DAR F23 Hockey Analytics

Hockey Analytics

Jeff Jung

2023-10-31

Contents

Weekly Work Summary

NOTE: Follow an outline format; use bullets to express individual points.

- RCS ID: jungj6
- Project Name: Hockey Analytics
- Summary of work since last week
  - I did cluster analysis with categorical and continous models, which also involved in using the elbow test to find the ideal number of clusters.
  - I calculated the balanced accuracy for logistic models for categorical and original datasets with updated and added variables.
- NEW: Summary of github issues added and worked
  - None
- Summary of github commits
  - Added assignment 5
- List of presentations, papers, or other outputs
  - None
- List of references (if necessary)
- Indicate any use of group shared code base
- Indicate which parts of your described work were done by you or as part of joint efforts
- **Required:** Provide illustrating figures and/or tables
  - Added below

Personal Contribution

- Clearly defined, unique contribution(s) done by you: code, ideas, writing...
  - I added the categorized dataset to StudentData so everyone can use it

Analysis: Accuracy analysis for categorical data vs continuous data

Question being asked

I am interested in finding out whether categorical data is better or worse at predicting the outcome of the goal.

Data Preparation

I had set the quantiles into thirds, and using those qunantiles, I created the categorical data for each continous variable. Then I saved that categorized dataframe into Rds file, so everyone can use it. Since we are trying to access the accuracy of data, I converted outcomes.goal into a binary variable. I dropped some variables to better estimate the model.

```
# Include all data processing code (if necessary), clearly commented

# Load required library
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidymodels)
```

```
## -- Attaching packages -------------------------------------- tidymodels 1.1.1 --

## v broom        1.0.5     v rsample      1.2.0
## v dials        1.2.0     v tibble       3.2.1
## v ggplot2      3.4.3     v tidyr        1.3.0
## v infer        1.0.5     v tune         1.1.2
## v modeldata    1.2.0     v workflows    1.1.3
## v parsnip      1.1.1     v workflowsets 1.0.1
## v purrr        1.0.2     v yardstick    1.2.0
## v recipes      1.0.8

## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x purrr::discard() masks scales::discard()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
## x recipes::step()  masks stats::step()
## * Use tidymodels_prefer() to resolve common conflicts.
```

```
shots_stats_goal.df <- readRDS("shots_stats_goal.df.Rds")

# Calculate quantiles to divide the data into thirds
puckDist_q <- quantile(shots_stats_goal.df$puckDist, probs = c(1/3, 2/3))
```

```r
puckAngle_q <- quantile(shots_stats_goal.df$puckAngle, probs = c(1/3, 2/3))
puckSpeed_q <- quantile(shots_stats_goal.df$puckSpeed, probs = c(1/3, 2/3))
shooterSpeed_q <- quantile(shots_stats_goal.df$shooterSpeed, probs = c(1/3, 2/3))
goalieDist_q <- quantile(shots_stats_goal.df$goalieDist, probs = c(1/3, 2/3))
goalieAngle_q <- quantile(shots_stats_goal.df$goalieAngle, probs = c(1/3, 2/3))
posTime_q <- quantile(shots_stats_goal.df$posTime, probs = c(1/3, 2/3))
defDist_q <- quantile(shots_stats_goal.df$defDist, probs = c(1/3, 2/3))
defAngle_q <- quantile(shots_stats_goal.df$defAngle, probs = c(1/3, 2/3))

# Instead of dividing the data into thirds based on data, I manually selected the range so that we use
# puckDist_q <- c(319,510)
# puckAngle_q <- c(62, 140)
# puckSpeed_q <- c(35, 46)
# shooterSpeed_q <- c(12, 18)
# goalieDist_q <- c(55, 73)
# goalieAngle_q <- c(53, 95)
# posTime_q <- c(15,38)
# defDist_q <- c(109, 188)
# defAngle_q <- c(54, 140)

# Create a categorical variable for each continoue variables into 3 numeric values
# 0 for low 1 for medium and 2 for high
shots_stats_goal.df <- shots_stats_goal.df %>%
