스타크래프트2 데이터를 이용한 분석 및 분류

기계학습 및 실습 기말 과제

통계학과 2015580011 박세혁
통계학과 2015580023 이동균
통계학과 2015580032 천준호

목차

1.서론

(1) 연구주제

현재 사회에서는 모든 분야에서 데이터 분석을 사용하고 있으며, 이는 E-sports 분야에서도 마찬가지이다. 현존하는 대부분의 e-sport팀에는 데이터 분석가들이 존재하여, 그 팀 선수뿐만 아니라 다른 팀의 모든 선수들의 데이터를 분석하여 팀 또는 개인이 승리하기 위해 도움이 되는 정보로 만들어 내는 역할을 하고 있다. 실제 E-sports 선수들뿐만 아니라, 게임에 관심 있는 대부분의 사람들은 게임을 하면서 '이영호, Faker같은 한 게임의 최고 수준에 도달한 선수들은 무엇을 잘하길래 게임을 저렇게 잘하는 것일까?'라는 생각을 해본 적이 있을 것이다. 그래서 사람들의 실력에 따라서 티어가 분류되는 게임의 데이터를 가지고 어떠한 차이가 상위 티어와 하위 티어를 분류하는 차이가 되는지 알아보려고 한다. 그리고 이와 더불어 데이터를 분류하는 여러가지 방법들 중 어떤 방법이 이 데이터에 가장 적합한 분류 방법인지도 알아볼 것이다.

(2) 데이터 획득방법

해당 데이터셋은 Kaggle에서 획득하였으며, 이는 Simon Fraser University의 심리학과에서 2013년 10월 20일 얻어진 데이터이다. 데이터 획득에 참여한 사람들로는 Mark Blair, Joe Thompson, Andrew Henrey, 그리고 Bill Chen이 있다.
주소: https://www.kaggle.com/sfu-summit/starcraft-ii-replay-analysis

(3) 데이터 설명

데이터는 20개의 변수로 총 3396개의 유저의 데이터가 있다. 우리의 목표는 어떤 변수가 상위랭커와 일반유저들의 랭크 차이를 만드는지 알아보기 위함 이므로, 반응변수로 게임랭크(LeagueIndex), 나머지 19개의 변수를 설명변수로 설정하였다.

먼저 반응변수LeagueIndex는 유저들의 티어를 나타내는 변수로 1부터 8까지의 정수로 이루어져 있다. 우리는 마스터(6), 그랜드마스터(7)의 유저를 상위티어, 나머지 브론즈(1), 실버(2), 골드(3), 플레티넘(4), 다이아몬드(5)를 하위티어라고 반응변수를 새롭게 정의하였다. 나머지 프로리그 티어를 나타내는 8은 결측치가 많아 제외한다.

브론즈(1)	하위티어(0)
실버(2)	
골드(3)	
플레티넘(4)	
다이아몬드(5)	
마스터(6)	상위티어(1)
그랜드마스터(7)	
프로페셔널리그(8)	결측치

나머지 19개의 설명변수들을 살펴보면 GameID, Age, Hours per Week, Total Hours, APM, Select by Hotkeys, Assign to hotkeys, Unique hotkeys, Minimap Attacks, Minimap Right clicks, Number of Pacs, Gap between Pacs, Action Latency, Action in Pac, Total map explored, Workers Made, Unique Units Made, Complex Units Made 그리고 Complex Ability Used로 이루어져 있다.
변수들에 대한 설명은 아래와 같다.

1.GameID: 유저 구분을 위한 변수 (필요하지 않다고 생각하여 제외하고 분석)
2.Age: 유저의 나이
3.Hours per Week: 일주일당 게임 이용시간
4.Total Hours: 지금까지 총게임 플레이 시간
5.APM(Action Per Minute): 분당 액션수
6.Select by Hotkeys: 시간단위당 단축키를 이용해 만든 유닛선택의 수
7.Assign to hotkeys: 시간단위당 단축키에 할당된 유닛의 수
8.UniqueHotkeys: 시간단위당 한번만 사용한 단축키의 수
9.MinimapAttacks: 시간단위당 미니맵으로 공격한 수
10.MinimapRightClicks: 시간단위당 미니맵에 오른쪽클릭(마우스) 수
11.NumberOfPACs: 시간단위당 PAC의 수
12.GapBetweenPACs: PAC사이의 평균시간
13.ActionLatency: PAC시작부터 첫 번째 액션까지의 평균대기시간이며 단위는 ms이다
14.ActionInPAC: PAC당 평균 액션횟수
15.TotalMapExplored: 시간단위당 플레이어가 조회한 24*24 게임좌표그리드 수
16.WorkersMade: 시간단위당 생산한 일꾼유닛의 수
17.UniqueUnitsMade: 시간단위당 생산한 유니크유닛의 수
18.ComplexUnitsmade: 시간단위당 생산한 마법유닛의 수
19.ComplexAbilityused: 시간단위당 시전한 마법의 수

데이터분석에서는 GameID 설명변수 제거, 3395개중 프로페셔널리그티어의 유저들 제거 및, 기타 결측치를 제거하여 3338명의 유저를 대상으로, 1개의 반응변수(LeagueIndex), 18개의 설명변수를 이용하여 분석을 진행한다.

2.본론

(1) 분석 방법 소개

먼저 데이터를 표준화시킨 후, 로지스틱 회귀분석을 통하여 18개의 설명변수들이 반응변수(티어)에 얼마나 영향을 끼치는지에 대해 분석한다. 회귀모델을 구하는 방법은 logistic regression, logistic ridge regression, logistic lasso regression, 주성분 로지스틱 회귀분석 방법을 사용한다. 우선은 4가지 방법을 어떻게 적용하는지에 대해 설명하고 각 방법에 대한 k-fold cv를 비교하여 어떠한 모델을 선택하는 것이 가장 적절한지 선택한다. 그 후 그에 따른 결과를 분석한다. 그리고 KNN, 선형판별분석 등 다양한 분류 방법을 통해 오분류율을 비교하여 이 데이터는 어떤 분류 방법을 사용하여 분류하는 것이 가장 좋은지 비교해본다.

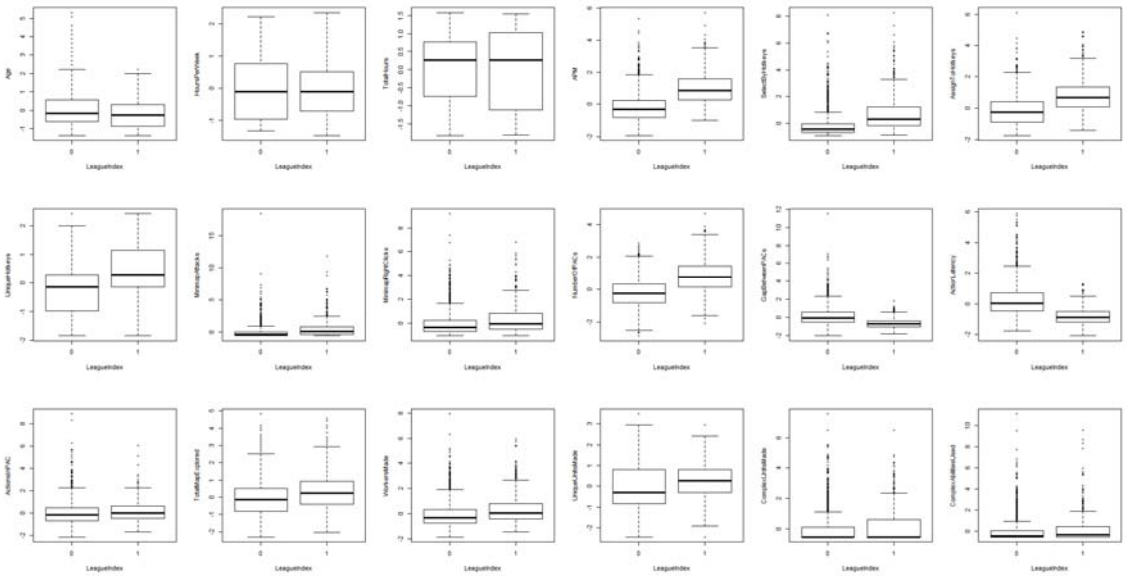
(2) 데이터 분석

데이터 분석을 시작하기 전에 데이터 표준화를 실시한다. 아래의 결과는 설명변수 데이터들을 표준화시킨 데이터의 summary이다.

```
      Age           HoursPerWeek       TotalHours           APM          SelectByHotkeys   AssignToHotkeys
 Min.   :-1.3433   Min.   :-1.4467   Min.   :-1.8096   Min.   :-1.9229   Min.   :-0.8512   Min.   :-1.73385
 1st Qu.:-0.6301   1st Qu.:-0.9571   1st Qu.:-0.7472   1st Qu.:-0.7346   1st Qu.:-0.5879   1st Qu.:-0.77334
 Median :-0.1546   Median :-0.1004   Median : 0.2578   Median :-0.1560   Median :-0.3339   Median :-0.07359
 Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.00000
 3rd Qu.: 0.5586   3rd Qu.: 0.7564   3rd Qu.: 0.7603   3rd Qu.: 0.5317   3rd Qu.: 0.1950   3rd Qu.: 0.61285
 Max.   : 5.3133   Max.   : 2.3476   Max.   : 1.5786   Max.   : 5.7212   Max.   : 8.2653   Max.   : 6.11436
 UniqueHotkeys     MinimapAttacks    MinimapRightClicks  NumberOfPACs      GapBetweenPACs    ActionLatency
 Min.   :-1.8499   Min.   :-0.5899   Min.   :-1.0577    Min.   :-2.85271   Min.   :-1.9961   Min.   :-2.0789
 1st Qu.:-0.5642   1st Qu.:-0.5899   1st Qu.:-0.6716    1st Qu.:-0.71514   1st Qu.:-0.6676   1st Qu.:-0.6998
 Median :-0.1356   Median :-0.3468   Median :-0.2833    Median :-0.05916   Median :-0.2143   Median :-0.1530
 Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000    Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000
 3rd Qu.: 0.7216   3rd Qu.: 0.1237   3rd Qu.: 0.3542    3rd Qu.: 0.59028   3rd Qu.: 0.4571   3rd Qu.: 0.5160
 Max.   : 2.4359   Max.   :18.4020   Max.   : 9.2003    Max.   : 4.69898   Max.   :11.5159   Max.   : 5.8917
  ActionsInPAC     TotalMapExplored   WorkersMade      UniqueUnitsMade   ComplexUnitsMade  ComplexAbilitiesUsed
 Min.   :-2.1512   Min.   :-2.3004   Min.   :-1.8333   Min.   :-2.4427   Min.   :-0.5379   Min.   :-0.5343
 1st Qu.:-0.6700   1st Qu.:-0.6877   1st Qu.:-0.6710   1st Qu.:-0.8289   1st Qu.:-0.5379   1st Qu.:-0.5343
 Median :-0.1199   Median :-0.0157   Median :-0.2436   Median :-0.2910   Median :-0.5379   Median :-0.4573
 Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
 3rd Qu.: 0.5067   3rd Qu.: 0.6563   3rd Qu.: 0.4371   3rd Qu.: 0.7848   3rd Qu.: 0.2459   3rd Qu.: 0.1520
 Max.   : 8.8572   Max.   : 4.8224   Max.   : 7.9142   Max.   : 3.4743   Max.   : 7.5535   Max.   :11.0740
```

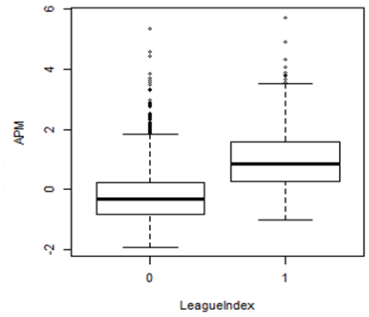
데이터를 표준화시켜주지 않으면 일반적으로 Scale이 큰 변수가 반응변수에 많은 영향을 끼친다는 결과를 가져올 수 있으므로 데이터를 표준화시켜주는 것이 중요하다.

어떠한 설명변수가 반응변수(LeagueIndex)에 영향을 많이 끼치는지 예측해보기 위해 boxplot들을 그려보았다.

