ISYE 6501: Homework 10

2025-03-24

Question 14.1

The breast cancer data set breast-cancer-wisconsin.data.txt from http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/ (description at http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29) has missing values. Use the mean/mode imputation method to impute values for the missing data. Use regression to impute values for the missing data. Use regression with perturbation to impute values for the missing data. (Optional) Compare the results and quality of classification models (e.g., SVM, KNN) build using

(1) the data sets from questions 1,2,3;

(2) the data that remains after data points with missing values are removed; and

(3) the data set when a binary variable is introduced to indicate missing values.

```
# initialize libraries
library(kknn)
# Set seed for reproducibility
set.seed(123)
# To begin, we import the data and save as a dataframe for kknn function
data <- as.data.frame(read.table('breast-cancer-wisconsin.data.txt',stringsAsFactors = FALSE, header = F

summary(data)
```

```
##        V1                  V2               V3              V4
##  Min.   :   61634   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
##  1st Qu.:  870688   1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 1.000
##  Median : 1171710   Median : 4.000   Median : 1.000   Median : 1.000
##  Mean   : 1071704   Mean   : 4.418   Mean   : 3.134   Mean   : 3.207
##  3rd Qu.: 1238298   3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.: 5.000
##  Max.   :13454352   Max.   :10.000   Max.   :10.000   Max.   :10.000
##        V5               V6              V7                V8
##  Min.   : 1.000   Min.   : 1.000   Length:699         Min.   : 1.000
##  1st Qu.: 1.000   1st Qu.: 2.000   Class :character   1st Qu.: 2.000
##  Median : 1.000   Median : 2.000   Mode  :character   Median : 3.000
##  Mean   : 2.807   Mean   : 3.216                      Mean   : 3.438
##  3rd Qu.: 4.000   3rd Qu.: 4.000                      3rd Qu.: 5.000
##  Max.   :10.000   Max.   :10.000                      Max.   :10.000
##        V9              V10              V11
##  Min.   : 1.000   Min.   : 1.000   Min.   :2.00
##  1st Qu.: 1.000   1st Qu.: 1.000   1st Qu.:2.00
##  Median : 1.000   Median : 1.000   Median :2.00
##  Mean   : 2.867   Mean   : 1.589   Mean   :2.69
##  3rd Qu.: 4.000   3rd Qu.: 1.000   3rd Qu.:4.00
##  Max.   :10.000   Max.   :10.000   Max.   :4.00
```

```
error_list <- which(data$V7== "?")
error_list #list of indices that have a ?
```

```
##  [1]  24  41 140 146 159 165 236 250 276 293 295 298 316 322 412 618
```

```
length(error_list)/nrow(data)
```

```
## [1] 0.02288984
```

```
data_good <- data[-error_list,]
data_bad <- data[error_list,]
table(data$V11)
```

```
##
##   2   4
## 458 241
```

```
table(data_good$V11)
```

```
##
##   2   4
## 444 239
```

```
table(data_bad$V11)
```

```
##
##  2  4
## 14  2
```

```
# mean imputation
data_mean <- data
# calculate mean
V7_mean <- mean(as.numeric(data_good$V7))
V7_mean
```

```
## [1] 3.544656
```

```
# replace errors w mean
data_mean[error_list,]$V7 <- V7_mean
data_mean$V7 = as.numeric(as.character(data_mean$V7))
summary(data_mean)
```

```
##        V1                  V2               V3               V4
##  Min.   :   61634   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
##  1st Qu.:  870688   1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 1.000
##  Median : 1171710   Median : 4.000   Median : 1.000   Median : 1.000
##  Mean   : 1071704   Mean   : 4.418   Mean   : 3.134   Mean   : 3.207
##  3rd Qu.: 1238298   3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.: 5.000
##  Max.   :13454352   Max.   :10.000   Max.   :10.000   Max.   :10.000
##        V5               V6               V7               V8
##  Min.   : 1.000   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
##  1st Qu.: 1.000   1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 2.000
##  Median : 1.000   Median : 2.000   Median : 1.000   Median : 3.000
##  Mean   : 2.807   Mean   : 3.216   Mean   : 3.545   Mean   : 3.438
##  3rd Qu.: 4.000   3rd Qu.: 4.000   3rd Qu.: 5.000   3rd Qu.: 5.000
##  Max.   :10.000   Max.   :10.000   Max.   :10.000   Max.   :10.000
##        V9              V10              V11
##  Min.   : 1.000   Min.   : 1.000   Min.   :2.00
```

```