  mutate(puckSpeedCategory = case_when(
    puckSpeed <= puckSpeed_q[1] ~ 0,
    puckSpeed <= puckSpeed_q[2] ~ 1,
    TRUE ~ 2
  ))
shots_stats_goal.df <- shots_stats_goal.df %>%
  mutate(puckAngleCategory = case_when(
    puckAngle <= puckAngle_q[1] ~ 0,
    puckAngle <= puckAngle_q[2] ~ 1,
    TRUE ~ 2
  ))
shots_stats_goal.df <- shots_stats_goal.df %>%
  mutate(puckDistCategory = case_when(
    puckDist <= puckDist_q[1] ~ 0,
    puckDist <= puckDist_q[2] ~ 1,
    TRUE ~ 2
  ))
shots_stats_goal.df <- shots_stats_goal.df %>%
  mutate(posTimeCategory = case_when(
    posTime <= posTime_q[1] ~ 0,
    posTime <= posTime_q[2] ~ 1,
    TRUE ~ 2
  ))
shots_stats_goal.df <- shots_stats_goal.df %>%
  mutate(goalieDistCategory = case_when(
    goalieDist <= goalieDist_q[1] ~ 0,
    goalieDist <= goalieDist_q[2] ~ 1,
    TRUE ~ 2
  ))
shots_stats_goal.df <- shots_stats_goal.df %>%
```

```
  mutate(shooterSpeedCategory = case_when(
    shooterSpeed <= shooterSpeed_q[1] ~ 0,
    shooterSpeed <= shooterSpeed_q[2] ~ 1,
    TRUE ~ 2
  ))
shots_stats_goal.df <- shots_stats_goal.df %>%
  mutate(goalieAngleCategory = case_when(
    goalieAngle <= goalieAngle_q[1] ~ 0,
    goalieAngle <= goalieAngle_q[2] ~ 1,
    TRUE ~ 2
  ))
shots_stats_goal.df <- shots_stats_goal.df %>%
  mutate(defDistCategory = case_when(
    defDist <= defDist_q[1] ~ 0,
    defDist <= defDist_q[2] ~ 1,
    TRUE ~ 2
  ))
shots_stats_goal.df <- shots_stats_goal.df %>%
  mutate(defAngleCategory = case_when(
    defAngle <= defAngle_q[1] ~ 0,
    defAngle <= defAngle_q[2] ~ 1,
    TRUE ~ 2
  ))

# Creates a variable goal_binary that contains binary values 0 for save 1 for goal
shots_stats_goal.df <- shots_stats_goal.df %>%
  mutate(goal_binary = as.integer(outcomes.goal == 2))

# Saves shots_stats_goal.df into an Rds file
# saveRDS(shots_stats_goal.df, "categorized_shots_stats_goal.df.Rds")

# Split the data into training and testing sets (e.g., 80% train, 20% test)
#Create training set
set.seed(100)
shotstat_split <- initial_split(shots_stats_goal.df, prop = 0.8)
shotstat_train <- training(shotstat_split)
shotstat_test <- testing(shotstat_split)

# Dropping categorized variables for the regular model and dropping continous variables for categorized
Catshotstat_train <- shotstat_train %>%
    select(- puckDist, - puckAngle, - puckSpeed, - shooterSpeed, - goalieDist, - goalieAngle, - posTime
shotstat_train <- shotstat_train %>%
    select(- puckDistCategory, - puckAngleCategory, - puckSpeedCategory, - shooterSpeedCategory, - goali
Catshotstat_test <- shotstat_test %>%
    select(- puckDist, - puckAngle, - puckSpeed, - shooterSpeed, - goalieDist, - goalieAngle, - posTime
shotstat_test <- shotstat_test %>%
    select(- puckDistCategory, - puckAngleCategory, - puckSpeedCategory, - shooterSpeedCategory, - goali

# Create linear regression models for categorized data and continous data
CatLR <- glm(goal_binary ~. ,family = "binomial",data=Catshotstat_train)
```

```
## Warning: glm.fit: algorithm did not converge
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
LR <- glm(goal_binary ~. ,family = "binomial",data=shotstat_train)
```

```
## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

Analysis: Methods and results

Based on the linear regression models created above, I pulled out the summary of the two models, and created matrices for calculating the balanced accuracy.

```r
# Include all analysis code, clearly commented
# If not possible, screen shots are acceptable.
# If your contributions included things that are not done in an R-notebook,
#   (e.g. researching, writing, and coding in Python), you still need to do
#   this status notebook in R.  Describe what you did here and put any products
#   that you created in github. If you are writing online documents (e.g. overleaf
#   or google docs), you can include links to the documents in this notebook
#   instead of actual text.

