stsci3740 final project modeling

Fiona Huang

2025-04-25

```r
# import datasets
red <- read.csv("C:/Users/xinya/Downloads/Cornell Classes/STSCI 3740/final project/winequality-red.csv"

white <- read.csv("C:/Users/xinya/Downloads/Cornell Classes/STSCI 3740/final project/winequality-white.

wine <- read.csv("C:/Users/xinya/Downloads/Cornell Classes/STSCI 3740/final project/wine-quality-white-
```

Fitting a KNN Model

```r
# normalize the data using z-score
normalize <- function(x) {
  return((x - mean(x)) / sd(x))
}

# wine_norm <- wine %>%
#   mutate(across(where(is.numeric) & !where(is.factor), normalize))

all_columns <- names(wine)
columns_to_normalize <- all_columns[all_columns != "quality" & sapply(wine, is.numeric)]
wine_norm <- wine
wine_norm[columns_to_normalize] <- lapply(wine[columns_to_normalize], normalize)



# change type of wine to white=1, red=2
wine_norm$type <- as.numeric(factor(wine_norm$type))

# split the dataset into train/test
set.seed(1)
index <- sample(1:nrow(wine_norm), size=nrow(wine_norm)*0.7, rep=FALSE)
training <- wine_norm[index, ]
testing <- wine_norm[-index, ]

training_X <- training %>% select(-quality)
testing_X <- testing %>% select(-quality)
```

```r
# try different values of k
k.values <- 1:20
```

```
knn.errors <- sapply(k.values, function(k) {
  knn.pred <- knn(training_X, testing_X, training$quality, k=k)
  mean(knn.pred != testing$quality)
})

print(knn.errors)
```

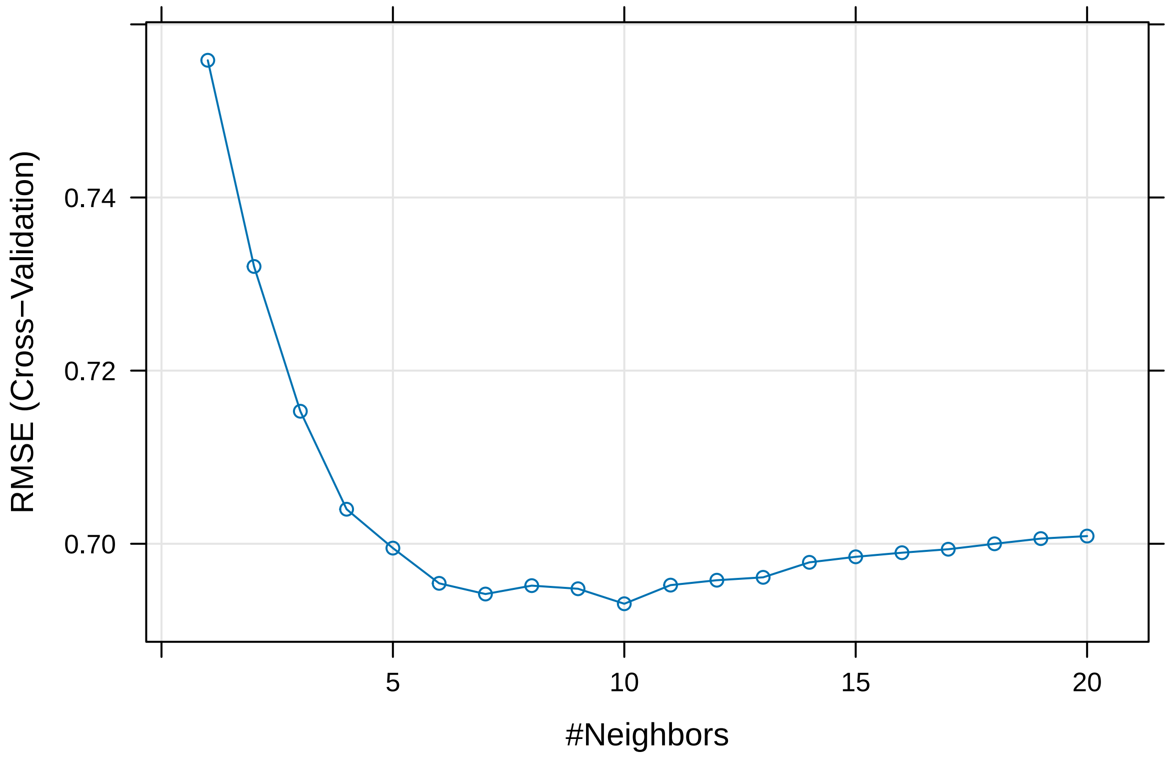
```
##  [1] 0.3989744 0.4794872 0.4707692 0.4574359 0.4482051 0.4456410 0.4507692
##  [8] 0.4528205 0.4451282 0.4528205 0.4364103 0.4420513 0.4317949 0.4379487
## [15] 0.4358974 0.4358974 0.4461538 0.4415385 0.4389744 0.4405128
```

The value of k that seems to perform the best on this data is k=1.