##  1st Qu.: 1.000   1st Qu.: 1.000   1st Qu.:2.00
##  Median : 1.000   Median : 1.000   Median :2.00
##  Mean   : 2.867   Mean   : 1.589   Mean   :2.69
##  3rd Qu.: 4.000   3rd Qu.: 1.000   3rd Qu.:4.00
##  Max.   :10.000   Max.   :10.000   Max.   :4.00
```

```
# regression imputation
data_regression <- data
# convert V7 in our good data set to numbers so we can use lm
data_good$V7 = as.numeric(as.character(data_good$V7))
# perform regression, make sure not to use column 1 due it being an ID (numbers dont have meaning)
# fit multiple linear regression model
impute_model <- lm(V7~V2+V3+V4+V5+V6+V8+V9+V10+V11, data=data_good)
# view model summary
summary(impute_model)
```

```
##
## Call:
## lm(formula = V7 ~ V2 + V3 + V4 + V5 + V6 + V8 + V9 + V10 + V11,
##     data = data_good)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.6030 -0.4262 -0.2194  0.8696  8.6294
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.25273    0.30981 -13.727  < 2e-16 ***
## V2           0.01853    0.03962   0.468  0.64019
## V3          -0.16215    0.06731  -2.409  0.01627 *
## V4           0.18437    0.06551   2.815  0.00503 **
## V5           0.22093    0.04125   5.356 1.17e-07 ***
## V6           0.01922    0.05523   0.348  0.72790
## V8           0.15128    0.05330   2.839  0.00467 **
## V9          -0.08738    0.03969  -2.201  0.02804 *
## V10         -0.06300    0.05218  -1.207  0.22770
## V11          2.50988    0.17811  14.091  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2 on 673 degrees of freedom
## Multiple R-squared:  0.7027, Adjusted R-squared:  0.6987
## F-statistic: 176.7 on 9 and 673 DF,  p-value: < 2.2e-16
```

```
impute_model_v2 <- lm(V7~V3+V4+V5+V8+V9+V11, data=data_good)
summary(impute_model_v2)
```

```
##
## Call:
## lm(formula = V7 ~ V3 + V4 + V5 + V8 + V9 + V11, data = data_good)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.9685 -0.4130 -0.2560  0.8506  8.5870
##
```

```
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.28748    0.30423 -14.093  < 2e-16 ***
## V3          -0.16300    0.06499  -2.508  0.01238 *
## V4           0.18770    0.06497   2.889  0.00399 **
## V5           0.21369    0.04068   5.253 2.01e-07 ***
## V8           0.15697    0.05301   2.961  0.00317 **
## V9          -0.09283    0.03907  -2.376  0.01779 *
## V11          2.54202    0.16391  15.509  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.998 on 676 degrees of freedom
## Multiple R-squared:  0.702,  Adjusted R-squared:  0.6993
## F-statistic: 265.4 on 6 and 676 DF,  p-value: < 2.2e-16
```

```
# predict values of V7
round(predict(impute_model_v2, data_bad))
```

```
##  24  41 140 146 159 165 236 250 276 293 295 298 316 322 412 618
##   7   3   1   2   1   1   2   1   2   6   1   1   2   1   1   1
```

```
predicted_V7 <- round(predict(impute_model_v2, data_bad))
# replace errors w regression
data_regression[error_list,]$V7 <- predicted_V7
data_regression$V7 = as.numeric(as.character(data_regression$V7))
summary(data_regression)
```

```
##        V1                 V2               V3               V4
##  Min.   :   61634   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
##  1st Qu.:  870688   1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 1.000
##  Median : 1171710   Median : 4.000   Median : 1.000   Median : 1.000
##  Mean   : 1071704   Mean   : 4.418   Mean   : 3.134   Mean   : 3.207
##  3rd Qu.: 1238298   3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.: 5.000
##  Max.   :13454352   Max.   :10.000   Max.   :10.000   Max.   :10.000
##        V5               V6               V7               V8
##  Min.   : 1.000   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
##  1st Qu.: 1.000   1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 2.000
##  Median : 1.000   Median : 2.000   Median : 1.000   Median : 3.000
##  Mean   : 2.807   Mean   : 3.216   Mean   : 3.511   Mean   : 3.438
##  3rd Qu.: 4.000   3rd Qu.: 4.000   3rd Qu.: 6.000   3rd Qu.: 5.000
##  Max.   :10.000   Max.   :10.000   Max.   :10.000   Max.   :10.000
##        V9              V10              V11
##  Min.   : 1.000   Min.   : 1.000   Min.   :2.00
##  1st Qu.: 1.000   1st Qu.: 1.000   1st Qu.:2.00
##  Median : 1.000   Median : 1.000   Median :2.00
##  Mean   : 2.867   Mean   : 1.589   Mean   :2.69
##  3rd Qu.: 4.000   3rd Qu.: 1.000   3rd Qu.:4.00
##  Max.   :10.000   Max.   :10.000   Max.   :4.00
```

```
# perturbation imputation
data_perturbation <- data
V7_mean #mean
```

```
## [1] 3.544656
```

```r
V7_sd <- sd(as.numeric(data_good$V7))
V7_sd
```

```
## [1] 3.643857
```