# Load required library
library(knitr)

# Summary for each model
summary(CatLR)
```

```
##
## Call:
## glm(formula = goal_binary ~ ., family = "binomial", data = Catshotstat_train)
##
## Deviance Residuals:
##         Min           1Q       Median           3Q          Max
## -4.814e-04  -2.000e-08  -2.000e-08  -2.000e-08   4.388e-04
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)            -811.43   39482.29  -0.021    0.984
## rightHanded            -177.81    9896.96  -0.018    0.986
## puckSpeedCategory       582.51   28067.00   0.021    0.983
## puckAngleCategory       159.72    7856.23   0.020    0.984
## puckDistCategory       -983.69   47700.56  -0.021    0.984
## posTimeCategory         -51.30    4482.35  -0.011    0.991
## goalieDistCategory       73.28    4086.55   0.018    0.986
## shooterSpeedCategory   -434.33   21103.91  -0.021    0.984
## goalieAngleCategory     195.69    9959.47   0.020    0.984
## defDistCategory        -451.58   21836.24  -0.021    0.984
## defAngleCategory        437.19   21048.51   0.021    0.983
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 8.1237e+01  on 167  degrees of freedom
## Residual deviance: 1.3186e-06  on 157  degrees of freedom
## AIC: 22
##
## Number of Fisher Scoring iterations: 25
```

```r
summary(LR)
```

```
##
## Call:
## glm(formula = goal_binary ~ ., family = "binomial", data = shotstat_train)
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -6.537e-05  -2.100e-08  -2.100e-08  -2.100e-08   1.005e-04
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -204.1893 72849.7026  -0.003    0.998
## puckDist        -0.6791   174.6555  -0.004    0.997
## puckAngle        0.6651   293.7816   0.002    0.998
## puckSpeed        2.7059   750.3934   0.004    0.997
## shooterSpeed     0.2733  2166.4556   0.000    1.000
## goalieDist       1.4945   438.8974   0.003    0.997
## goalieAngle      0.7294   247.0782   0.003    0.998
## posTime         -0.7086  1022.6284  -0.001    0.999
## rightHanded    -63.9866 26313.0563  -0.002    0.998
## defDist         -0.3066   246.7929  -0.001    0.999
## defAngle         0.6377   224.5187   0.003    0.998
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 8.1237e+01  on 167  degrees of freedom
## Residual deviance: 3.9098e-08  on 157  degrees of freedom
## AIC: 22
##
## Number of Fisher Scoring iterations: 25
```

```r
# Making predictions using test
catpredictions <- predict(CatLR, newdata = Catshotstat_test)
predictions <- predict(LR, newdata = shotstat_test)

# Converting into a matrix for categorized data
catcm <- as.matrix(table(Actual = shotstat_test$goal_binary, Predicted = catpredictions>0.5))
kable(catcm)
```

	FALSE	TRUE
0	33	2
1	4	3

```r
# Balanced accuracy test for categorized data
balancedAccuracyTestCat<-(catcm[1,1]/(catcm[1,1]+catcm[1,2]) + catcm[2,2]/(catcm[2,1]+catcm[2,2]))/2
balancedAccuracyTestCat
```

```
## [1] 0.6857143
```

```r
# Converting into a matrix for continous data
cm <- as.matrix(table(Actual = shotstat_test$goal_binary, Predicted = predictions>0.5))
kable(cm)
```

	FALSE	TRUE
0	35	0
1	4	3

```
# Balanced accuracy test for continous data
balancedAccuracyTest<-(cm[1,1]/(cm[1,1]+cm[1,2]) + cm[2,2]/(cm[2,1]+cm[2,2]))/2
balancedAccuracyTest
```

