STA567 TakehomeExam

Lina Lee

10/8/2019

Task1

KNN classification

```r
set.seed(12345)
landmod_knn <- train(Class~ .,
                     data=land_sat,
                     method="knn",
                     preProcess=c("center","scale"),
                     tuneGrid=expand.grid(k=seq(1,15,1)),
                     trControl = trainControl(method="cv",number = 5))
# result of KNN classification
landmod_knn$results
```

```
##     k  Accuracy     Kappa  AccuracySD     KappaSD
## 1   1 0.9933174 0.9653992 0.002307162 0.012217075
## 2   2 0.9937833 0.9677545 0.002269058 0.012220981
## 3   3 0.9945603 0.9717339 0.002130883 0.011389662
## 4   4 0.9944049 0.9709510 0.001937919 0.010356164
## 5   5 0.9933171 0.9651757 0.002836491 0.015117528
## 6   6 0.9939387 0.9685611 0.002359808 0.012497305
## 7   7 0.9942495 0.9701429 0.002671883 0.014160450
## 8   8 0.9940942 0.9693390 0.002614488 0.013837004
## 9   9 0.9942493 0.9701228 0.002614945 0.013849167
## 10 10 0.9937834 0.9677676 0.001825127 0.009688628
## 11 11 0.9947162 0.9726016 0.001685503 0.008894928
## 12 12 0.9940943 0.9692941 0.002101220 0.011116746
## 13 13 0.9944054 0.9709634 0.002292521 0.011987249
## 14 14 0.9937839 0.9676858 0.003685862 0.019367096
## 15 15 0.9934731 0.9660535 0.002938520 0.015355428
```

```r
# Give the predictive Accuracy of model
max(landmod_knn$results$Accuracy)
```

```
## [1] 0.9947162
```

LDA classification

```r
set.seed(12345)
landmod_lda <- train(Class~ .,
```

```r
                  data=land_sat,
                  method="lda",
                  preProcess=c("center","scale"),
                  metric="Accuracy",
                  trControl = trainControl(method="cv",number = 5))

# result of LDA
landmod_lda$results

##   parameter  Accuracy   Kappa  AccuracySD    KappaSD
## 1      none 0.9863235 0.92573 0.003914653 0.02245629

# Give the predictive Accuracy of model
landmod_lda$results$Accuracy

## [1] 0.9863235
```

QDA classification

```r
set.seed(12345)
landmod_qda <- train(Class~ .,
                  data=land_sat,
                  method="qda",
                  preProcess=c("center","scale"),
                  trControl = trainControl(method="cv",number = 5))


# result of QDA
landmod_qda$results

##   parameter  Accuracy    Kappa  AccuracySD    KappaSD
## 1      none 0.9692299 0.858611 0.003240305 0.01354962

# Give the predictive Accuracy of model
landmod_qda$results$Accuracy

## [1] 0.9692299
```

Logit regression

```r
set.seed(12345)
landmod_logit <- train(Class~ .,
                     data=land_sat,
                     method="glm",family="binomial",
                     preProcess=c("center","scale"),
                     trControl = trainControl(method="cv",number = 5))


# Give the predictive Accuracy of model
landmod_logit$results$Accuracy
```

```
## [1] 0.9866338
```

Explanation: Since response variable “Class” is a qualitative variable, I tried four types of models including KNN, LDA, QDA, and logistic classification. In KNN classification, sequence from 1 to 15 by 1 was applied into K, there were not parameters for the other models (LDA, QDA, and logistic). For each k-nearest neiborhood model, accuracy was calculated, and the model which has maximum accuracy was selected as final. I fitted the model using train function with method=“method name”. The train function calculated the accuracy of each model by applying 5th fold Cross Validation. I compared the accuracy for each model to find best predictive model. (Accuracy is the percentage of correctly classifies instances out of all instances.) The accuracy for each model is displayed on the table as follows

<Table 1Accuracy of classification models>

Model	Accuracy
KNN(k=11)	0.9947
LDA	0.9863235
QDA	0.9692299
Logistic	0.9866338

KNN classification model (k=11) best predicts the Class for the satellite images (cotton crop or soil) using the Sp11, Sp12, . . . , Sp49 variables in the land_sat dataset since the model has the highest accuracy.

Task2

```r
# Remove missing values
gaming<-gaming[!(gaming$Age=="?"),]
gaming<-gaming[!(gaming$HoursPerWeek=="?"),]
gaming<-gaming[!(gaming$TotalHours=="?"),]

# mutate factor variables into numeric variables
gaming <- gaming %>%
mutate(Age=as.numeric(Age),HoursPerWeek=as.numeric(HoursPerWeek),TotalHours=a
s.numeric(TotalHours))
```

MLR

```r
# Backward Stepwise Regression from AIC
gaming_mod <- lm(APM~. ,data=gaming)
stepBackward <- step(gaming_mod)

stepBackward


## Call:
## lm(formula = APM ~ Age + SelectByHotkeys + MinimapAttacks +
MinimapRightClicks +
##     NumberOfPACs + ActionLatency + ActionsInPAC + TotalMapExplored +
```

```
##     WorkersMade, data = gaming)
##
## Coefficients:
##        (Intercept)                 Age     SelectByHotkeys
##         -4.686e+01          -1.132e-01           5.529e+03
##     MinimapAttacks  MinimapRightClicks        NumberOfPACs
##          2.835e+03           3.946e+03           2.333e+04
##      ActionLatency         ActionsInPAC    TotalMapExplored
##         -1.723e-01           1.280e+01          -3.826e-02
##        WorkersMade
##          2.594e+03
```

```
# Cross Validation
set.seed(12345)
gaming_mlr <- train(APM~. ,
              data=gaming,
              method="lm",
              trControl=trainControl(method="cv",number = 5),
              preProcess = c("center", "scale"))



# RMSE calculated from 5th-fold cross validation
min(gaming_mlr$results$RMSE)
```

```
## [1] 8.078973
```