```r
# create normal distribution
perturb <- rnorm(length(error_list), mean = V7_mean, sd = V7_sd)
perturb
```

```
##  [1]  1.502363  2.705922  9.224366  3.801578  4.015762  9.794108  5.224169
##  [8] -1.065047  1.041862  1.920727  8.005035  4.855766  5.005010  3.947968
## [15]  1.519250 10.055912
```

```r
# bringing back the model so we can round at the end
predicted_unrounded <- predict(impute_model_v2, data_bad)
round(predicted_unrounded + perturb)
```

```
##  24  41 140 146 159 165 236 250 276 293 295 298 316 322 412 618
##   9   6  10   5   5  11   7   0   3   8   9   6   7   5   3  11
```

```r
# must limit to between 1-10 due to data set specifications
perturbed_V7 <- round(predicted_unrounded + perturb)
min(perturbed_V7) # 0
```

```
## [1] 0
```

```r
max(perturbed_V7) # 11
```

```
## [1] 11
```

```r
perturbed_V7[perturbed_V7 > 10] <- 10
perturbed_V7[perturbed_V7 < 1] <- 1
perturbed_V7
```

```
##  24  41 140 146 159 165 236 250 276 293 295 298 316 322 412 618
##   9   6  10   5   5  10   7   1   3   8   9   6   7   5   3  10
```

```r
data_perturbation[error_list,]$V7 <- perturbed_V7
data_perturbation$V7 = as.numeric(as.character(data_perturbation$V7))
summary(data_regression)
```

```
##        V1                 V2              V3              V4
##  Min.   :   61634   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
##  1st Qu.:  870688   1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 1.000
##  Median : 1171710   Median : 4.000   Median : 1.000   Median : 1.000
##  Mean   : 1071704   Mean   : 4.418   Mean   : 3.134   Mean   : 3.207
##  3rd Qu.: 1238298   3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.: 5.000
##  Max.   :13454352   Max.   :10.000   Max.   :10.000   Max.   :10.000
##        V5              V6              V7              V8
##  Min.   : 1.000   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
##  1st Qu.: 1.000   1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 2.000
##  Median : 1.000   Median : 2.000   Median : 1.000   Median : 3.000
##  Mean   : 2.807   Mean   : 3.216   Mean   : 3.511   Mean   : 3.438
##  3rd Qu.: 4.000   3rd Qu.: 4.000   3rd Qu.: 6.000   3rd Qu.: 5.000
##  Max.   :10.000   Max.   :10.000   Max.   :10.000   Max.   :10.000
##        V9              V10             V11
##  Min.   : 1.000   Min.   : 1.000   Min.   :2.00
##  1st Qu.: 1.000   1st Qu.: 1.000   1st Qu.:2.00
```

```
##  Median : 1.000   Median : 1.000   Median :2.00
##  Mean   : 2.867   Mean   : 1.589   Mean   :2.69
##  3rd Qu.: 4.000   3rd Qu.: 1.000   3rd Qu.:4.00
##  Max.   :10.000   Max.   :10.000   Max.   :4.00
```

```
#optional portion
data_mean$V11[data_mean$V11== 2] <- 0
data_mean$V11[data_mean$V11== 4] <- 1
data_regression$V11[data_regression$V11== 2] <- 0
data_regression$V11[data_regression$V11== 4] <- 1
data_perturbation$V11[data_perturbation$V11== 2] <- 0
data_perturbation$V11[data_perturbation$V11== 4] <- 1
datasets <- list(data_mean, data_regression, data_perturbation)
results <- list()
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:kknn':
##
##     contr.dummy
```

```
train_control <- trainControl(method = "cv", number = 5)
# define a range of k values to test
k_values <- data.frame(k = seq(1, 10, by = 1))
# Convert V11 to a factor in each dataset
for (i in 1:length(datasets)) {
  datasets[[i]]$V11 <- as.factor(datasets[[i]]$V11)
}
# Loop through each dataset
for (i in 1:length(datasets)) {
  data_test <- datasets[[i]]
  # define variables
  X <- data_test[, 2:10] # V2 to V10
  y <- data_test$V11 # V11 as the binary response
  # Combine X and y into a new data frame
  dataset <- cbind(X, V11 = y)
  # train model
  knn_model <- train(V11~
                       ., data = dataset,
                     method = "knn",
                     trControl = train_control,
                     tuneGrid = k_values)
  # store the best k and its accuracy
  best_k <- knn_model$bestTune$k
  best_accuracy <- max(knn_model$results$Accuracy)
  results[[paste0("data", i)]] <- list(best_k = best_k, best_accuracy = best_accuracy)
}
print("Best k values and accuracies for each dataset:")
```

```
## [1] "Best k values and accuracies for each dataset:"
```

```r
print(results)
```

```
## $data1
## $data1$best_k
## [1] 7
##
## $data1$best_accuracy
## [1] 0.9713562
##
##
## $data2
## $data2$best_k
## [1] 6
##
## $data2$best_accuracy
## [1] 0.9713562
##
##
## $data3
## $data3$best_k
## [1] 4
##
## $data3$best_accuracy
## [1] 0.9728052
```