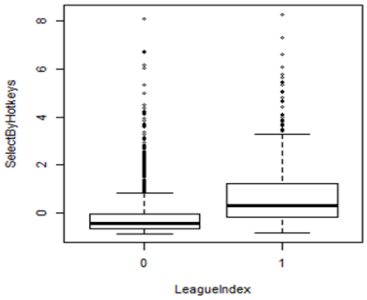


Boxplot을 확인해본 결과 반응변수의 0, 1 값에 대해 뚜렷한 차이를 식별할 수 있는 설명변수는 APM, Select by Hotkeys, Assign to Hotkeys, UniqueHotkeys, NumberOfPACs, ActionLatency 정도로 보인다. 그 중 반응변수 값이 0에서 1로 될 때 값이 커지는 설명변수로는 APM, Select by Hotkeys, Assign to Hotkeys, UniqueHotkeys, NumberOfPACs가 있고 값이 작아지는 설명변수로는 ActionLatency가 있을것으로 예측된다.

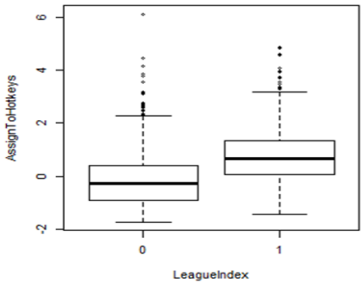
APM)



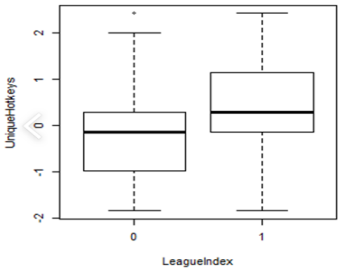
Select by Hotkeys)



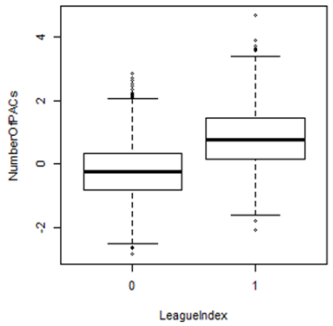
Assign to Hotkeys)



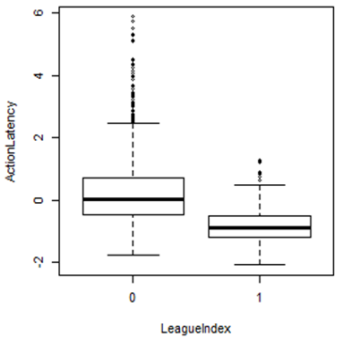
UniqueHotkeys)



NumberOfPACs)



ActionLatency)



이제 분석을 통해 model을 적합시켜 본다.

①로지스틱 회귀

(I) Logistic Regression

전체 데이터에 대해 LeagueIndex를 반응변수, 나머지 변수들을 설명변수로 하여 Logistic Regression을 하면 다음과 같은 결과가 나온다.

```
Call:
glm(formula = LeagueIndex ~ ., family = "binomial", data = data)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.31564  -0.53668  -0.25231  -0.05519   3.03383

Coefficients:
                    Estimate Std. Error z value Pr(>|z|)
(Intercept)        -2.469866   0.094984 -26.003  < 2e-16 ***
Age                -0.047186   0.068991  -0.684 0.494012
HoursPerWeek        0.370009   0.054078   6.842 7.80e-12 ***
TotalHours         -0.085318   0.081072  -1.052 0.292633
APM                -1.197106   0.335348  -3.570 0.000357 ***
SelectByHotkeys     0.950617   0.186622   5.094 3.51e-07 ***
AssignToHotkeys     0.249428   0.062390   3.998 6.39e-05 ***
UniqueHotkeys       0.296242   0.062804   4.717 2.39e-06 ***
MinimapAttacks      0.274311   0.051305   5.347 8.96e-08 ***
MinimapRightClicks  0.032143   0.056039   0.574 0.566249
NumberOfPACs        0.652629   0.201610   3.237 0.001208 **
GapBetweenPACs     -0.565957   0.123945  -4.566 4.97e-06 ***
ActionLatency      -1.373630   0.210302  -6.532 6.50e-11 ***
ActionsInPAC        0.392788   0.170794   2.300 0.021461 *
TotalMapExplored   -0.057843   0.072858  -0.794 0.427243
WorkersMade         0.180755   0.056911   3.176 0.001493 **
UniqueUnitsMade    -0.081955   0.070854  -1.157 0.247403
ComplexUnitsMade   -0.028096   0.065163  -0.431 0.666347
ComplexAbilitiesUsed -0.007368   0.062602  -0.118 0.906304
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 3308.3  on 3337  degrees of freedom
Residual deviance: 2161.5  on 3319  degrees of freedom
AIC: 2199.5
```

변수선택을 따로 하지 않았을 때 나오는 계수의 추정치고 Age, TotalHours, APM, GapBetweenPACS, ActionLatency, TotalMapExplored, UniqueUnitsMade, ComplexUnitsMade 의 변수들의 계수는 음수값을 가지므로 값이 증가 할 수록 티어가 낮아진다고 할 수 있다. 위 변수들을 제외한 나머지 변수들의 계수는 다 양수값으로 값이 증가하면 티어를 올릴 수 있다고 할 수 있다.

모델의 AIC는 2199.5이고 오분류율을 구하면 0.1482이 나온다.

오분류율은 다음과 같이 구할 수 있다.

(오분류율= $\frac{362+133}{2549+362+133+294}$ = 0.1482)

logis_pred	0	1
0	2549	362
1	133	294

다음으로는 반응변수에 유의미한 영향력을 미치지 않는 변수들을 제외시키기 위해 AIC기준으로 변수 선택을 하여 모델을 만들면 다음과 같다. (stepwise selection 사용)

```
Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.27870  -0.53932  -0.25316  -0.05538   3.02659

Coefficients:
                 Estimate Std. Error z value Pr(>|z|)
(Intercept)     -2.46951    0.09493 -26.015  < 2e-16 ***
HoursPerWeek     0.37517    0.05332   7.036 1.97e-12 ***
TotalHours      -0.08554    0.08130  -1.052 0.292750
APM             -1.14372    0.32885  -3.478 0.000505 ***
SelectByHotkeys  0.92556    0.18301   5.057 4.25e-07 ***
AssignToHotkeys  0.24719    0.06210   3.981 6.87e-05 ***
UniqueHotkeys    0.28965    0.06241   4.641 3.47e-06 ***
MinimapAttacks   0.27050    0.05009   5.400 6.67e-08 ***
NumberOfPACs     0.61711    0.20000   3.086 0.002032 **
GapBetweenPACs  -0.59042    0.12128  -4.868 1.13e-06 ***
ActionLatency   -1.35966    0.20851  -6.521 6.99e-11 ***
ActionsInPAC     0.37436    0.16977   2.205 0.027450 *
WorkersMade      0.18034    0.05622   3.208 0.001337 **
UniqueUnitsMade -0.11735    0.05953  -1.971 0.048685 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 3308.3  on 3337  degrees of freedom
Residual deviance: 2163.2  on 3324  degrees of freedom
AIC: 2191.2
```

stepwise selection을 사용했을 때 설명 변수들은 총 13개가 사용되었다. (Age, MinimapRightClicks, TotalMapExplored, ComplexunitsMade, ComplexAbilitiesUsed변수가 제외됨)

AIC는 2191.2로 앞선 모 델보다 작아졌고 이 모델을 식으로 쓰면 다음과 같다.

$$\begin{aligned}\log\left(\frac{\hat{\pi}(x)}{1-\hat{\pi}(x)}\right) = &-2.46951 + 0.37517\,HoursPerWeek - 0.08554\,TotalHours - 1.13372\,APM \\ &+ 0.92556\,SelectByHotkeys + 0.24719\,AssignToHotkeys + 0.28965\,UniqueHotkeys \\ &+ 0.2705\,MinimapAttacks + 0.61711\,NmberOfPacs - 0.59042\,GapBetweenPACs \\ &- 1.35966\,ActionLatency + 0.37436\,ActionsInPAC + 0.18034\,WorkersMade \\ &- 0.11735\,UniqueUnitsMade\end{aligned}$$

계수들에 대해 설명하자면 TotalHours, APM, GapBetweenPACs, ActionLatency, UniqueUnitsMade의 변수들의 계수는 음의 값을 가지므로 값이 증가하면 티어가 낮아진다고 할 수 있다. 위 변수를 제외한 나머지 설명변수들의 계수는 다 양수 값으로 값이 증가하면 티어가 올라간다. 이 외에도 더 세밀하게 오즈로 계수를 해석할 수 있다. HoursPerWeek는 다른 변수가 고정되어 있을 때 1의 단위가 증가하면 상위티어가 될 오즈는 $e^{0.37517} \simeq 1.4552$ 배 증가한다. 나머지 중 티어에 영향력이 높게 작용할 것 같은 변수들을 택해서 위와 동일하게 적용하면 APM은 단위가 1증가 할 때 상위티어가 될 오즈는 $e^{-1.13372} \simeq 0.3221$배 증가, ActionLatency는 단위가 1증가 할 때 상위티어가 될 오즈는 $e^{-1.35966} \simeq 0.25674$배 증가, 마지막으로 SelectByHotkeys는 단위가1 증가 할 때 최상위랭크가 될 오즈는 $e^{0.92556} \simeq 2.52328$ 배 증가한다.

오분류율을 구해보면 0.1470이 나온다.

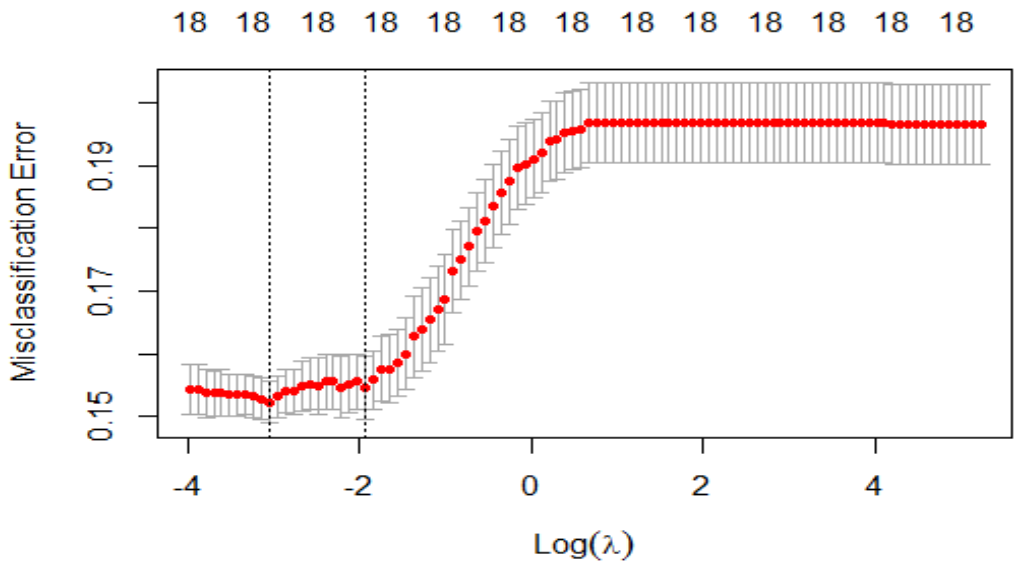
(오분류율 = $\frac{362+133}{2549+362+133+294}$ = 0.1470)

logis_pred	0	1
0	2551	360
1	131	296

결과를 보면, 앞서 변수선택을 하지 않은 모델에 비해 오분류율이 0.1482에서 0.1470로 줄어든 것을 알 수 있다. 오분류율만으로 어떤 모델이 좋다고 하기는 어렵지만 더 좋은 모델을 판단하는 요소가 될 수 있다. 따라서 변수선택 후 모델이 더 낮은 오분류율을 가지고 있으므로 변수선택 전 모델보다는 더 나은 모델이라고 할 수 있다.