KNN regression

```
set.seed(12345)
gaming_knn <- train(APM~. ,
                    data=gaming,
                    method="knn",
                    preProcess=c("center","scale"),
                    tuneGrid=expand.grid(k=seq(1,50,1)),
                    trControl = trainControl(method="cv",number = 5))



# Tuning parameter of the final KNN regression
gaming_knn$bestTune
```

```
##   k
## 8 8
```

```
# RMSE of the final model calculated from 5th-fold cross validation
min(gaming_knn$results$RMSE)
```

```
## [1] 15.2585
```

lasso regression

```r
set.seed(12345)
gaming_lasso<-train(APM~. ,
                    data=gaming,
                    method="glmnet",
                    trControl=trainControl(method="cv",number=5),
                    preProcess = c("center","scale"),
                    tuneGrid=expand.grid(alpha=0,lambda=seq(0,5,0.5))
                    )


# Tuning parameter of the final lasso model
gaming_lasso$bestTune

##   alpha lambda
## 8     0    3.5

# RMSE of the final lasso model calculated from 5th-fold cross validation
min(gaming_lasso$results$RMSE)

## [1] 8.912704
```

ridge regression

```r
set.seed(12345)
gaming_ridge<-train(APM~. ,
                    data=gaming,
                    method="glmnet",
                    trControl=trainControl(method="cv",number=5),
                    preProcess = c("center","scale"),
                    tuneGrid=expand.grid(alpha=1,lambda=seq(0,5,0.1))
)


# Tuning parameter
gaming_ridge$bestTune

##   alpha lambda
## 2     1    0.1

# RMSE calculated from 5th-fold cross validation
min(gaming_ridge$results$RMSE)

## [1] 8.080087
```

Enet

```
### elastic net
set.seed(12345)
gam_mod_enet <- train(APM~ .,
                   data=gaming,
                   method="glmnet",
                   trControl=trainControl(method="cv"),
                   preProcess = c("center", "scale"),
                   tuneLength=20)

## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info =
## trainInfo, : There were missing values in resampled performance measures.

# Tuning parameter
gam_mod_enet$bestTune

##     alpha   lambda
## 306     1 0.108901

# RMSE calculated from 5th-fold cross validation
min(gam_mod_enet$results$RMSE)

## [1] 8.012523
```

pcr

```
set.seed(12345)
gaming_pcr <- train(APM~ .,
                   data=gaming,
                   method="pcr",
                   preProcess=c("center","scale"),
                   trControl = trainControl(method="cv"),
                   tuneGrid = data.frame(ncomp=1:13))



# Tuning parameter
gaming_pcr$bestTune

##    ncomp
## 13    13

# RMSE calculated from 5th-fold cross validation
min(gaming_pcr$results$RMSE)

## [1] 8.010184
```

plsr

```
set.seed(12345)
gaming_plsr <- train(APM~ .,
                     data=gaming,
                     method="pls",
                     preProcess=c("center","scale"),
                     trControl = trainControl(method="cv"),
                     tuneGrid = data.frame(ncomp=1:13))

# Tuning parameter
gaming_plsr$bestTune

##    ncomp
## 10    10

# RMSE calculated from 5th-fold cross validation
min(gaming_plsr$results$RMSE)

## [1] 8.010172
```

Explanation: Since response is quantitative variable, I tried Multivariate linear regression (MLR), K-Nearest Neiborhood regression, Lasso, Ridge, Elastic net(Enet), Principal Component(PCR), and partial Least Squre regression(PLSR). Before fitting MLR, I selected variables using backward stepwise approach. The selected model for MLR are APM ~ Age + SelectByHotkeys + MinimapAttacks + MinimapRightClicks + NumberOfPACs + ActionLatency + ActionsInPAC + TotalMapExplored + WorkersMade. In KNN regression, I applied sequence from 1 to 15 by 1 into k. For each k-nearest neiborhood model, RMSE was calculated, and the model which has minimum RMSE was selected as final. In Lasso, alpha is 0, and sequence from 0 to 5 by 0.1 were used for lambda. For ridge, alpha is 1, and sequence from 0 to 5 by 0.1 were applied into lambda. In both PCR and PLSR, 1 to 13 number of components were tried because the number of components should be lower than the number of variables (14 in this case). I applied 5th fold Cross Validation to calculate RMSE of each model. In each type of model, different tuning parameters provided different RMSE. Among them, the model which has the minimum RMSE was chosen as final model. Finally, I compared RMSE to find best predictive model. The tuning parameters and RMSE for each model are displayed on the table below.

<Table 2RMSE of models>

Model	Tuning Parameter	RMSE
MRL		8.0789
KNN(k=7)	k=8	15.2849
lasso	lambda=3.5	8.9127
ridge	lambda=0.1	8.0800
enet	alpha=1, lambda=0.29	8.0125
PCR	ncomp=13	8.01018
PLSR	ncomp=10	8.01017

Partial least squares regression has the lowest RMSE as 8.01017, and Principal component regression has second lowest RMSE as 8.01018, which is almost similar with that of PCR. Therefore, I think both PLSR and PCR models best predict the APM for the gaming records in the gaming using all other numeric variables in the gaming dataset.

Reference

https://machinelearningmastery.com/machine-learning-evaluation-metrics-in-r/