Handling Missing Data in the Breast Cancer Dataset

Introduction

The breast cancer dataset includes missing values in column **V7**, which can affect model accuracy and introduce bias. This analysis examines the missing data, evaluates its potential bias, and compares different imputation methods to handle it effectively.

Identifying Missing Data

When importing the dataset into R, **V7** was recognized as a character variable because it contained "?" symbols representing missing values. The dataset has **699 observations**, with **16 missing values** in **V7**, making up **2.29%** of the data. Since this is below the common 5% threshold for serious concern, several imputation techniques can be considered without significant risk of distortion.

Is There Bias in the Missing Data?

To check for bias, we compared the class distribution (V11) between cases with and without missing values:

- Overall dataset: 458 benign (2) vs. 241 malignant (4)
- Without missing values: 444 benign vs. 239 malignant
- With missing values: 14 benign vs. 2 malignant

Since most missing values occur in benign cases, improperly handling them could skew classification results.

Imputation Methods

1. Mean Imputation

- The average V7 value in the complete dataset was 3.54.
- Missing values were replaced with this mean.

- While simple and easy to apply, this method reduces variability and may introduce bias by assigning the same value to all missing cases.

2. Regression-Based Imputation

- A multiple linear regression model was created using V7 as the dependent variable and other columns (V2-V6, V8-V11) as predictors.
- Some variables (V2, V6, V10) were statistically insignificant and removed in a refined model.
- The final model achieved an **R-squared of 0.702**, meaning it explained about 70% of the variation in **V7**.
- Missing values were predicted and rounded to whole numbers.
- This approach leverages existing data patterns but assumes missing values follow the same trend as the observed data.

3. Regression with Perturbation

- This method builds on regression-based imputation but adds a small random variation to each predicted value.
- By introducing randomness, this technique prevents artificial uniformity and better maintains data variability.
- It offers a balance between accuracy and natural variation but requires careful tuning.

Comparison of Methods

Method	Advantages	Disadvantages
Mean Imputation	Simple, fast, preserves sample size	Reduces variance, introduces bias
Regression Imputation	Uses relationships in data for better predictions	Assumes missing data follows the same pattern as complete data
Regression + Perturbation	Balances accuracy with variability	More complex, requires tuning

Conclusion

While the missing data in **V7** is minimal, it is skewed toward benign cases. **Regression with perturbation** provides the most balanced approach, preserving both predictive accuracy and variability. However, the best imputation method depends on the analysis goals: mean imputation is a quick solution, while regression based methods are preferable for predictive modeling.

Question 15.1

Describe a situation or problem from your job, everyday life, current events, etc., for which optimization would be appropriate. What data would you need?

An example of optimization in everyday life is planning a workout schedule to maximize fitness results while balancing recovery and time constraints.

Problem:

I want to optimize my gym routine to achieve my fitness goals (e.g., weight loss, muscle gain, or endurance improvement) while ensuring proper rest, avoiding overtraining, and fitting workouts into their busy schedule.

Data Needed:

1. **Personal Data:**

- Fitness goals (e.g., strength, hypertrophy, endurance)
- Current fitness level and experience
- Available workout days per week
- Time available per workout
- Recovery rate (how many days muscles need to recover)

2. **Exercise Data:**
   - Muscle groups worked per exercise
   - Optimal frequency and intensity for each muscle group
   - Average calorie burn per workout
   - Exercise efficiency (e.g., compound movements vs. isolation exercises)
3. **Scheduling Constraints:**
   - Rest day requirements
   - Gym peak hours (to avoid overcrowding)
   - Other commitments (e.g., work, school, social life)

Optimization Goal:

Create the most effective weekly workout schedule that:
- **Maximizes** muscle gain or fat loss
- **Minimizes** recovery time issues (avoiding overtraining)
- **Balances** gym sessions with other commitments
- **Avoids** inefficient workouts