(II) Logistic Ridge Regression

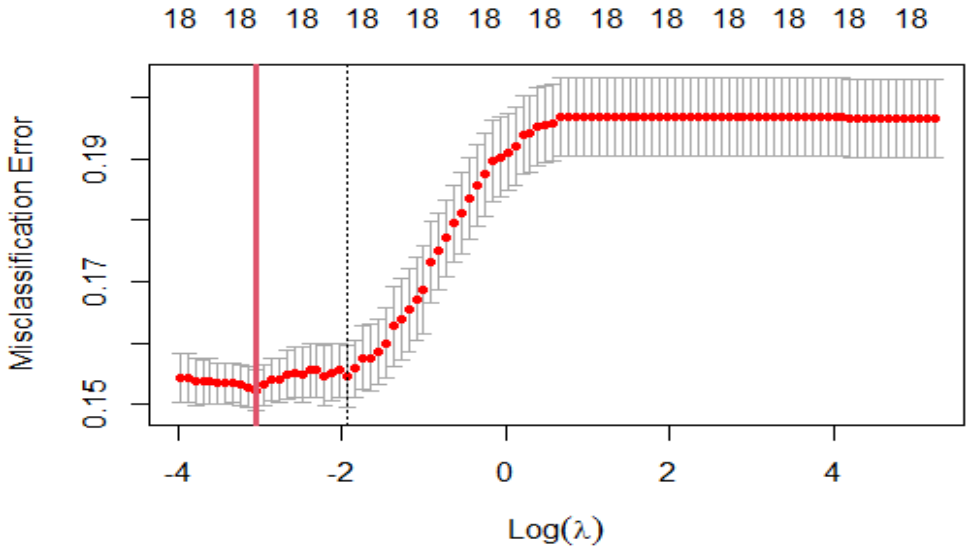
먼저 전체 데이터에 대해 Lamda 값을 구하고 로그(Lambda)로 하여 그래프를 나타내면 다음과 같이 나온다.



그 중 최소의 Lambda 값을 구하면 0.04739가 나온다. (Lambda.min 사용)

```
> best_lambda
[1] 0.04739202
```

Best Lambda값을 이용하여 로그(Best Lambda)를 구하면 -3.049301이 나온다. 위의 그래프에서 -3.049301에 해당하는 곳에 수직선을 그으면 왼쪽 점선에 해당한다.

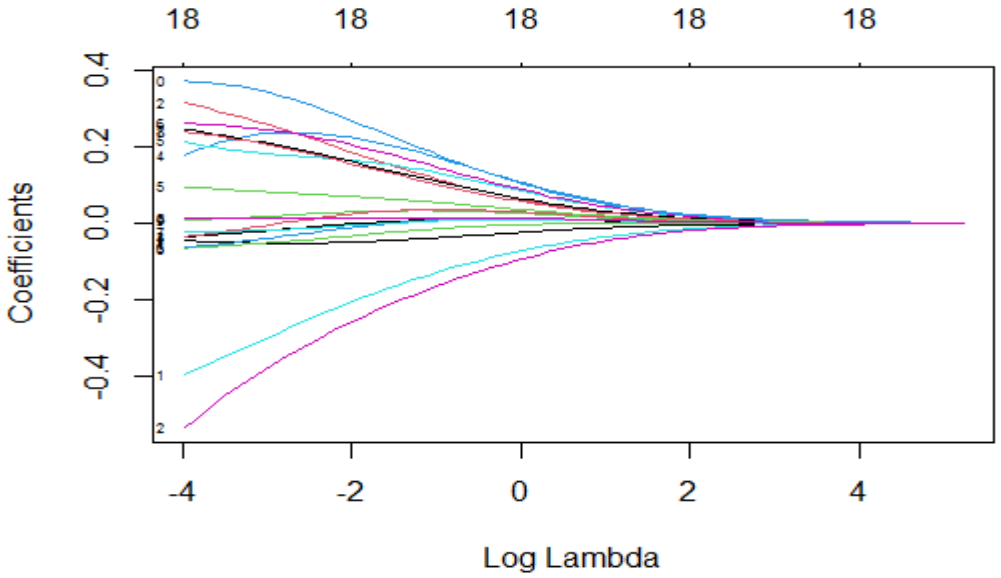


그 후 Best Lambda값을 사용하여 Logistic Ridge Regression을 하여 오분류율을 구하면 다음과 같이 나온다.

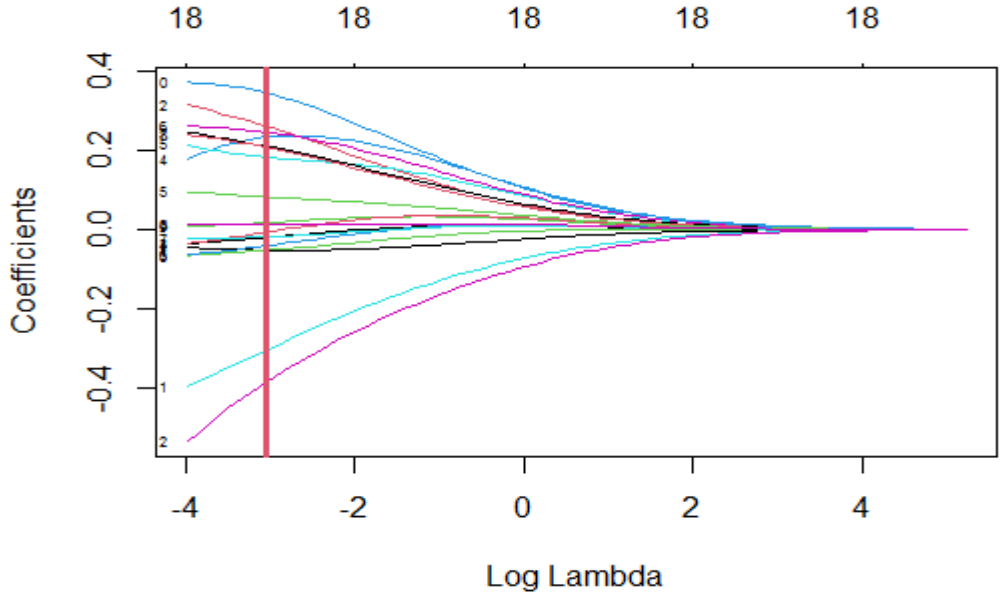
오분류율 = $\frac{104+404}{2589+404+104+252}$ = 0.1521 이다.

ydata	0	1
0	2578	404
1	104	252

다음으로 계수를 추정하는 방법은 다음과 같다.



각 설명변수와 Log(Lambda)의 관계를 그래프로 나타내면 위와 같은 선으로 표현되고 여기에 Log(best_Lambda)값에 해당하는 선을 수직으로 그으면 다음과 같이 나온다.



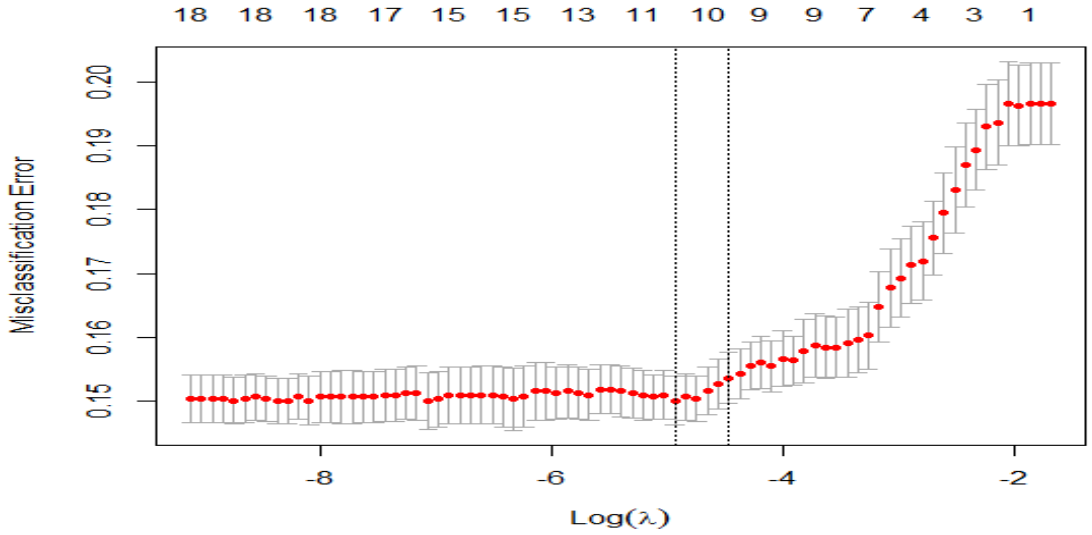
빨간색의 수직선과 설명변수의 선들이 만나는 지점이 계수의 추정치가 되고 값은 다음과 같이 나온다.

```
19 x 1 sparse Matrix of class "dgCMatrix"
                                1
(Intercept)          -2.001881976
Age                  -0.052043700
HoursPerWeek          0.262304472
TotalHours           -0.052821858
APM                   0.233504932
SelectByHotkeys       0.182863261
AssignToHotkeys       0.246254281
UniqueHotkeys         0.211489866
MinimapAttacks        0.207961501
MinimapRightClicks    0.017191437
NumberOfPACs          0.345622430
GapBetweenPACs       -0.306460416
ActionLatency        -0.387046049
ActionsInPAC         -0.020819833
TotalMapExplored     -0.007573988
WorkersMade           0.082195843
UniqueUnitsMade      -0.041833302
ComplexUnitsMade     -0.018811249
ComplexAbilitiesUsed  0.011584221
```

Age, TotalHours, ActionLatency, GapBetweenPAC, ActionsInPAC, TotalMapExplored, UniqueUnitsMade, ComplexUnitsMade 값들은 계수가 음의 값을 가지고 있다. 따라서 이 설명변수들에 해당하는 값이 올라간다면 티어가 낮아진다고 할 수 있다. 반면 HoursPerWeek, APM, SelectByHotkeys 등 위에서 언급되지 않은 설명변수들은 계수가 양의 값을 가지고 있으므로 이 설명변수들에 해당하는 값이 올라가면 티어가 높아진다고 할 수 있다. 절대값을 이용하여 모든 설명변수 중 티어에 가장 비교적 많은 영향을 끼치는 3가지 변수를 찾는다면 ActionsInPac, NumberofPACs, GapBetweenPACs가 될 수 있다.

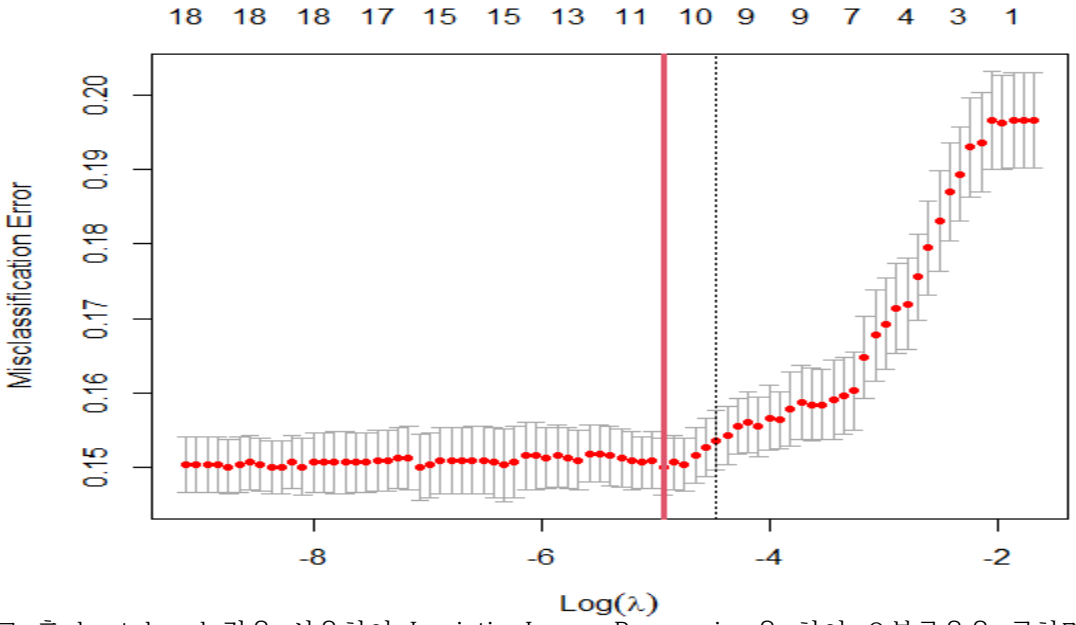
(III) Logistic Lasso Regression

Ridge 방법과 마찬가지로 먼저 전체 데이터에 대해 Lamda 값을 구하고 로그(Lambda)로 하여 그래프를 나타내면 다음과 같이 나온다.