```
set.seed(1)
# 10-fold cross validation
control <- trainControl(method = "cv", number = 10)

knn_cv <- train(
  quality ~ .,
  data = wine_norm,
  method = "knn",
  trControl = control,
  tuneGrid = expand.grid(k = 1:20)
)

knn_cv
```

```
## k-Nearest Neighbors
##
## 6497 samples
##   12 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 5847, 5846, 5847, 5847, 5847, 5848, ...
## Resampling results across tuning parameters:
##
##   k   RMSE       Rsquared   MAE
##    1  0.7558513  0.3795189  0.4239693
##    2  0.7320306  0.3612061  0.4939238
##    3  0.7153067  0.3654421  0.5125768
##    4  0.7039897  0.3718375  0.5170525
##    5  0.6994976  0.3727748  0.5246780
##    6  0.6954288  0.3756623  0.5260905
##    7  0.6941926  0.3755005  0.5286038
##    8  0.6951634  0.3725416  0.5314023
##    9  0.6948018  0.3720174  0.5344289
##   10  0.6930709  0.3743660  0.5342596
##   11  0.6952228  0.3699285  0.5365115
##   12  0.6957883  0.3683444  0.5382253
##   13  0.6961228  0.3676329  0.5401778
##   14  0.6978546  0.3643808  0.5417053
##   15  0.6984920  0.3632550  0.5426139
##   16  0.6989681  0.3623560  0.5431027
##   17  0.6993684  0.3614560  0.5446430
##   18  0.6999960  0.3602518  0.5456382
##   19  0.7005973  0.3591466  0.5467203
##   20  0.7008890  0.3587130  0.5473792
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 10.
```

```
plot(knn_cv)
```



```
model <- knn(training_X, testing_X, training$quality, k=10)

confusion_matrix <- table(Predicted = model, Actual = testing$quality)
print(confusion_matrix)
```

```
##          Actual
## Predicted   3   4   5   6   7   8   9
##         3   0   0   0   0   0   0   0
##         4   0   2   5   0   0   0   0
##         5   4  32 392 186  16   4   0
##         6   4  33 219 547 155  22   1
##         7   0   0  19 123 144  31   1
##         8   0   0   0   5   2   2   1
##         9   0   0   0   0   0   0   0
```

```
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
accuracy
```

```
## [1] 0.5574359
```

```
# fitting the knn models using less number of variables
X <- c("alcohol", "volatile.acidity", "residual.sugar", "total.sulfur.dioxide")

# use RFE (recursive feature elimination to select different subsets)
rfe_ctrl <- rfeControl(functions = caretFuncs, method = "cv", number = 10)