```
## [1] 0.7142857
```

Discussion of results

When I used manually selected quantiles, the results were the same, and this is very likely due to the small sample size. But since we are interested in the comparison of the two models, I used quantiles drawn specifically from shots_stats_goal.df, and this showed the difference between the two models. The model for continous data had a better accuracy than the model for categorical data, and I believe this is due to the fact that categorical data estimates the likelyhood in "chunks" (every increment in categorical data is a lot greater than one increment in continous data).

One interesting observation from the coefficients of variables is that the some signs of continous and categorical variables for the same variable are not the same. For example, the shooterSpeed for categorical variable is negative, but for continous data, it is positive. This implies that a high shooter speed does not necessarily lead to a higher chances of goal because high speed category for shooter speed has a low chances of making a goal. But for most variables, the signs and the relative size of coefficients match i.e. large coefficient in continous variables means large coefficient in categorical variables with matching signs.

Analysis: How well do clusters match?

Question being asked

Using categorical and continous data, we can make clusters for each one. And using these clusters, I wanted to analyze how well they match.

Data Preparation

I created two types of clusters: one using continous and the other using categorical variables. I dropped some non-numeric variables so that the data can be clustered using kmeans. Then I took vectors of two models to only have cluster assignments; then created confusion matrix using those vectors. The number of clusters are determined by the elbow test in the next analysis, which gave 4 for continous model and 6 for the categorical model.

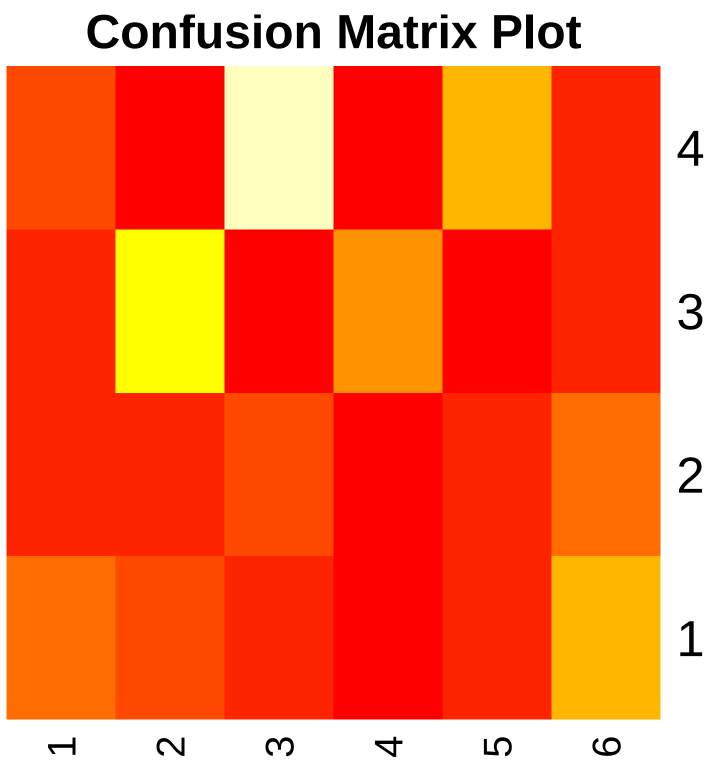
```
# Include all data processing code (if necessary), clearly commented
#
shots <- readRDS('shots_stats_goal.df.Rds')
catshots <- readRDS('categorized_shots_stats_goal.df.Rds')

# Dropping some variables necessary so that the data can be used for kmeans cluster
catshots <- catshots %>%
    select(- puckDist, - puckAngle, - puckSpeed, - shooterSpeed, - goalieDist, - goalieAngle, - posTime
org_shots <- shots %>%
    select(- closestDef, - shotOutcome, - outcomes.goal)

# Making models using k means
org_model <- kmeans(org_shots, centers = 4)
cat_model <- kmeans(catshots, centers = 6)
```

```r
# Get cluster assignments
org_cluster <- org_model$cluster
cat_cluster <- cat_model$cluster

# Create a confusion matrix
conf_mat <- table(org_cluster, cat_cluster)

# Calculate means for each cluster for continuous variables
means_org <- aggregate(catshots, by = list(org_cluster), FUN = mean)

# Calculate means for each cluster for categorical variables
means_cat <- aggregate(org_shots, by = list(cat_cluster), FUN = mean)
```

Analysis: Methods and Results

All the necessary data preparation had been done above, so I only needed to plot the confusion matrix using heatmap function. I also tried to add a legend to the confusion matrix, but the heatmap function does not allow adding legend. However, the high frequency of observations are represented by darker color, in this case this is represented by red, and the low frequency of observations are closer to white, which are represented by light yellow in the matrix below.