그 중 최소의 Lambda 값을 구하면 0.007203165가 나온다. (Lambda.min 사용)

이 값을 이용하여 로그(best_lambda)를 구하면 -4.933235가 나온다. 위의 그래프에서 -4.933235에 해당하는 곳에 수직선을 그리면 왼쪽 점선에 해당한다.

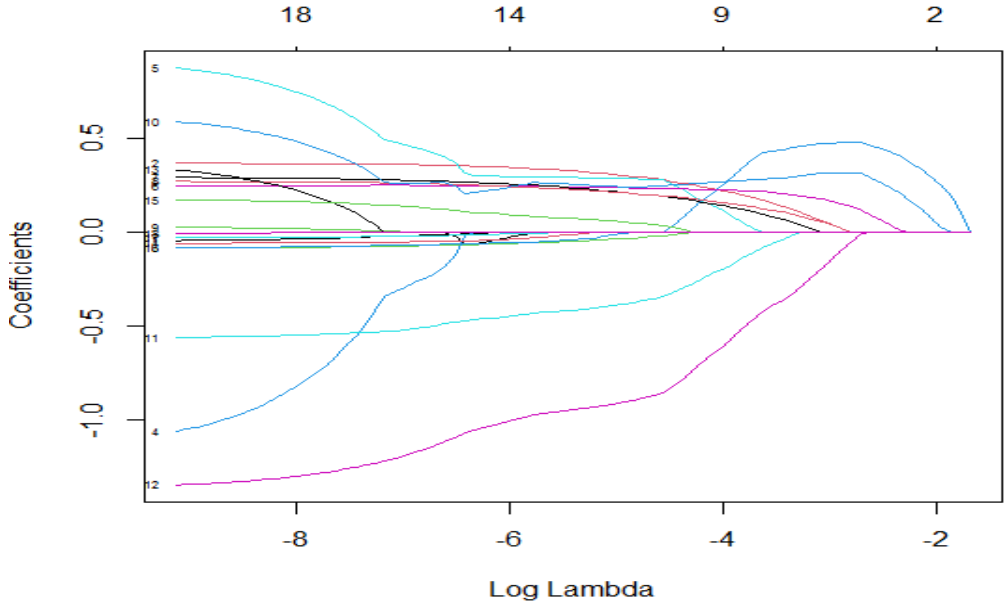


그 후 best_lamda값을 사용하여 Logistic Lasso Regression을 하여 오분류율을 구하면 다음과 같이 나온다.

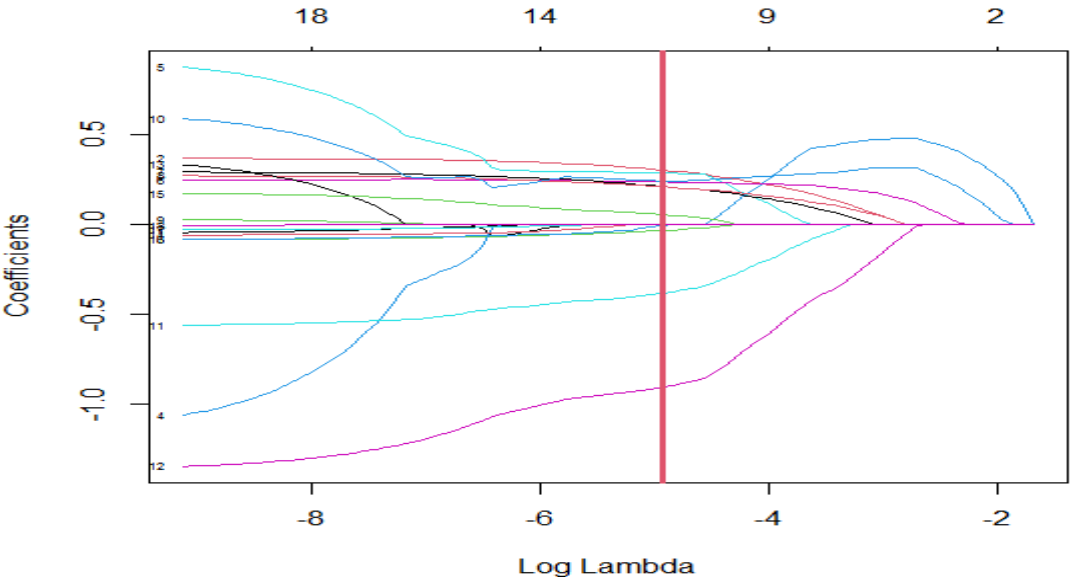
오분류율 = $\frac{111+384}{2571+384+111+272}$ = 0.1482 이다.

ydata		
	0	1
0	2571	384
1	111	272

다음으로 계수를 추정하는 방법은 다음과 같다.



각 설명변수와 Log(Lambda)의 관계를 그래프로 나타내면 위와 같은 선으로 표현되고 여기에 Log(best_Lambda)값에 해당하는 선을 수직으로 그리면 다음과 같이 나온다.



빨간색의 수직선과 설명변수의 선들이 만나는 지점이 계수의 추정치가 되고 값은 다음과 같이 나온다. 여기서 Ridge 방법을 통한 계수추정과 Lasso 방법과의 차이가 발생한다. Ridge에서는 모든 설명변수를 사용하지만 Lasso에서는 모델링을 할 때 일부의 설명변수가 제외 된다. 이를 시각적으로 확인하기 위해 위의 그래프에서 x축 윗부분을 보면 앞선 Ridge 그래프에서는 모두 18이었지만 Lasso에서는 Log(Lambda)의 값이 커질수록 숫자가 줄어드는 것을 알 수 있다.

```
19 x 1 sparse Matrix of class "dgCMatrix"                    1
(Intercept)          -2.186068930
Age                   .
HoursPerWeek          0.305246826
TotalHours           -0.032445191
APM                   .
SelectByHotkeys       0.289185135
AssignToHotkeys       0.238797113
UniqueHotkeys         0.213880791
MinimapAttacks        0.211107785
MinimapRightClicks    .
NumberOfPACs          0.242833181
GapBetweenPACs       -0.382659721
ActionLatency        -0.906407423
ActionsInPAC          .
TotalMapExplored      .
WorkersMade           0.056693807
UniqueUnitsMade      -0.005100938
ComplexUnitsMade      .
ComplexAbilitiesUsed  .
```

Age, APM, MinimapRightClicks, ActionsInPAC, TotalMapExplored, ComplexUntisMade, ComplexAbilitiesUsed의 설명변수들은 모델링과정에서 제외되어 총 11개의 변수들이 선택되었음을 알 수 있다. 여기서 TotalHours, ActionLatency, GapBetweenPAC, UniqueUnitsMade 값들은 계수가 음의 값을 가지고 있다. 따라서 이 설명변수들에 해당하는 값이 올라간다면 티어가 낮아진다고 할 수 있다. 반면 위에서 언급되지 않은 나머지 설명변수들은 계수가 양의 값을 가지고 있으므로 이 설명변수들에 해당하는 값이 올라가면 티어가 높아진다고 할 수 있다.

절댓값을 이용하여 모든 설명변수 중 티어에 가장 비교적 많은 영향을 끼치는 3가지 변수를 찾는다면 ActionsInPac, HoursPerWeek, GapBetweenPACs가 된다.

(IV) 주성분 로지스틱 회귀분석

아래와 같이 18개의 주성분을 만들었다.

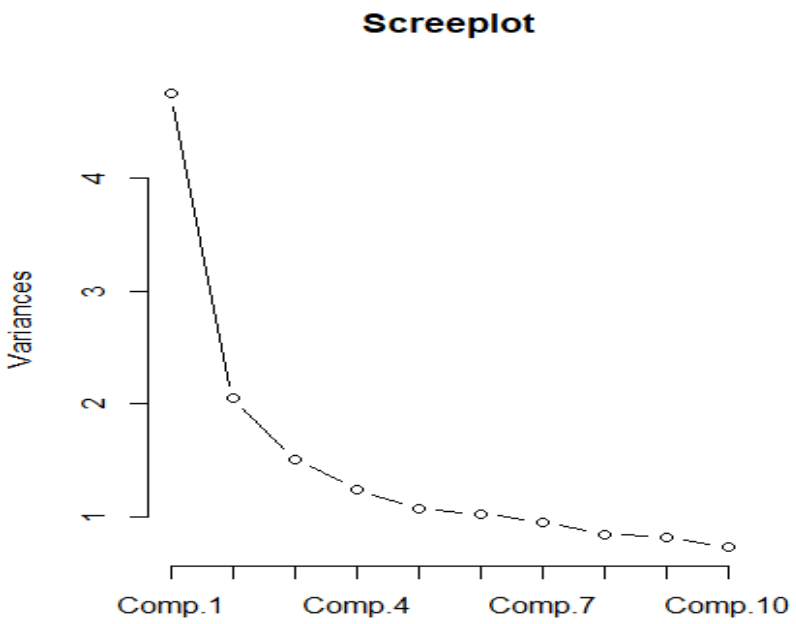
	Comp.1	Comp.2	Comp.3	Comp.4	Comp.5	Comp.6	Comp.7	Comp.8
Age	0.10492606	0.077705620	0.058218748	0.54150467	0.08404449	0.24409419	0.204437327	0.39503615
HoursPerWeek	0.01243259	0.028559921	-0.028347590	0.20033083	0.51068279	-0.43134133	0.654900927	-0.15053047
TotalHours	-0.01870790	0.004464755	0.039688465	-0.07944769	-0.62197971	0.34797466	0.632709063	-0.20915174
APM	-0.39803716	-0.250362689	-0.006819237	-0.05856370	0.09246461	0.08416308	0.014738062	0.07847962
SelectByHotkeys	-0.27606413	-0.241426662	-0.127098917	-0.07041409	0.26947798	0.36318622	-0.006418566	0.08676795
AssignToHotkeys	-0.30150487	-0.068114328	-0.119137331	0.04555622	0.17525768	0.29394375	0.004564000	-0.05917130
UniqueHotkeys	-0.22602186	0.078755810	-0.177895244	0.26014546	0.11103850	0.35037930	0.082804180	0.11632597
MinimapAttacks	-0.14381918	-0.039094699	0.131821185	0.49230783	-0.03712105	0.03477429	-0.310805370	-0.50267632
MinimapRightClicks	-0.17200874	-0.081470044	0.371707290	0.32146795	-0.19764635	-0.19442270	-0.045470321	-0.01488918
NumberOfPACs	-0.36249872	0.107934628	-0.341394962	-0.07227644	-0.10546232	-0.19834864	0.011215112	0.02676974
GapBetweenPACs	0.30328494	0.216015238	-0.013376547	-0.01841933	0.11870866	0.21562092	-0.084811256	0.18387703
ActionLatency	0.38679039	0.056306410	0.157252465	0.10601004	0.10073576	0.23885546	-0.041288656	0.06693629
ActionsInPAC	-0.09769818	-0.370457197	0.552095000	0.03157110	0.03549659	0.02561913	0.043114281	0.02102140
TotalMapExplored	-0.22468749	0.410148876	-0.072333458	0.18411925	-0.12839503	-0.12165576	-0.079433542	0.08760168
WorkersMade	-0.20274868	-0.075084777	0.246739089	-0.10187885	-0.14231345	-0.21438353	0.035830957	0.63062729
UniqueUnitsMade	-0.18101394	0.460458849	0.051380300	0.17330675	-0.13395201	-0.04116185	-0.027737211	0.08652612
ComplexUnitsMade	-0.17331422	0.381862368	0.368829334	-0.26337615	0.18055378	0.12107346	0.011807198	-0.04187973
ComplexAbilitiesUsed	-0.15698655	0.350325082	0.354391911	-0.27386374	0.23816580	0.15954406	0.061656035	-0.19965328

