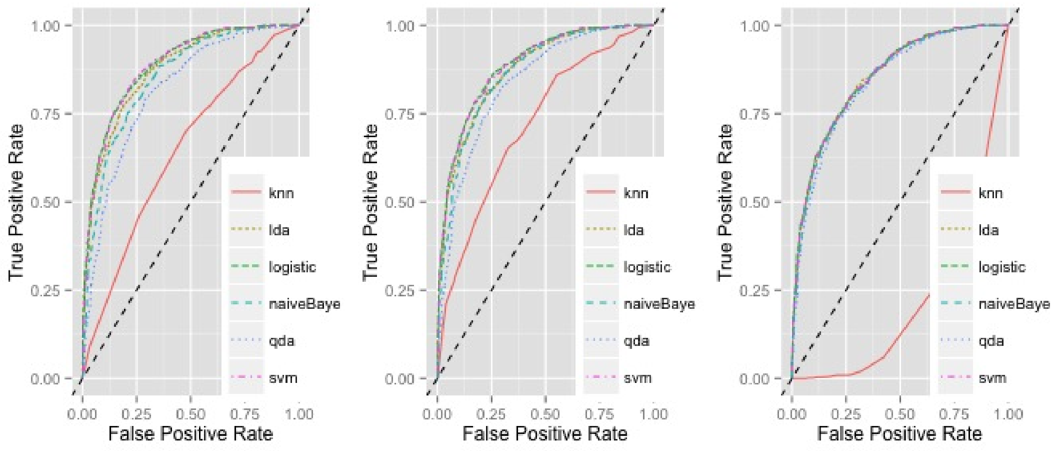

Figure 5: Testing ROC Analysis

testing roc

What's interesting here is the change of KNN testing accuracy, half feature selection is whre it finds its optimal performance. The ROC curve of other methods are all similar. The slope where the ROC curve approaches to 1 seems to be before 25% (but I don't know how to interpret it)

Discussion

Although using less features occationally bring benefit to both training and testing accuracies, overall it is detrimental to predictions. Support vector machine and logistic classifiers both perform well on these game data.

Possible future work includes:

- using other feature selection method, such as Principle Component Analysis and Singular Value Decomposition. Variable importance based selection won't be able to deal with highly correlated variables.
- incorporating randomforest, boosting and neural network to analysis to improve prediction accuracies. I was going to use them in my paper but it involves too much computation and iteration.