```r
# Include all analysis code, clearly commented
# If not possible, screen shots are acceptable.
# If your contributions included things that are not done in an R-notebook,
#   (e.g. researching, writing, and coding in Python), you still need to do
#   this status notebook in R.  Describe what you did here and put any products
#   that you created in github. If you are writing online documents (e.g. overleaf
#   or google docs), you can include links to the documents in this notebook
#   instead of actual text.

# Plot the confusion matrix as a heatmap
heatmap(conf_mat, Colv = NA, Rowv = NA, col = heat.colors(10), scale = "none",
        main = "Confusion Matrix Plot")
```




```
# Print means for each cluster for continous variables
print(means_org)
```

```
##   Group.1 NumOffense NumDefense rightHanded puckSpeedCategory puckAngleCategory
## 1       1 0.38461538   2.000000   0.6153846         1.0000000         1.2307692
## 2       2 0.50000000   1.611111   0.5000000         1.2777778         1.0555556
## 3       3 0.07692308   1.384615   0.3846154         0.8461538         0.7692308
## 4       4 0.97142857   2.828571   0.4000000         0.9714286         0.9714286
##   puckDistCategory posTimeCategory goalieDistCategory shooterSpeedCategory
## 1        0.6538462       1.0000000          0.9615385            1.2307692
## 2        1.5000000       0.9444444          1.2222222            0.8333333
## 3        0.0000000       0.6538462          0.3461538            1.2307692
## 4        1.7428571       1.2285714          1.4000000            0.7428571
##   goalieAngleCategory defDistCategory defAngleCategory goal_binary
## 1           1.2692308       0.9230769        0.9230769  0.07692308
## 2           1.0000000       2.0000000        1.0000000  0.00000000
## 3           0.6923077       0.4230769        1.0769231  0.26923077
## 4           1.0285714       0.9714286        1.0000000  0.00000000
```

```
# Print means for each cluster for categorized variables
print(means_cat)
```

```
##   Group.1 puckDist puckAngle puckSpeed shooterSpeed goalieDist goalieAngle
## 1       1 421.0526  62.90532  38.16093     22.90159   94.03130    55.93682
## 2       2 222.5990  74.40173  34.94147     12.39629   48.13140    66.04039
## 3       3 548.0565  66.77332  41.68051     11.72421   81.79129    60.14884
## 4       4 181.4884  61.90396  46.44958     21.96774   44.92136    27.54925
## 5       5 521.4789 109.06354  28.67343     12.16912   72.43388   102.12399
## 6       6 414.8612 128.59672  57.08257     19.09483   66.41333   128.77321
##    posTime NumOffense NumDefense rightHanded  defDist  defAngle
## 1 47.21429  0.1428571   1.857143   0.6428571 155.2432 143.92956
```