rfe_model <- rfe(wine_norm %>% select(-quality),
                 wine_norm$quality,
                 sizes = c(2:11), # test sets with 2 to 11 variables
                 rfeControl = rfe_ctrl,
                 method = "knn")

# Best variable set
print(rfe_model$optVariables)
```

```
# plot the model performance as a function of number of variables
plot(rfe_model, type = c("g", "o"))
```

```
new_wine <- wine
new_wine$type <- as.numeric(factor(new_wine$type))

# Fit a linear model
lm_model <- lm(quality ~ ., data = new_wine)

# Stepwise model selection
step_model <- step(lm_model, direction = "backward")
```

```
## Start:  AIC=-4021.28
## quality ~ type + fixed.acidity + volatile.acidity + citric.acid +
##     residual.sugar + chlorides + free.sulfur.dioxide + total.sulfur.dioxide +
##     density + pH + sulphates + alcohol
##
##                        Df Sum of Sq    RSS     AIC
## - citric.acid           1     0.332 3485.1 -4022.7
## <none>                              3484.7 -4021.3
## - chlorides             1     2.755 3487.5 -4018.1
## - total.sulfur.dioxide  1    10.092 3494.8 -4004.5
## - fixed.acidity         1    15.650 3500.4 -3994.2
## - pH                    1    16.295 3501.0 -3993.0
## - type                  1    21.785 3506.5 -3982.8
## - free.sulfur.dioxide   1    22.312 3507.1 -3981.8
## - density               1    28.235 3513.0 -3970.8
## - sulphates             1    48.161 3532.9 -3934.1
## - residual.sugar        1    59.499 3544.2 -3913.3
## - alcohol               1    81.573 3566.3 -3872.9
## - volatile.acidity      1   180.876 3665.6 -3694.5
##
## Step:  AIC=-4022.66
## quality ~ type + fixed.acidity + volatile.acidity + residual.sugar +
##     chlorides + free.sulfur.dioxide + total.sulfur.dioxide +
##     density + pH + sulphates + alcohol
##
##                        Df Sum of Sq    RSS     AIC
## <none>                              3485.1 -4022.7
## - chlorides             1     3.168 3488.2 -4018.8
## - total.sulfur.dioxide  1    10.537 3495.6 -4005.0
## - fixed.acidity         1    15.390 3500.5 -3996.0
## - pH                    1    16.775 3501.9 -3993.5
## - free.sulfur.dioxide   1    22.351 3507.4 -3983.1
## - type                  1    22.479 3507.6 -3982.9
## - density               1    28.697 3513.8 -3971.4
## - sulphates             1    47.876 3533.0 -3936.0
## - residual.sugar        1    59.787 3544.9 -3914.1
## - alcohol               1    81.509 3566.6 -3874.5
## - volatile.acidity      1   197.707 3682.8 -3666.2
```