	Comp.9	Comp.10	Comp.11	Comp.12	Comp.13	Comp.14	Comp.15
Age	0.525859790	5.077861e-02	0.332954363	0.142809799	0.018384541	0.047621565	0.037753742
HoursPerWeek	-0.192607068	-1.417292e-01	-0.031054454	-0.015591985	-0.023955326	-0.021622977	0.014758973
TotalHours	-0.097858669	-1.595870e-01	0.045046346	0.036914087	0.009043604	0.026670674	0.022361117
APM	-0.126288316	4.947332e-05	0.262196135	-0.042510389	0.121330124	0.007291750	0.007703406
SelectByHotkeys	-0.310345433	-7.632511e-02	0.422232694	0.052918572	0.143868948	-0.193676386	-0.035493461
AssignToHotkeys	0.037842286	-3.454323e-02	-0.266634814	0.214730157	-0.792393129	0.136164036	0.012471779
UniqueHotkeys	-0.077976110	1.973583e-01	-0.650393714	-0.221667368	0.412323288	-0.013127436	0.008531004
MinimapAttacks	0.053494655	-5.839861e-01	-0.009391575	0.005861872	0.134870461	-0.038033308	-0.010086404
MinimapRightClicks	-0.316163872	4.317824e-01	-0.066511149	0.572980455	0.041909111	-0.123351827	0.043245640
NumberOfPACs	0.128536490	-2.620643e-02	0.056007622	0.215361405	0.121324790	0.009398649	0.018179398
GapBetweenPACs	-0.481936593	-2.052930e-01	0.035115719	0.229860411	0.050185840	0.175432874	0.001323701
ActionLatency	-0.235037565	-7.714463e-02	-0.057763918	0.009613032	-0.091387829	-0.080751428	-0.040887002
ActionsInPAC	-0.037546109	1.406822e-01	0.026537075	-0.463890185	-0.088829738	0.260176260	0.005156525
TotalMapExplored	-0.251344266	2.720447e-02	0.201563758	-0.184646412	-0.034777315	0.666509536	0.099123243
WorkersMade	-0.004152502	-5.419481e-01	-0.261364894	0.080626204	0.001868051	-0.043536734	-0.141498392
UniqueUnitsMade	-0.174488824	8.367839e-02	0.151605838	-0.389763985	-0.283346108	-0.563355892	-0.246950751
ComplexUnitsMade	0.118520959	-7.465647e-02	-0.012084761	0.039679931	0.056482608	-0.164990594	0.719356634
ComplexAbilitiesUsed	0.202680639	5.313380e-02	0.004664126	0.203020855	0.166055043	0.166777527	-0.619678146

	Comp.16	Comp.17	Comp.18
Age	0.0849746513	0.013691969	0.006515428
HoursPerWeek	-0.0047943880	-0.003362079	0.001076755
TotalHours	-0.0008709253	0.001028879	0.001233593
APM	0.1397499418	-0.253948386	0.753686482
SelectByHotkeys	-0.2838614716	0.194257029	-0.413877557
AssignToHotkeys	0.0292476178	0.014071941	0.008998759
UniqueHotkeys	0.0121699738	0.017718491	-0.003024368
MinimapAttacks	0.0484463273	0.004731565	-0.006432521
MinimapRightClicks	-0.0524066692	0.044756315	-0.019910326
NumberOfPACs	0.2923229540	-0.600793392	-0.394453772
GapBetweenPACs	0.6241225509	0.089135116	-0.001228362
ActionLatency	-0.4047219380	-0.701687052	0.023229332
ActionsInPAC	0.3160691826	-0.164850751	-0.321573970
TotalMapExplored	-0.3182079924	0.041564187	0.001495235
WorkersMade	-0.1467364085	0.067459925	-0.016351378
UniqueUnitsMade	0.1531535372	0.011016977	0.008787921
ComplexUnitsMade	-0.0151086338	-0.011202170	-0.002866625

주성분들의 분산이 데이터를 설명하는 비율과 Screeplot을 아래와 같이 나타내보았다.

Comp.1	Comp.2	Comp.3	Comp.4	Comp.5	Comp.6	Comp.7
13.4	8.8	7.6	6.9	6.4	6.3	6.0
Comp.8	Comp.9	Comp.10	Comp.11	Comp.12	Comp.13	Comp.14
5.7	5.6	5.3	5.2	4.8	4.4	4.0
Comp.15	Comp.16	Comp.17	Comp.18			
3.7	3.1	2.1	0.8			



분산이 설명하는 비율을 보면 3번째 주성분까지 약 72% 정도를 설명하고, Screeplot을 봐도 적절하므로 10개의 주성분을 선택한다.

주성분이 각각 어떤 변수를 잘 설명하고 있는지 파악하기 위해서 10번째 주성분까지를 반올림하여 나타냈다.

	Comp.1	Comp.2	Comp.3	Comp.4	Comp.5
Age	0.105	0.078	0.058	0.542	0.084
HoursPerWeek	0.012	0.029	-0.028	0.200	0.511
TotalHours	-0.019	0.004	0.040	-0.079	-0.622
APM	-0.398	-0.250	-0.007	-0.059	0.092
SelectByHotkeys	-0.276	-0.241	-0.127	-0.070	0.269
AssignToHotkeys	-0.302	-0.068	-0.119	0.046	0.175
UniqueHotkeys	-0.226	0.079	-0.178	0.260	0.111
MinimapAttacks	-0.144	-0.039	0.132	0.492	-0.037
MinimapRightClicks	-0.172	-0.081	0.372	0.321	-0.198
NumberOfPACs	-0.362	0.108	-0.341	-0.072	-0.105
GapBetweenPACs	0.303	0.216	-0.013	-0.018	0.119
ActionLatency	0.387	0.056	0.157	0.106	0.101
ActionsInPAC	-0.098	-0.370	0.552	0.032	0.035
TotalMapExplored	-0.225	0.410	-0.072	0.184	-0.128
WorkersMade	-0.203	-0.075	0.247	-0.102	-0.142
UniqueUnitsMade	-0.181	0.460	0.051	0.173	-0.134
ComplexUnitsMade	-0.173	0.382	0.369	-0.263	0.181
ComplexAbilitiesUsed	-0.157	0.350	0.354	-0.274	0.238

	Comp.6	Comp.7	Comp.8	Comp.9	Comp.10
Age	0.244	0.204	0.395	0.526	0.051
HoursPerWeek	-0.431	0.655	-0.151	-0.193	-0.142
TotalHours	0.348	0.633	-0.209	-0.098	-0.160
APM	0.084	0.015	0.078	-0.126	0.000
SelectByHotkeys	0.363	-0.006	0.087	-0.310	-0.076
AssignToHotkeys	0.294	0.005	-0.059	0.038	-0.035
UniqueHotkeys	0.350	0.083	0.116	-0.078	0.197
MinimapAttacks	0.035	-0.311	-0.503	0.053	-0.584
MinimapRightClicks	-0.194	-0.045	-0.015	-0.316	0.432
NumberOfPACs	-0.198	0.011	0.027	0.129	-0.026
GapBetweenPACs	0.216	-0.085	0.184	-0.482	-0.205
ActionLatency	0.239	-0.041	0.067	-0.235	-0.077
ActionsInPAC	0.026	0.043	0.021	-0.038	0.141
TotalMapExplored	-0.122	-0.079	0.088	-0.251	0.027
WorkersMade	-0.214	0.036	0.631	-0.004	-0.542
UniqueUnitsMade	-0.041	-0.028	0.087	-0.174	0.084
ComplexUnitsMade	0.121	0.012	-0.042	0.119	-0.075
ComplexAbilitiesUsed	0.160	0.062	-0.200	0.203	0.053

첫 번째 주성분은 ActionsLatency와 음의 방향의 APM을 잘 설명하고 있다. 두 번째 주성분은 TotalMapEXplored와 UniqueUnitsmade, ComplexUnitsmade, Complex AbilitiesUsed를 잘 설명하고 있다. 이러한 방식으로 10개의 주성분을 설명할 수 있다.

이제 LeagueIndex를 반응변수로 하고 10개의 주성분을 설명변수로 하여 로지스틱 회귀분석을 실시한다.

```
Call:
glm(formula = LeagueIndex ~ Comp.1 + Comp.2 + Comp.3 + Comp.4 +
    Comp.5 + Comp.6 + Comp.7 + Comp.8 + Comp.9 + Comp.10, family = "binomial",
    data = data2)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.32810  -0.54236  -0.28421  -0.09476   2.92310

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -2.23193    0.07705 -28.968  < 2e-16 ***
Comp.1      -0.85382    0.03719 -22.958  < 2e-16 ***
Comp.2      -0.27334    0.03698  -7.391 1.46e-13 ***
Comp.3      -0.31032    0.04278  -7.253 4.07e-13 ***
Comp.4       0.02681    0.04799   0.559  0.57645
Comp.5       0.04868    0.04980   0.977  0.32836
Comp.6      -0.12559    0.05302  -2.369  0.01785 *
Comp.7      -0.10021    0.05546  -1.807  0.07077 .
Comp.8      -0.17760    0.05606  -3.168  0.00153 **
Comp.9       0.38681    0.06802   5.686 1.30e-08 ***
Comp.10     -0.03611    0.05875  -0.615  0.53877
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 3308.3  on 3337  degrees of freedom
Residual deviance: 2246.3  on 3327  degrees of freedom
AIC: 2268.3

Number of Fisher Scoring iterations: 6
```

위의 결과에서 4,5,10번째 주성분의 계수가 유의하지 않으므로 설명변수에서 제외하고 다시 로지스틱 회귀분석을 실시하였다.

```
Call:
glm(formula = LeagueIndex ~ Comp.1 + Comp.2 + Comp.3 + Comp.6 +
    Comp.7 + Comp.8 + Comp.9, family = "binomial", data = data2)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.25143  -0.54157  -0.28376  -0.09188   2.92025

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -2.24264    0.07683 -29.190  < 2e-16 ***
Comp.1      -0.86034    0.03692 -23.305  < 2e-16 ***
Comp.2      -0.27753    0.03680  -7.542 4.64e-14 ***
Comp.3      -0.30899    0.04269  -7.237 4.57e-13 ***
Comp.6      -0.12807    0.05256  -2.437 0.014820 *
Comp.7      -0.10698    0.05509  -1.942 0.052173 .
Comp.8      -0.18588    0.05498  -3.381 0.000722 ***
Comp.9       0.38980    0.06764   5.763 8.28e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 3308.3  on 3337  degrees of freedom
Residual deviance: 2247.8  on 3330  degrees of freedom
AIC: 2263.8

Number of Fisher Scoring iterations: 6
```

회귀식: $\text{logit}(\pi(x))$ = -2.2426437 -0.8603351 * Comp.1 -0.2775257 * Comp.2 -0.3089852 * Comp.3 -0.1280696 * Comp.6 -0.1069761 * Comp.7 -0.1858756 * Comp.8 0.3897958 * Comp.9

추정된 회귀계수들은 원래의 로지스틱 회귀분석과 마찬가지로 오즈비의 개념으로 해석할 수 있지만, 여기서는 영향을 끼치는 변수에 대해서만 간단하게 해석해보자. 회귀계수가 가장 큰 2개의 주성분 1,9번째 주성분에 대해서 보면 첫 번째 주성분은 ActionLatency, 음의 방향의 APM에 대해 잘 설명해주는데 계수가 음수이므로 APM이 높을수록, ActionLatency 값이 낮을수록 LeagueIndex가 1이 될 확률이 높은 것을 알 수 있다. 9번째 주성분은 Age와 음의 방향의 GapbetweenPACs에 대해 잘 설명하고 있는 주성분이고 계수가 양수이므로 유저들 중에서는 Age가 높을수록, GapbetweenPACs 수치가 낮을수록 eagueIndex가 1이 될 확률이 높다고 볼 수 있다. 나머지 주성분들도 이런 방식으로 설명이 가능하다. 여기서 가장 반응변수에 영향을 많이 주는 변수들은 APM,ActionLatency,Age,GapbetweenPACs 정도로 볼 수 있다.

pred	0	1
0	2605	453
1	77	203

예측된 모형으로 분류를 하고 오분류율을 구해보면 0.1620이다.