```
## 2 12.25000  0.2000000   1.350000   0.2000000 158.9548 120.48689
## 3 28.40000  1.2800000   2.920000   0.4800000 222.0535 138.00127
## 4 35.80000  0.0000000   1.300000   0.3000000 105.8849 109.38731
## 5 32.86667  0.9333333   2.733333   0.4666667 189.1327  25.79820
## 6 28.19048  0.1428571   1.714286   0.6666667 232.5356  48.11776
```

Discussion of results

In the resulting matrix, the row clusters are for the continuous variables and the column clusters are the categorical variables. Overall, we can see that the clusters from two models match very well because almost half of the cells are dark red, meaning there was high frequency of observations of two clusters. And there was one groups of clusters that did not match well, cluster 4 from continous 3 from categorical.

Analysis: Quality of clusters

Question being asked

We have analyzed how well the clusters match above. Now we want to analyze the quality of clusters using the elbow test. I should have done this before the analysis 2, but I just added this after talking to professor Bennet today.

Data Preparation

Catshots and org_shots are data used in the second analysis. These dataframes are modified so that they can be used for kmeans function. I took the number of clusters from 1 to 10 to compare the within cluster sum of squares by number of clusters. Then the wss_values have been saved to be used for plotting.

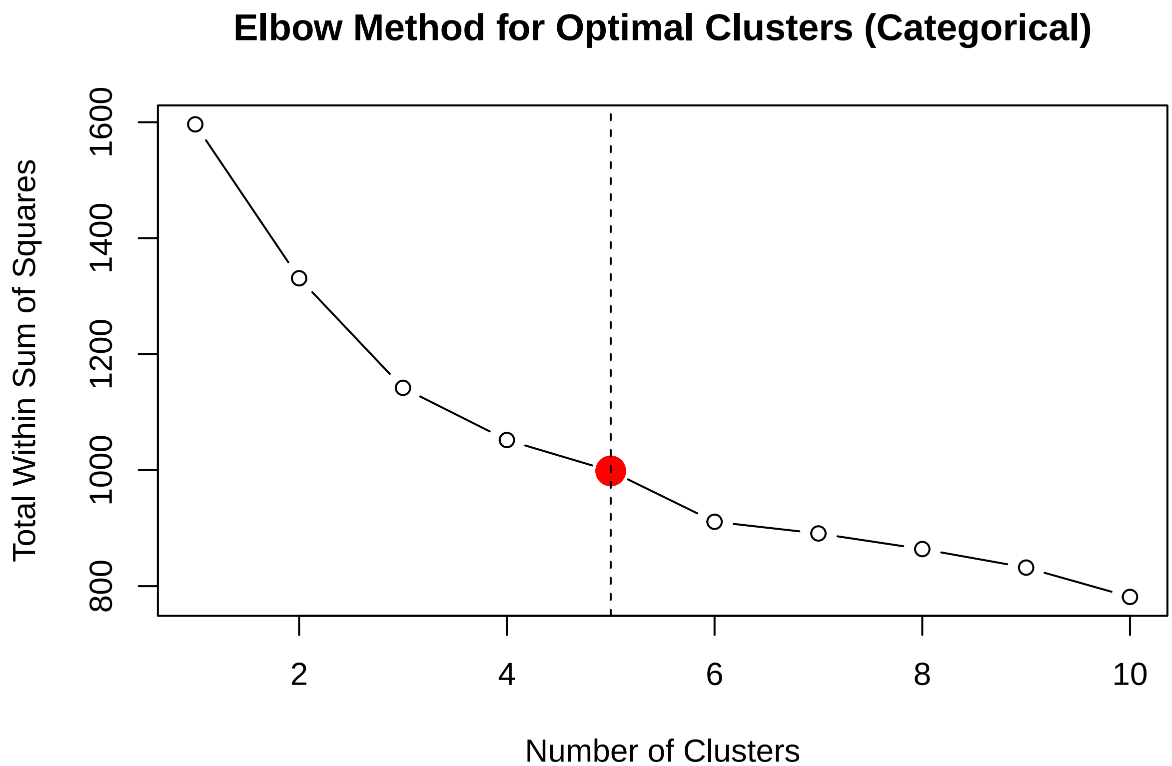
```r
# Include all data processing code (if necessary), clearly commented

# Compute the total within-cluster sum of squares for different numbers of clusters
cat_wss_values <- c()
for (i in 1:10) {
  cat_elbow <- kmeans(catshots, centers = i)
  cat_wss_values <- c(cat_wss_values, cat_elbow$tot.withinss)
}