```
step_model
```

```
##
## Call:
## lm(formula = quality ~ type + fixed.acidity + volatile.acidity +
##     residual.sugar + chlorides + free.sulfur.dioxide + total.sulfur.dioxide +
##     density + pH + sulphates + alcohol, data = new_wine)
##
## Coefficients:
##          (Intercept)                  type         fixed.acidity
##            1.058e+02            -3.655e-01             8.244e-02
##     volatile.acidity        residual.sugar             chlorides
##           -1.471e+00             6.256e-02            -8.007e-01
##  free.sulfur.dioxide  total.sulfur.dioxide               density
##            4.941e-03            -1.427e-03            -1.046e+02
##                   pH             sulphates               alcohol
##            5.044e-01             7.186e-01             2.210e-01
```

```
# try stepwise with normalized data

# Fit a linear model
lm_model <- lm(quality ~ ., data = wine_norm)

# Stepwise model selection
step_model <- step(lm_model, direction = "both")
```

```
## Start:  AIC=-4021.28
## quality ~ type + fixed.acidity + volatile.acidity + citric.acid +
##     residual.sugar + chlorides + free.sulfur.dioxide + total.sulfur.dioxide +
##     density + pH + sulphates + alcohol
##
##                        Df Sum of Sq    RSS     AIC
## - citric.acid           1     0.332 3485.1 -4022.7
## <none>                              3484.7 -4021.3
## - chlorides             1     2.755 3487.5 -4018.1
## - total.sulfur.dioxide  1    10.092 3494.8 -4004.5
## - fixed.acidity         1    15.650 3500.4 -3994.2
## - pH                    1    16.295 3501.0 -3993.0
## - type                  1    21.785 3506.5 -3982.8
## - free.sulfur.dioxide   1    22.312 3507.1 -3981.8
## - density               1    28.235 3513.0 -3970.8
## - sulphates             1    48.161 3532.9 -3934.1
## - residual.sugar        1    59.499 3544.2 -3913.3
## - alcohol               1    81.573 3566.3 -3872.9