(오분류율= $\frac{453+77}{2605+453+77+203}$ =0.1620)

(V) 로지스틱 모델 비교(with K-fold)

위에서 4가지 변수선택법에 대해서 배워보았다. 따라서, 이제 데이터 분류를 위해서 어떤 모델을 이용하는 것이 좋을지 알아보기 위해서 각각의 방법의K-fold CV(K=10)를 구하여 비교해보았다.

분석방법 / 오분류율	Logistic	Ridge	Lasso	주성분 분석
Training data	0.1469	0.1509	0.1531	0.1546
Test data	0.1486	0.1550	0.1577	0.1617

오분류율을 확인해본 결과 Logistic regression 방법을 이용한 모델의 오분류율이 현저히 낮음을 볼 수 있다. 그러므로 4가지 방법 중에서 위에서 분석한 Logistic regression을 이용하여 데이터를 분석하는 것이 제일 좋을 것이라고 생각한다. 그러므로 (I)에서 분석한 결과를 가져오면 APM, ActionLatency, SelectByHotkeys 정도의 변수들이 반응변수에 많은 영향을 끼칠 것으로 생각된다.

오분류율을 줄여줄 다른 좋은 분류 방법이 있는지 알아보기 위해 다양한 분류방법 이용해 분류를 하고 K-fold CV(K=10)를 이용하여 오분류율을 비교해 보았다.

②방법론

KNN, LDA, QDA는 k-fold(k=10)를 적용하여 training, test 오분류율을 구할 수 있으나 뒤에 소개할 SVM과 Tree-based method는 장비의 한계로 k-fold의 적용이 어려워 단순 sampling으로 데이터를 구분하여 training, test 오분류율을 구하였다.

(I) KNN, LDA, QDA

a.KNN

knn(k=2~10) 의 K-fold CV(K=10)를 구해보았다.

k / 오분류율	2	3	4	5	6	7	8	9	10
training data	0.0623	0.0551	0.0698	0.0689	0.0757	0.0747	0.0817	0.0800	0.0839
test data	0.1338	0.1085	0.1145	0.1070	0.1054	0.1034	0.1303	0.1065	0.1099

b. LDA,QDA

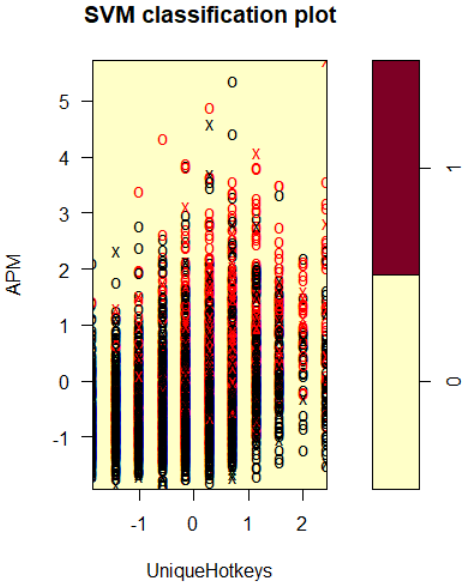
LDA,QDA는 반응변수의 범주가 2개 이상일 때, 번주의 구분이 명확할 때 로지스틱 회귀분석보다 주로 나은 성과를 보인다. 그러나 설명변수가 다변량 정규분포나 다변량 가우시안분포를 따른다고 가정할 때 LDA,QDA를 사용할 수 있다. 우리 설명변수에 정규성검정을 시행했으나 정규성이 있다고 하기 어려웠다. 그러나 우리는 오분류의 최소화가 목적이므로 데이터에 LDA와 QDA도 실행해보기로하였다

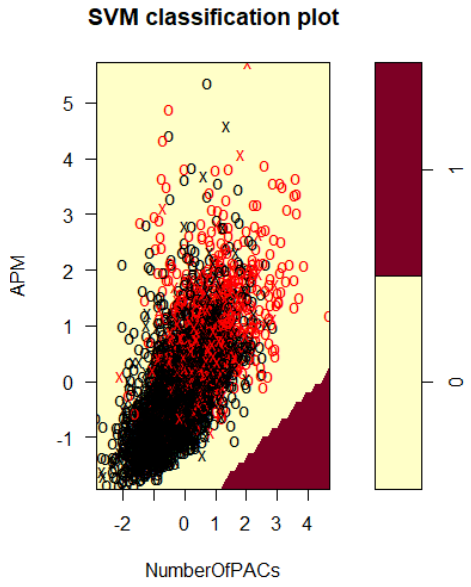
분류방법 / 오분류율	LDA	QDA
Training data	0.1578	0.1820
Test data	0.1610	0.1815

세 가지 분류 방법 중에서는 KNN 방법이 가장 오분류율이 작게 나왔다.

(II)SVM

a. 지지벡터 분류기





모형 적합결과를 보기 전에 먼저 선형분류가 제대로 되었는지 변수 2개 간의 plot을 몇 개 그려서 알아볼 수 있다.

```
Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost
    5

- best performance: 0.1599668

- Detailed performance results:
  cost     error dispersion
1  0.1 0.1599668 0.04548701
2  1.0 0.1605656 0.03955971
3  5.0 0.1599668 0.03815134
```

```
best.tune(method = svm, train.x = LeagueIndex ~
    ., data = q_train, ranges = list(cost = c(0.1,
    1, 5)), kernel = "linear")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  5

Number of Support Vectors:  599

 ( 303 296 )


Number of Classes:  2

Levels:
 0 1
```

결과를 살펴보면 지지벡터는 총 599개이며, 반응변수가 0인 쪽의 지지벡터는 303개, 1인 쪽의 지지벡터는 296개이다. 모델 선택에서는 cost가 5일 때, 오분류율이 가장 낮은 모델이 된다. 따라서, cost가 5인 모델을 training data에 적합시키고 training data, test data를 예측하고 분류하여 오분류율을 구해본다.

```
best.train    0    1
          0 1307  222
          1   36  104
```

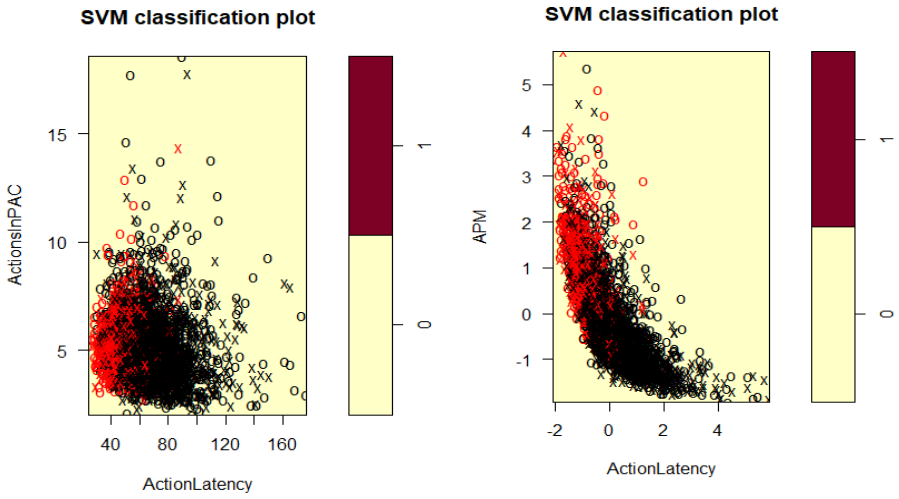
training 데이터에 대한 오분류율 : (36+222)/1669 = 0.1545

```
best.test    0    1
         0 1287  208
         1   52  122
```

test 데이터에 대한 오분류율: (208+52)/1669 = 0.1557

b. 지지벡터 기계

(1) radial kernel 사용



비선형인 분류결과를 나타내야하는데, 아래 table표를 보면 알다시피 1로 분류된 관측치가 거의 없으므로 그림에서도 대부분이 0으로 나타난다. 지지벡터들의 분포는 비선형처럼 보이긴 한다.

```
Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost gamma
   10   0.5

- best performance: 0.1947623

- Detailed performance results:
    cost gamma     error dispersion
1    0.1   0.5 0.1953611 0.02968717
2    1.0   0.5 0.1953611 0.02968717
3   10.0   0.5 0.1947623 0.03348091
4    0.1   1.0 0.1953611 0.02968717
5    1.0   1.0 0.1953611 0.02968717
6   10.0   1.0 0.1953611 0.02968717
7    0.1   2.0 0.1953611 0.02968717
8    1.0   2.0 0.1953611 0.02968717
9   10.0   2.0 0.1953611 0.02968717
10   0.1   3.0 0.1953611 0.02968717
11   1.0   3.0 0.1953611 0.02968717
12  10.0   3.0 0.1953611 0.02968717
13   0.1   4.0 0.1953611 0.02968717
14   1.0   4.0 0.1953611 0.02968717
15  10.0   4.0 0.1953611 0.02968717
```

```
Call:
best.tune(method = svm, train.x = LeagueIndex ~
    ., data = q_train, ranges = list(cost = c(0.1,
    1, 10), gamma = c(0.5, 1, 2, 3, 4)), kernel = "radial")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  10

Number of Support Vectors:  1593

 ( 1267 326 )


Number of Classes:  2

Levels:
 0 1
```

결과를 살펴보면 지지벡터는 총 1593개이며, 반응변수가 0인 쪽의 지지벡터는 1267개, 1인 쪽의 지지벡터는 326개이다. 모델 선택에서는 cost가 10이고 gamma는 0.5일 때, 오분류율이 가장 낮은 모델이 된다. 따라서, cost=1, gamma=0.5인 모델을 training data에 적합시키고 training data, test data를 예측하고 분류하여 오분류율을 구해본다.

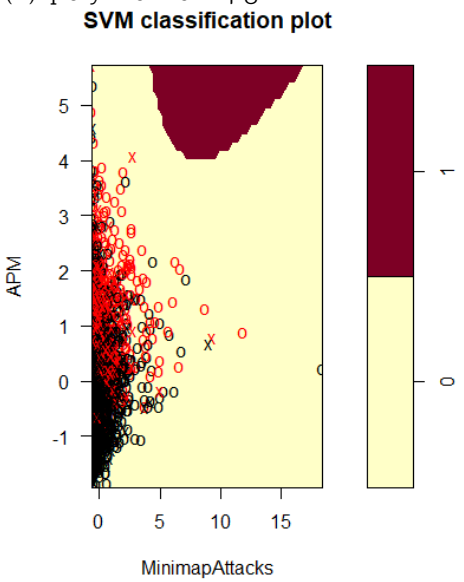
```
radial.train    0    1
           0 1343    0
           1    0  326
```

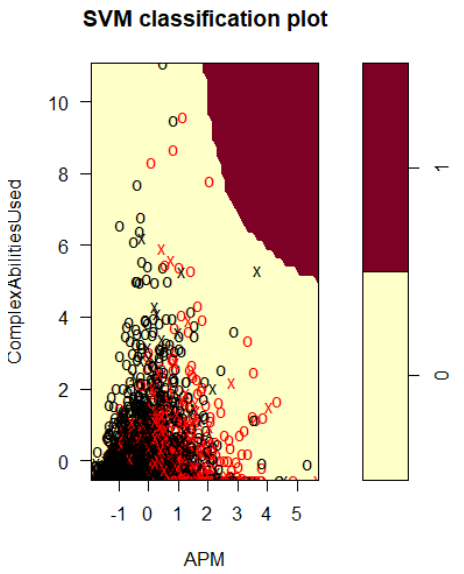
training 데이터에 대한 오분류율 : 0/1669 = 0

```
radial.test    0    1
           0 1337  322
           1    2    8
```

test 데이터에 대한 오분류율: (2+322)/1669 = 0.1941

(2) poly kernel 사용





poly kernel을 사용한 지지벡터기계로 분류되는 형태를 알아볼 수 있다.

```
Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost degree
    1      3

- best performance: 0.1659584

- Detailed performance results:
   cost degree     error dispersion
1   0.1      2 0.1857370 0.02146687
2   1.0      2 0.1785369 0.02382131
3  10.0      2 0.1755321 0.02605318
4   0.1      3 0.1743489 0.01958130
5   1.0      3 0.1659584 0.02389440
6  10.0      3 0.1941130 0.02966885
7   0.1      4 0.1785405 0.02229041
8   1.0      4 0.1803225 0.02980340
9  10.0      4 0.1965262 0.02862829
10  0.1      5 0.1737465 0.02196356
11  1.0      5 0.1761453 0.02425319
12 10.0      5 0.1827429 0.02724742
```

```
Call:
best.tune(method = svm, train.x = LeagueIndex ~
    ., data = q_train, ranges = list(cost = c(0.1,
    1, 10), degree = c(2, 3, 4, 5)), kernel = "poly")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  polynomial
       cost:  1
     degree:  3
     coef.0:  0

Number of Support Vectors:  642

 ( 371 271 )


Number of Classes:  2

Levels:
 0 1
```