org_wss_values <- c()
for (i in 1:10) {
  org_elbow <- kmeans(org_shots, centers = i)
  org_wss_values <- c(org_wss_values, org_elbow$tot.withinss)
}
```

Analysis methods used

Using the wss_values calculated above, I identified the elbow point by comparing the value from i-1 and i+1. Then they have been plotted, and for a better visualization effect, I also have highlighted the elbow points.

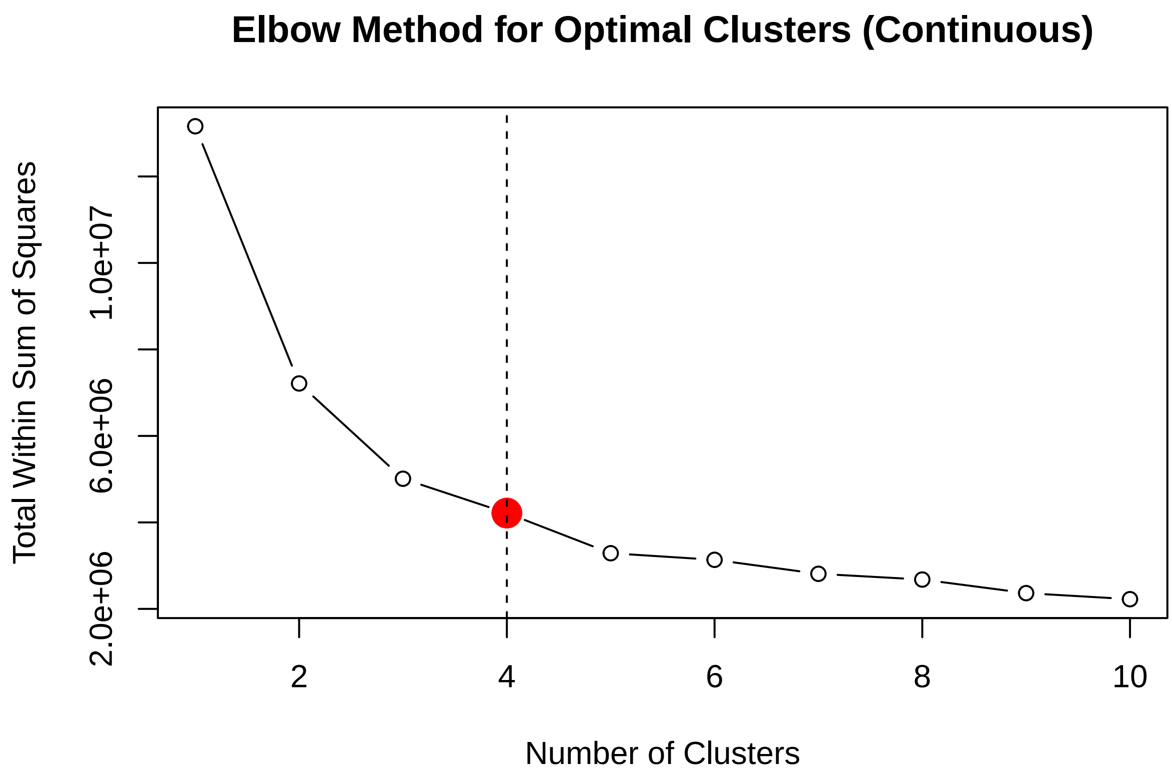
```r
# Include all analysis code, clearly commented
# If not possible, screen shots are acceptable.
# If your contributions included things that are not done in an R-notebook,
#   (e.g. researching, writing, and coding in Python), you still need to do
#   this status notebook in R.  Describe what you did here and put any products
#   that you created in github. If you are writing online documents (e.g. overleaf
#   or google docs), you can include links to the documents in this notebook
#   instead of actual text.
```

```r
# This is for categorical model
# Plot the Elbow plot to visualize the within-cluster sum of squares by number of clusters
plot(1:10, cat_wss_values, type = "b", xlab = "Number of Clusters", ylab = "Total Within Sum of Squares
     main = "Elbow Method for Optimal Clusters (Categorical)")

# Identify the elbow point
cat_elbow_k <- 1
for (i in 2:(length(cat_wss_values) - 1)) {
  if ((cat_wss_values[i] - cat_wss_values[i - 1]) > (cat_wss_values[i + 1] - cat_wss_values[i])) {
    cat_elbow_k <- i
    break
  }
}

# Highlight the elbow point in the plot
points(cat_elbow_k, cat_wss_values[cat_elbow_k], col = "red", cex = 2, pch = 19)
abline(v = cat_elbow_k, lty = 2)
```




```r
# This is for the continous model
# Plot the Elbow plot to visualize the within-cluster sum of squares by number of clusters
plot(1:10, org_wss_values, type = "b", xlab = "Number of Clusters", ylab = "Total Within Sum of Squares
     main = "Elbow Method for Optimal Clusters (Continuous)")

# Identify the elbow point
org_elbow_k <- 1
for (i in 2:(length(org_wss_values) - 1)) {
  if ((org_wss_values[i] - org_wss_values[i - 1]) > (org_wss_values[i + 1] - org_wss_values[i])) {
    org_elbow_k <- i
    break
```

```
  }
}

# Highlight the elbow point in the plot
points(org_elbow_k, org_wss_values[org_elbow_k], col = "red", cex = 2, pch = 19)
abline(v = org_elbow_k, lty = 2)
```



Discussion of results

Based on the results, the ideal number of clusters for the continous model is 4, and the ideal number of clusters for the categorical model is 5.

Summary and next steps

I will discuss the results with professor on Wednesday and decide what to work on.