## - volatile.acidity      1   180.876 3665.6 -3694.5
##
## Step:  AIC=-4022.66
## quality ~ type + fixed.acidity + volatile.acidity + residual.sugar +
##     chlorides + free.sulfur.dioxide + total.sulfur.dioxide +
##     density + pH + sulphates + alcohol
##
##                        Df Sum of Sq    RSS     AIC
## <none>                              3485.1 -4022.7
## + citric.acid           1     0.332 3484.7 -4021.3
## - chlorides             1     3.168 3488.2 -4018.8
## - total.sulfur.dioxide  1    10.537 3495.6 -4005.0
## - fixed.acidity         1    15.390 3500.5 -3996.0
## - pH                    1    16.775 3501.9 -3993.5
## - free.sulfur.dioxide   1    22.351 3507.4 -3983.1
## - type                  1    22.479 3507.6 -3982.9
## - density               1    28.697 3513.8 -3971.4
## - sulphates             1    47.876 3533.0 -3936.0
## - residual.sugar        1    59.787 3544.9 -3914.1
## - alcohol               1    81.509 3566.6 -3874.5
## - volatile.acidity      1   197.707 3682.8 -3666.2
```

```
step_model
```

```
##
## Call:
## lm(formula = quality ~ type + fixed.acidity + volatile.acidity +
##     residual.sugar + chlorides + free.sulfur.dioxide + total.sulfur.dioxide +
##     density + pH + sulphates + alcohol, data = wine_norm)
##
## Coefficients:
##          (Intercept)                  type         fixed.acidity
##              6.45936              -0.36546               0.10688
##     volatile.acidity        residual.sugar             chlorides
##             -0.24220               0.29767              -0.02805
##  free.sulfur.dioxide  total.sulfur.dioxide               density
##              0.08770              -0.08065              -0.31358
##                   pH             sulphates               alcohol
##              0.08111               0.10694               0.26354
```

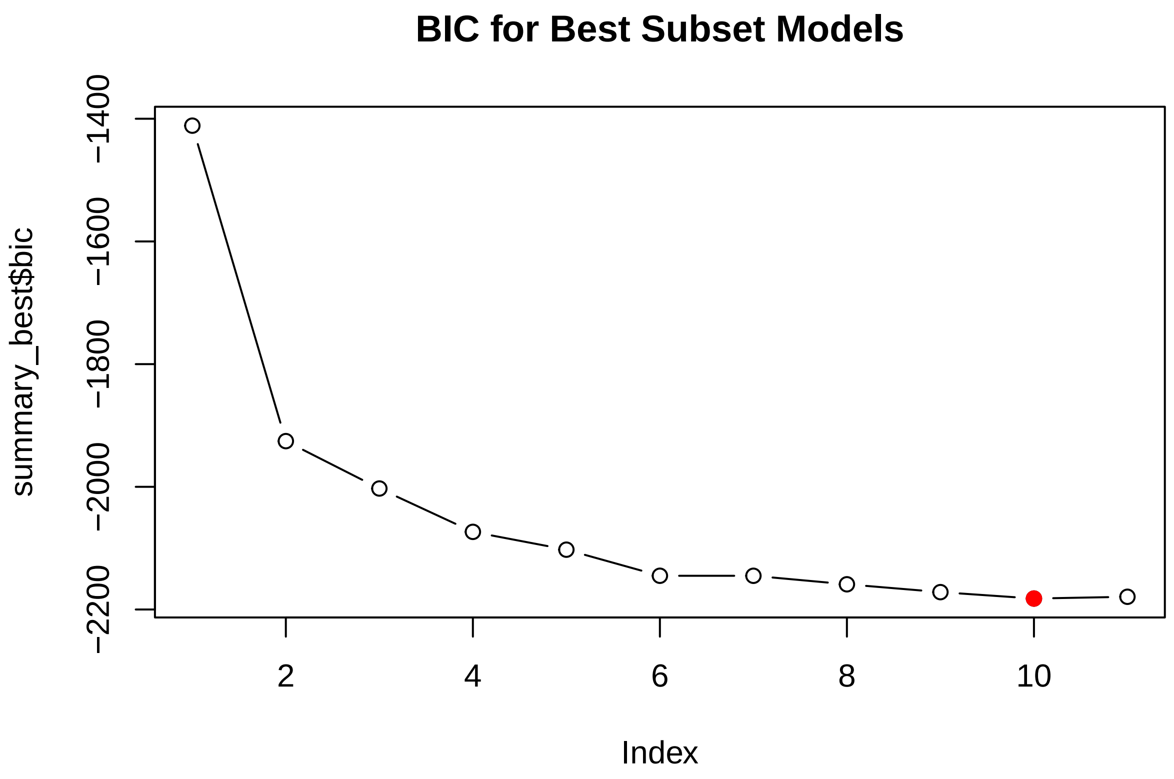
```
# best subset selection
library(leaps)

best_fit <- regsubsets(quality ~ ., data = new_wine, nvmax = 11)
summary(best_fit)
```

```