결과를 살펴보면 지지벡터는 총 642개이며, 반응변수가 0인 쪽의 지지벡터는 371개, 1인 쪽의 지지벡터는 271개이다. 모델 선택에서는 cost가 1이고 degree=3일 때, 오분류율이 가장 낮은 모델이 된다. 따라서, cost=1, degree=3인 모델을 training data에 적합시키고 training data, test data를 예측하고 분류하여 오분류율을 구해본다.

```
poly.train    0    1
          0 1329  172
          1   14  154
```

training 데이터에 대한 오분류율 : (14+172)/1669 = 0.1114

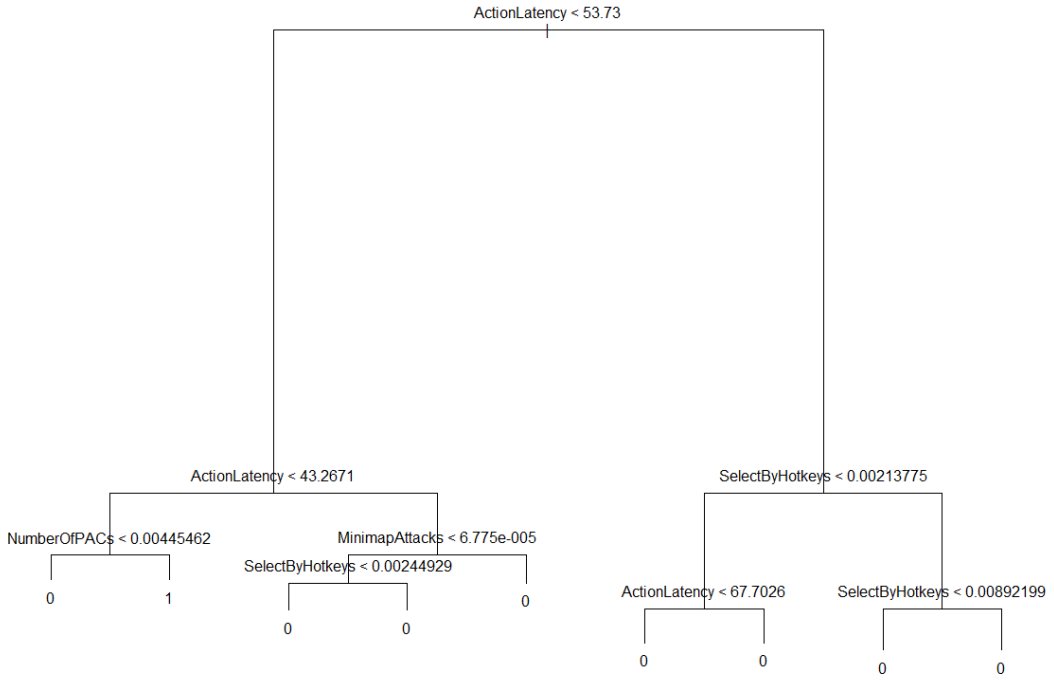
```
poly.test    0    1
         0 1287  216
         1   52  114
```

test 데이터에 대한 오분류율: (52+216)/1669 = 0.1605

(III) Tree-Based Method

방법론 중 분류나무를 이용한 방법도 있다. 또한 트리의 결과를 보면 상위티어, 하위티어에 영향을 많이 주는 변수를 쉽게 예측할 수 있다

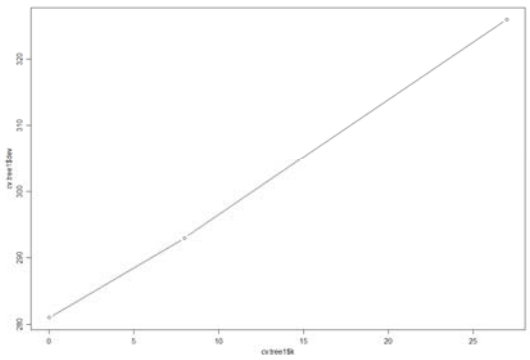
a. Classification tree

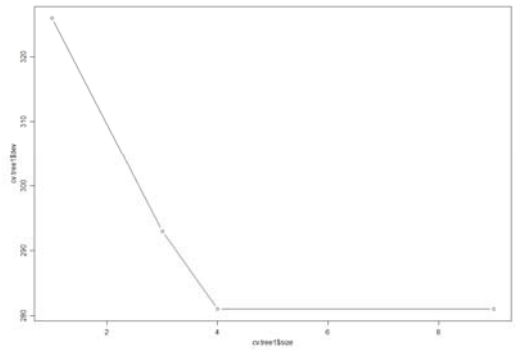


4개의 변수로 8개의 규칙이 만들어졌으며, ActionLatency < 43.2671 & NumberOfPACs > 0.00445462일 때만 상위권 티어로 분류될 수 있다. 사전에 sampling 해놓은 test data 로 예측하면 오분류율 약 0.1678을 얻을 수 있다.

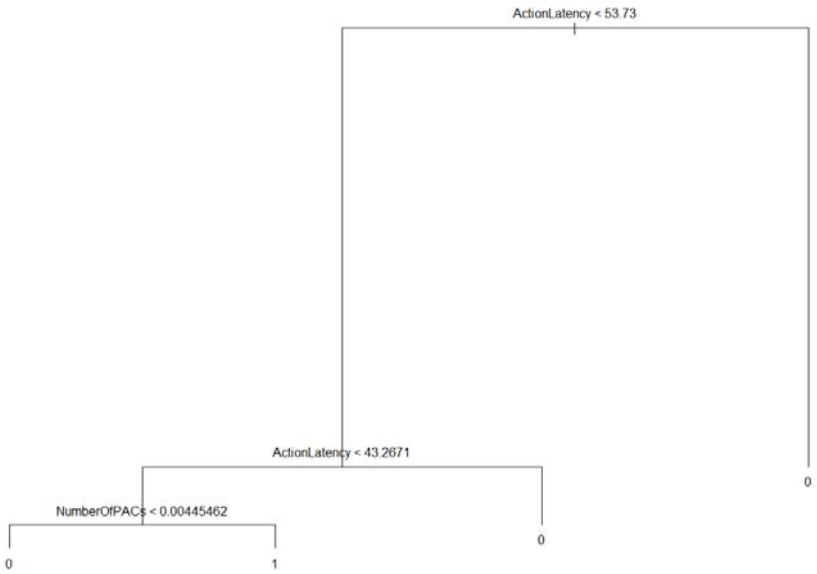
다음은 cv를 통해 가지치기를 한다

b. Pruning tree by cross-validation





「cv.tree1 = cv.tree(tree1,FUN = prune.misclass) -------#r코드」의 결과로 size(9,4,3,1) 중에서 4를 best size로 선택한다



2개의 변수로만 3가지 규칙을 만들어 4가지 경우로 분류한다. ActionLatency<43.2671 & NumberOfPACs >0.00445462 일 때만 상위 티어로 분류된다. 1과 마찬가지로 미리 sampling 해둔 test data로 예측하면 오분류율은 약 0.1678이다. 그림을 비교했을 때 노드의 수는 9개에서 4개로 줄었지만 분류의 결과(오분류율)은 전혀 바뀌지 않아 그림 4가 훨씬 효율적인 tree라고 할 수있다.

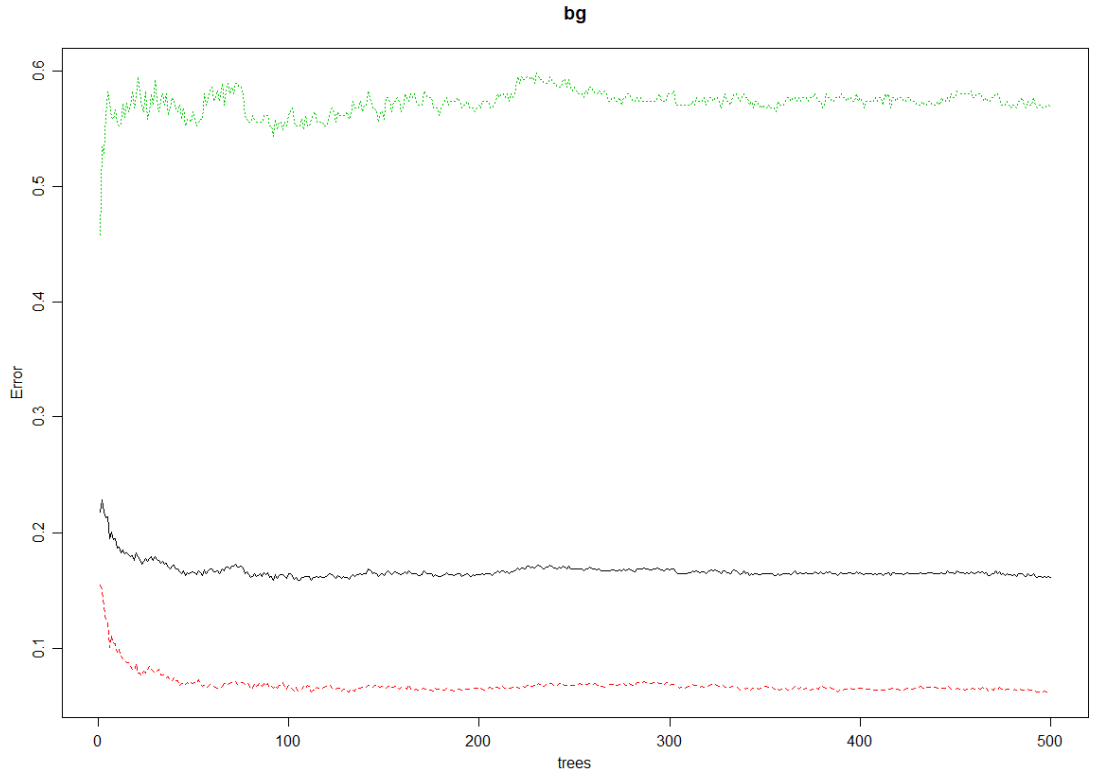
c. Bagging

```
> (bg = randomForest(LeagueIndex~.,data = trd,mtry = 18,importance = T))

Call:
 randomForest(formula = LeagueIndex ~ ., data = trd, mtry = 18,      importance = T)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 18

        OOB estimate of  error rate: 16.12%
Confusion matrix:
     0   1 class.error
0 1259  84  0.06254654
1  185 141  0.56748466
```

training data에 대한 오분류율 0.1612를 얻었다



- green : (1-상위티어를 상위티어라고 예측할확률)
- red : (1- 하위티어를 하위티어라고 예측할확률)
- black : 오분류율

사전에 sampling한 test data 로 예측하면 test 오분류율은 0.1564를 얻을 수있다.

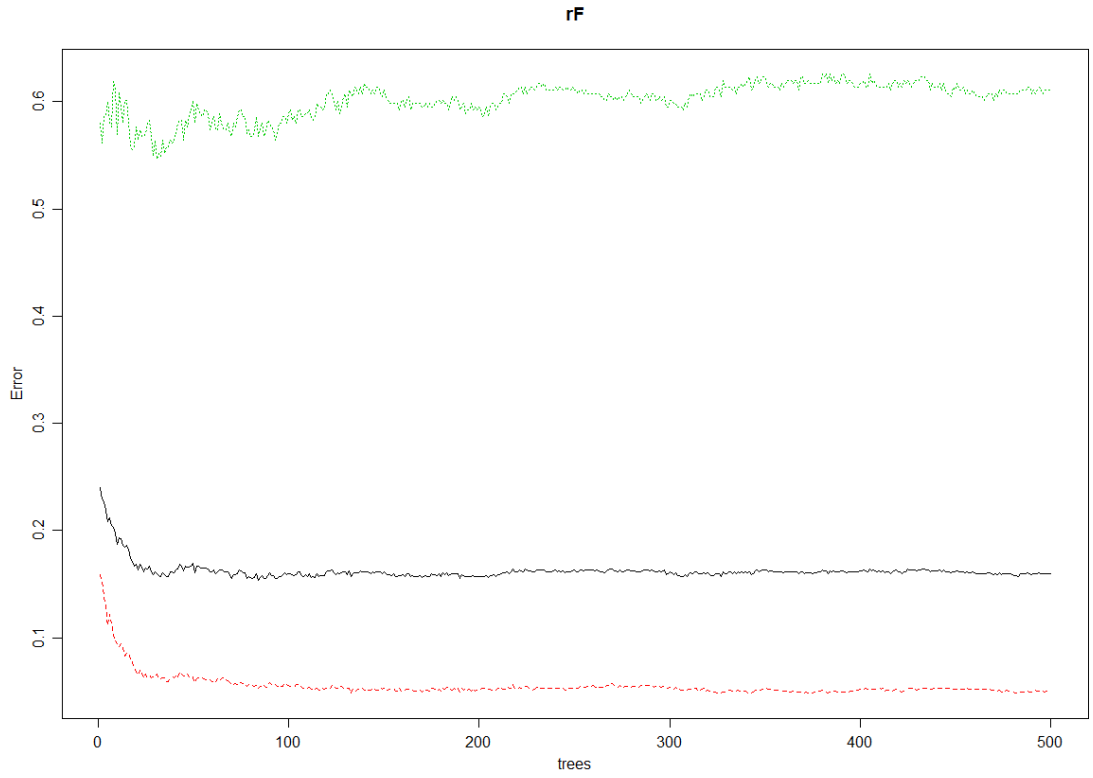
d. RainForest

```
> #rf mtry = sqrt(18)
> (rF = randomForest(LeagueIndex~.,data = trd,mtry = sqrt(18),importance = T))

Call:
 randomForest(formula = LeagueIndex ~ ., data = trd, mtry = sqrt(18),      importance = T)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 4

        OOB estimate of  error rate: 15.94%
Confusion matrix:
     0   1 class.error
0 1276  67  0.04988831
1  199 127  0.61042945
```

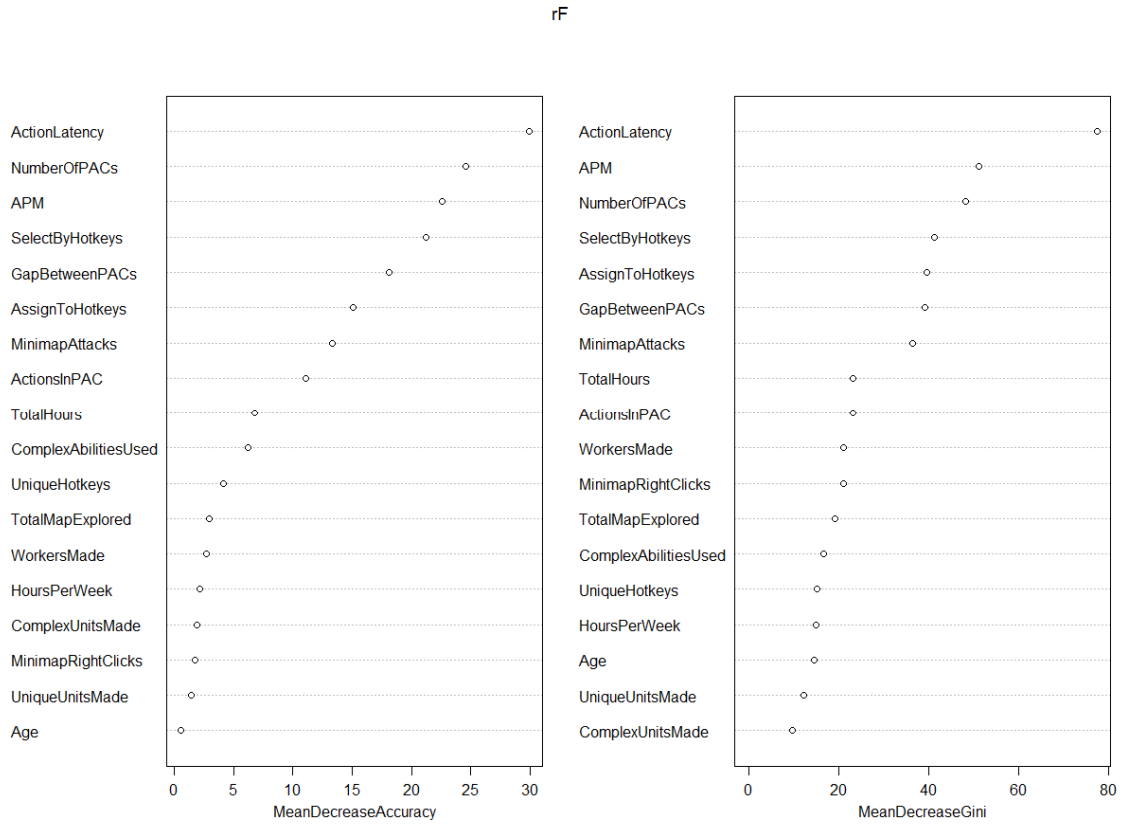
rainforest의 mtry의 값은 설명변수 개수(p=18)의 제곱근을 사용하였으며 training 오분류율 0.1594를 얻었다



- green : (1-상위티어를 상위티어라고 예측할확률)
- red : (1- 하위티어를 하위티어라고 예측할확률)
- black : 오분류율

마찬가지로 test data를 이용하여 예측하면 오분류율은 0.1438로 특히 하위티어를 상위티어로 예측한 경우가 줄어들었다.

마지막으로 varImpPlot함수를 이용하여 중요한 변수들을 살펴보면 하위티어에 중요한 변수들은 ActionLatency, NumberOfPACs, APM, SelectByHotkeys 순으로 말할 수 있다. 상위티어에서는 ActionLatency, APM, NumberOfPACs, SelectByHotkeys 순으로 상위 4개 변수는 동일하다.



3. 결론

(1) 분석 결과 요약

Logistic Regreesion에서는 stepwise selection을 이용하여 18개의 변수 중 13개의 변수만 선택하였고, SelectByHotkeys, APM, ActionLatency 3개의 변수가 가장 티어에 영향을 많이 주는 변수이다. stepwise selction 후 축소된 모델의 오분류율은 약 14.7%를 얻었다.

Logistic Ridge Regression에서는 최적의 lambda값을 구하여 조금 더 유연한 모델을 적합하였다. 벌점화 이후 계수를 분석하여 ActionsInPac, NumberofPACs, GapBetweenPACs 3개의 변수가 가장 티어에 영향을 많이 주는 변수이고, 오분류율은 약 15.2%를 얻었다.

Logistic Lasso Rregression에서도 최적의 lambda값을 구하여 모델을 적합하였다. Ridge와 차이는 기저함수로 몇몇 변수들의 계수를 0으로 만들 수 있다. CV 이후 11개의 변수만 계수를 가지게 되었고 ActionsInPac, HoursPerWeek, GapBetweenPACs 3개의 변수가 가장 영향을 많이 끼치는 변수이다. 적합 모델로 예측한 결과는 약 14.8%의 오분류율을 가져왔다.

마지막 로지스틱 관련 분석으로 주성분 로지스틱 회귀분석을 시행하였다. 주성분 분석으로 18개의 설명변수를 10차원으로 차원축소를 이끌어냈고, 로지스틱 회귀모델 적용단계에서 유의미한 회귀계수 7개만 선택하여 모델을 만들었다. 회귀계수가 가장 큰 2개의 주성분을 분석하여 ActionLatency, APM, GapbetweenPACs가 가장 많은 비중을 차지하고 있는 주성분이였다. 오분류율은 약 16.2%를 얻었다.

이후 4가지 로지스틱 분석에 K-fold를 구현하여 다시 training, test 오분류율을 비교해 보았다. Logistic Regresstion이 가장 좋은 결과를 가져왔고 그에 따라 SelectByHotkeys, APM, ActionLatency 3개의 변수를 가장 중요한 변수로 볼 수 있다.

방법론으로는 KNN, LDA, QDA, SVM, TREE 총 5가지 분석 방법을 사용하였다. KNN(with K-fold)는 모든 분석 방법 중에서 가장 작은 오분류율을 보여주었다. 집합의 개수를 2~10개를 설정하였는데 training 오분류율은 K = 3, test 오분류율은 K = 7에서 가장 작은 오분류율(약 5.5%, 약10.3%)을 보여주었다.

LDA와 QDA는 각각 약 16%, 18%의 오분류율을 얻었으며, SVM은 지지벡터 분류기, 지지벡터 기계(radial, poly kernel) 총 3가지 방법을 사용했으며 training 오분류율 15.5%, 0%, 11.1%를 얻었으며, test 오분류율은 15.5%, 19.4%, 16.1%를 얻었다. 여기서 주목할만한 오분류율은 radial kernel의 결과로 training 오분류율은 0%이다. 추가적으로 살펴보면 1593개의 지지벡터를 가지고 있다. 데이터의 개수인 1669의 95%에 달하는 지지벡터 개수를 가지고 있으므로 과적합의 예시로 볼 수 있다.

마지막 분석 방법인 Tree-Based-Method에선 분류나무, 가지치기, Bagging, RainForest

총 4가지 방법으로 오분류율을 얻었다 각각 약16.8% 16.8% 15.6% 14.4%의 test 오분류율을 얻었으며 가장 중요한 변수로는 ActionLatency, APM, NumberOfPACs, SelectByHotkeys 총 4개를 얻을 수 있었다.
이 데이터는 18개의 설명변수를 가지고 있고, 데이터의 경계가 복잡하기 때문에 KNN의 방법이 가장 좋은 분류 결과를 보여준다. 하지만 KNN에서는 어떠한 변수가 가장 영향을 많이 주는지 분석하기 어렵기 때문에 4가지 로지스틱 회귀분석에서 가장 좋았던 Logistic Regression와 RainForest의 결과를 빌려 가장 영향을 많이 미치는 변수를 꼽으면 ActionLatency ,APM, SelectByHotkeys 3개의 변수를 꼽을 수 있다. 이 3가지 요소가 상위티어와 하위티어의 차이를 가장 잘 설명할 수 있는 변수라고 결론을 내렸다. 즉, 이 데이터는 상위티어로 가기 위해 마우스, 키보드를 사용하는 손 속도와 단축키를 이용하는 속도와 횟수가 중요하다고 말하고 있다.

(2) 분석의 장점 및 한계점

로지스틱 회귀분석 4가지 방법들은 방법론과는 달리 오즈비를 통한 확률 비교가 용이하다. 또한, 어떠한 변수가 유의미한 영향을 주는지 쉽게 파악할 수 있고 반대로 의미가 없는 변수를 찾기도 어렵지 않다. 그러나 Ridge는 의미 없는 변수를 제거해줄 수 없기 때문에 단순히 계수만으로 중요 변수를 예상할 수 밖에 없었다.

장치의 성능에 대한 문제로 지지벡터 분류기와 지지벡터 기계 방법에서 K-fold 방법을 이용한 오분류율을 구하지 못하고 training data와 test data를 임의로 나눠본 것만 진행했다.

(3) 제안사항

사전정보가 완벽하여 변수들의 교호작용을 예상하여 모형에 추가시키거나 장비의 성능이 좋아서 모든 교호작용에 대한 변수를 추가한 모형을 일일이 확인해볼 수 있다면 더 좋은 모형을 만들 수 있을 것이다.

게임 수준이 가장 높은 프로페셔널리그(LeagueIndex=8) 데이터들이 결측치여서 상위랭크에 포함시키지 못했는데 분석에 영향을 줄 수도 있는 프로페셔널리그의 온전한 데이터들을 찾을 수 있다면 보다 정확하게 연구주제에 맞는 분석을 할 수 있을 것이다.