## Subset selection object
## Call: regsubsets.formula(quality ~ ., data = new_wine, nvmax = 11)
## 12 Variables  (and intercept)
##                      Forced in Forced out
## type                     FALSE      FALSE
## fixed.acidity            FALSE      FALSE
## volatile.acidity         FALSE      FALSE
## citric.acid              FALSE      FALSE
## residual.sugar           FALSE      FALSE
## chlorides                FALSE      FALSE
## free.sulfur.dioxide      FALSE      FALSE
## total.sulfur.dioxide     FALSE      FALSE
## density                  FALSE      FALSE
## pH                       FALSE      FALSE
## sulphates                FALSE      FALSE
## alcohol                  FALSE      FALSE
## 1 subsets of each size up to 11
## Selection Algorithm: exhaustive
##           type fixed.acidity volatile.acidity citric.acid residual.sugar
## 1  ( 1 )  " "  " "           " "              " "         " "           
## 2  ( 1 )  " "  " "           "*"              " "         " "           
## 3  ( 1 )  " "  " "           "*"              " "         " "           
## 4  ( 1 )  " "  " "           "*"              " "         "*"           
## 5  ( 1 )  "*"  " "           "*"              " "         "*"           
## 6  ( 1 )  " "  " "           "*"              " "         "*"           
## 7  ( 1 )  "*"  " "           "*"              " "         "*"           
## 8  ( 1 )  "*"  " "           "*"              " "         "*"           
## 9  ( 1 )  "*"  "*"           "*"              " "         "*"           
## 10  ( 1 ) "*"  "*"           "*"              " "         "*"           
## 11  ( 1 ) "*"  "*"           "*"              " "         "*"           
##           chlorides free.sulfur.dioxide total.sulfur.dioxide density pH 
## 1  ( 1 )  " "       " "                 " "                  " "     " "
## 2  ( 1 )  " "       " "                 " "                  " "     " "
## 3  ( 1 )  " "       " "                 " "                  " "     " "
## 4  ( 1 )  " "       " "                 " "                  " "     " "
## 5  ( 1 )  " "       " "                 " "                  " "     " "
## 6  ( 1 )  " "       "*"                 "*"                  " "     " "
## 7  ( 1 )  " "       "*"                 " "                  "*"     " "
## 8  ( 1 )  " "       "*"                 "*"                  "*"     " "
## 9  ( 1 )  " "       "*"                 " "                  "*"     "*"
## 10  ( 1 ) " "       "*"                 "*"                  "*"     "*"
## 11  ( 1 ) "*"       "*"                 "*"                  "*"     "*"
##           sulphates alcohol
## 1  ( 1 )  " "       "*"    
## 2  ( 1 )  " "       "*"    
## 3  ( 1 )  "*"       "*"    
## 4  ( 1 )  "*"       "*"    
## 5  ( 1 )  "*"       "*"    
## 6  ( 1 )  "*"       "*"    
## 7  ( 1 )  "*"       "*"    
## 8  ( 1 )  "*"       "*"    
## 9  ( 1 )  "*"       "*"    
## 10  ( 1 ) "*"       "*"    
## 11  ( 1 ) "*"       "*"    
```

```
# find model size with lowest BIC
summary_best <- summary(best_fit)
which.min(summary_best$bic)
```

```
## [1] 10
```

```
plot(summary_best$bic, type = "b", main = "BIC for Best Subset Models")
points(which.min(summary_best$bic), summary_best$bic[which.min(summary_best$bic)],
       col = "red", pch = 19)
```



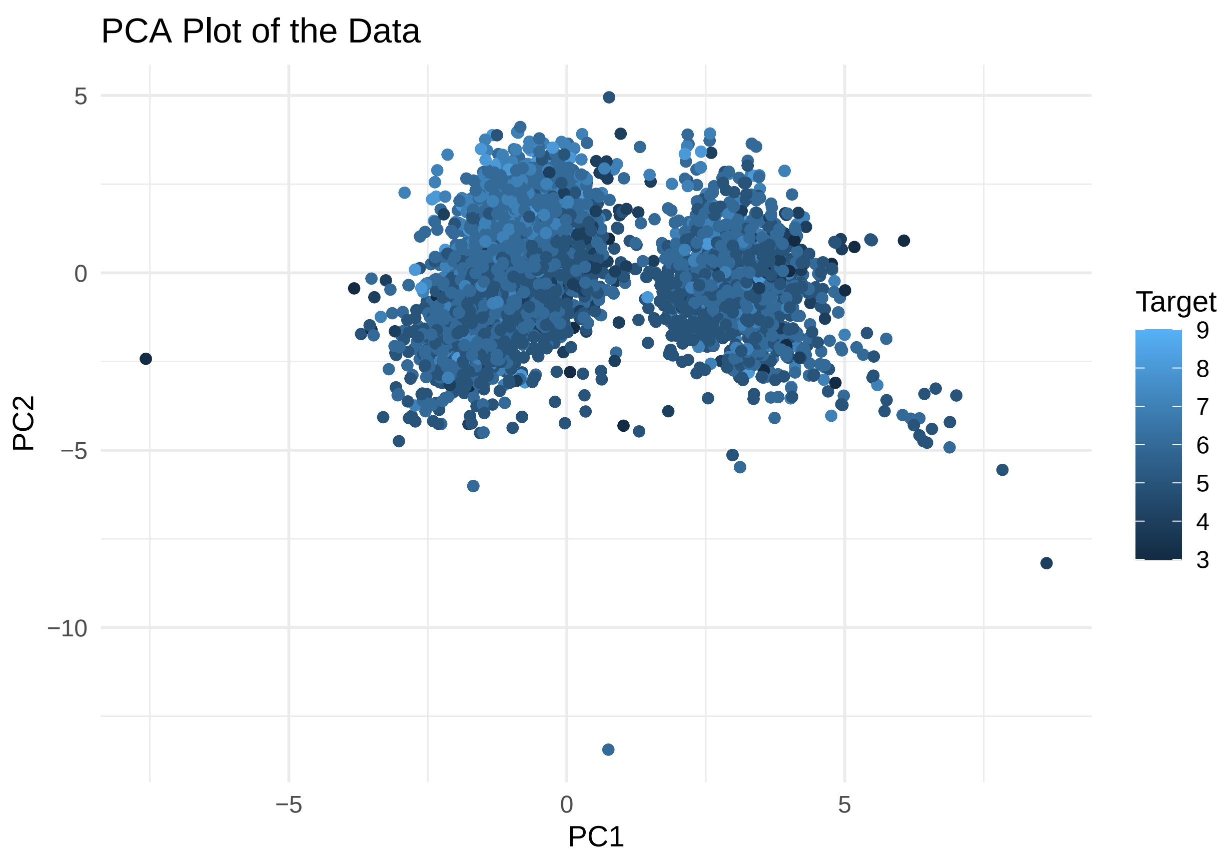
```
# apply PCA before fitting KNN
new_wine <- wine
new_wine$type <- as.numeric(factor(new_wine$type))

target <- new_wine$quality
predictors <- new_wine %>% select(-quality)

# standardize data
scaled_data <- scale(predictors)

# Perform PCA
pca_result <- prcomp(scaled_data, center = TRUE, scale. = TRUE)
summary(pca_result)
```

```
## Importance of components:
```

```
##                           PC1    PC2    PC3    PC4     PC5     PC6    PC7
## Standard deviation     1.9518 1.5902 1.2496 0.9853 0.85077 0.78329 0.7324
## Proportion of Variance 0.3175 0.2107 0.1301 0.0809 0.06032 0.05113 0.0447
## Cumulative Proportion  0.3175 0.5282 0.6583 0.7392 0.79952 0.85065 0.8953
##                            PC8     PC9   PC10    PC11    PC12
## Standard deviation     0.70921 0.59368 0.5068 0.34552 0.15539
## Proportion of Variance 0.04192 0.02937 0.0214 0.00995 0.00201
## Cumulative Proportion  0.93727 0.96664 0.9880 0.99799 1.00000
```

```
# reduce dimensionality
# choose the first two PCs
pca_data <- pca_result$x[, 1:2]
```

```
# plot the first two PCs
pca_df <- data.frame(PC1 = pca_data[, 1], PC2 = pca_data[, 2], Target = target)
ggplot(pca_df, aes(x = PC1, y = PC2, color = Target)) +
  geom_point() +
  theme_minimal() +
  ggtitle("PCA Plot of the Data")
```

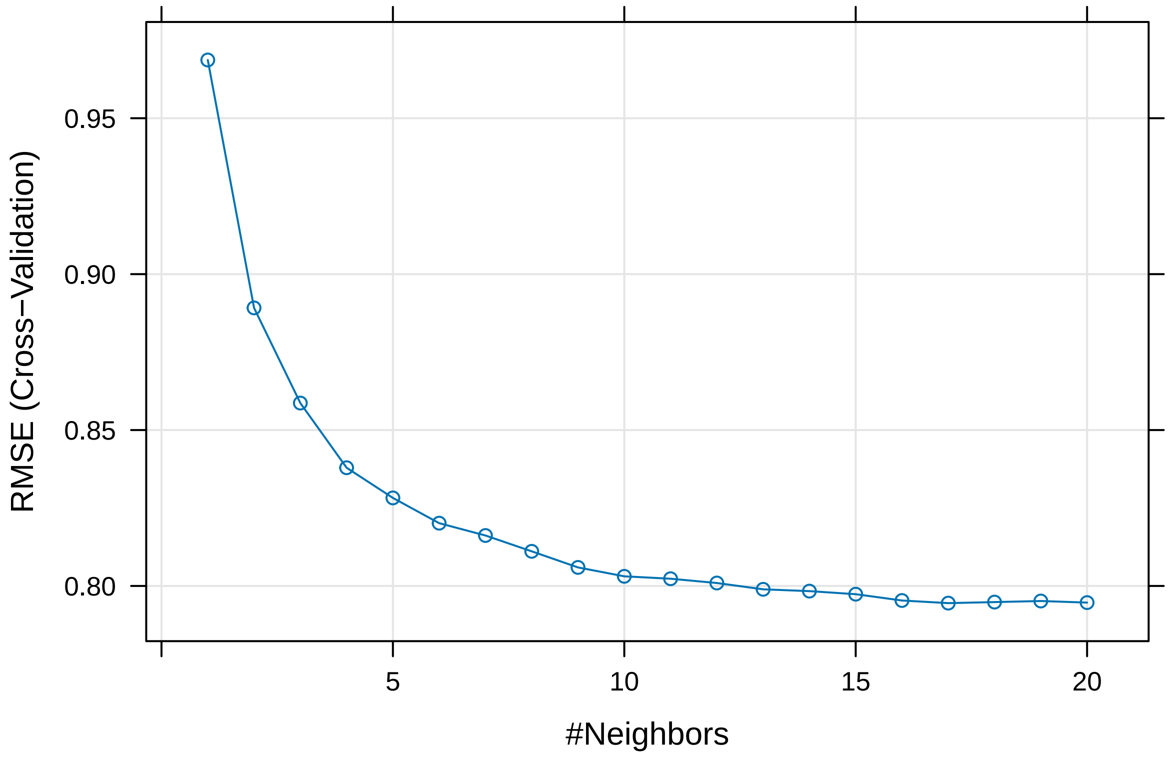


```
# apply KNN on PCA-reduced data
set.seed(1)
index_2 <- sample(1:nrow(pca_data), size=nrow(pca_data)*0.7, rep=FALSE)
training_data <- pca_data[index_2, ]
testing_data <- pca_data[-index_2, ]
```

```r
training_targt <- target[index_2]
testing_target <- target[-index_2]

# 10-fold cross validation
control_new <- trainControl(method = "cv", number = 10)

knn_cv_2 <- train(
  training_data,
  training_targt,
  method = "knn",
  trControl = control_new,
  tuneGrid = expand.grid(k = 1:20)
)

knn_cv_2
```

```
## k-Nearest Neighbors
##
## 4547 samples
##    2 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 4091, 4091, 4093, 4092, 4092, 4091, ...
## Resampling results across tuning parameters:
##
##   k   RMSE       Rsquared   MAE
##    1  0.9686765  0.1472305  0.6149149
##    2  0.8891703  0.1484428  0.6421051
##    3  0.8586539  0.1512376  0.6453570
##    4  0.8379117  0.1576487  0.6414926
##    5  0.8282310  0.1565788  0.6409306
##    6  0.8201285  0.1584923  0.6402407
##    7  0.8161855  0.1595432  0.6400824
##    8  0.8111004  0.1627934  0.6381061
##    9  0.8059442  0.1685110  0.6372944
##   10  0.8030868  0.1706833  0.6364861
##   11  0.8023147  0.1701515  0.6368780
##   12  0.8009456  0.1710912  0.6355124
##   13  0.7989230  0.1732253  0.6344440
##   14  0.7983415  0.1734361  0.6334287
##   15  0.7973559  0.1741752  0.6327819
##   16  0.7953342  0.1769786  0.6309361
##   17  0.7944945  0.1780838  0.6306238
##   18  0.7948295  0.1767531  0.6314171
##   19  0.7951690  0.1756815  0.6323106
##   20  0.7946628  0.1763394  0.6324910
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 17.
```

```r
plot(knn_cv_2)
```



```r
knn_model <- knn(train = training_data, test = testing_data, cl = training_targt, k = 17)

confusion_matrix <- table(Predicted = knn_model, Actual = testing_target)
print(confusion_matrix)
```

```
##          Actual
## Predicted   3   4   5   6   7   8   9
##         3   0   0   0   0   0   0   0
##         4   0   2   3   2   0   0   0
##         5   5  31 302 238  59  10   1
##         6   3  28 313 517 179  34   1
##         7   0   6  17 104  79  15   1
##         8   0   0   0   0   0   0   0
##         9   0   0   0   0   0   0   0
```

```r
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
accuracy
```

```
## [1] 0.4615385
```