Midterm II

2024-03-03

Your name:

Questions

Q1 Logistic Regression

We're interested in estimating the probability of developing hypertension based on systolic blood pressure using logistic regression. The logistic regression model has been trained, and the coefficients are $\beta_0 = -3.78$ for the intercept and $\beta_1 = 0.021$ for systolic blood pressure.

a. Calculate the odds for a systolic blood pressure of 135 mmHg. (5 points)

```
## [1] 0.3886796
```

b. Convert the odds in part a to the probability of developing hypertension. (5 points)

```
## [1] 0.2798915
```

Q2 k-Nearest Neighbor

Consider a binary classification problem where we use a k-Nearest Neighbors (k-NN) algorithm. We have a confusion matrix as above, which shows the classification performance:

a. (5 points)

Given the confusion matrix, calculate the model's precision and recall for the "Positive" class. What does this imply about the model's performance in identifying positive cases?

In terms of the model's performance in identifying positive cases:

The recall is relatively high, suggesting that the model is quite good at identifying most of the positive cases. The precision is slightly lower than the recall, indicating that when the model predicts a case as positive, there's a 20.8 % chance it's actually negative.

Table 1: Confusion Matrix

	Predicted Negative	Predicted Positive
Actual Negative	65	25
Actual Positive	15	95

These metrics imply that while the model is quite sensitive to detecting positive cases, it does sacrifice some accuracy, resulting in some false positives. This trade-off is common in predictive modeling and must be balanced against the costs of false positives and false negatives in the specific application context

b. (5 points)

Discuss the influence of the k parameter size on the occurrence of overfitting and underfitting within the k-Nearest Neighbors (k-NN) algorithm. How can adjusting k help in achieving a balance to optimize model performance?

The parameter k in k-Nearest Neighbors ($\mathrm{k-NN}$) determines the number of nearest neighbors to consider when making a classification decision. The choice of k can significantly affect the algorithm's tendency to overfit or underfit: - Overfitting: If k is too small, the model becomes highly sensitive to noise in the training data, considering only the closest points. This can lead to overfitting, where the model captures noise and irregularities in the training data and may not generalize well to new, unseen data. - Underfitting: Conversely, if k is too large, the model becomes overly generalized, which can smooth out the prediction too much and result in underfitting. In this case, the model may not capture sufficient nuances of the data distribution and may perform poorly even on the training data.

Balancing with k : - To achieve a balance and optimize model performance, k should be set to a value that provides a good generalization capability without capturing noise. This is typically done through crossvalidation, where different values of k are tested, and the one that yields the best performance on a validation set is chosen. The optimal k leads to a model that has good performance metrics on both training and validation datasets, indicating a balance between bias and variance.

Q3 Miscellaneous: Multiple Choice (5 points each)

a. In logistic regression, odds are defined as:

A) The ratio of the probability of the event occurring to the probability of it not occurring.
B) A direct measure of probability that an event occurs, identical in interpretation.
C) The logarithmic transformation of probabilities for easier computation.
D) The probability of an event occurring divided by the probability of all other events.

b. In the context of the k-Nearest Neighbors (k-NN) algorithm, consider how the choice of k impacts the model's bias and variance. Select the correct statement from the options below:

A) Increasing the value of k decreases both bias and variance, leading to a universally better model performance across different datasets.
B) Decreasing the value of k to 1 minimizes the model's variance while maximizing its bias, as the prediction is entirely based on the nearest neighbor.
C) Increasing the value of k generally increases the model's bias but reduces its variance, as the classification decision is based on a larger, more generalized set of neighbors.
D) A very large value of k (approaching the size of the training set) results in a model with low bias and high variance, as it becomes too sensitive to the noise in the training data.

c. When the K-Means clustering algorithm reaches a point where the assignment of observations to clusters remains constant across successive iterations, it indicates:

A) The algorithm has potentially reached a local minimum, but not necessarily the global optimum.
B) The algorithm has converged, indicating stability in cluster assignment and optimal clustering has been achieved.
C) The algorithm requires re-initialization as it has likely converged to a suboptimal solution.
D) The clustering process is incomplete and requires more iterations for accurate cluster assignment.

d. The utilization of cross-validation techniques in machine learning models:

A) Exclusively mitigates overfitting by training the model on multiple subsets of the data.
B) Can effectively eliminate the need for a separate test set by using the training data for validation.
C) Helps in mitigating both overfitting and underfitting by validating the model's performance on different subsets of the dataset.
D) Guarantees improvement in model performance on unseen data by optimizing hyperparameters.

e. In logistic regression, changing the decision threshold (default is 0.5) affects the model's sensitivity and specificity. Which of the following statements is true when increasing the decision threshold above 0.5 ?

A) Sensitivity increases while specificity decreases.
B) Both sensitivity and specificity increase.
C) Sensitivity decreases while specificity increases.
D) Both sensitivity and specificity decrease.

Q4 Loops and functions

Consider the following data frame `df_waste` which represents the waste generation (in tonnes) of five types of waste materials: Plastic, Metal, Glass, Paper, and Organic in a city over 12 months in a city.

```
glimpse(df_waste)
## Rows: 12
## Columns: 6
## $ month   <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
## $ plastic <dbl> 150, 120, 130, 200, 210, 180, 190, 220, 250, 260, 270, 280
## $ metal   <dbl> 100, 110, 120, 130, 140, 150, 130, 120, 110, 115, 125, 135
## $ glass   <dbl> 200, 190, 180, 170, 160, 150, 160, 170, 180, 185, 190, 195
## $ paper   <dbl> 90, 85, 95, 100, 105, 110, 115, 120, 125, 130, 135, 140
## $ organic <dbl> 300, 310, 320, 330, 340, 350, 360, 370, 380, 390, 400, 410
```

a (10 points)

Given `df_waste` write a loop to calculate the monthly average waste generation for each type of material. Your resulting output should be a data frame with each row representing the average waste generation for a month and each column representing a type of material.

```
# Initialize an empty data frame to store monthly averages
monthly_averages <- data.frame(matrix(ncol = ncol(df_waste)-1, nrow = 1))
names(monthly_averages) <- names(df_waste)[-1]

for (i in 1:(ncol(df_waste)-1)) {
  monthly_averages[1, i] <- mean(df_waste[, i+1])
}

monthly_averages
##   plastic  metal glass paper organic
## 1     205 123.75 177.5 112.5     355
```

b (10 points)

Given the dataset `df_waste` representing monthly waste generation (in tonnes) for various materials and the vector `seasons` indicating the corresponding season for each month, calculate the average waste generation for each type of material across the four seasons: Winter, Spring, Summer, and Autumn. The seasons are defined as follows:

- Winter: December, January, February
- Spring: March, April, May
- Summer: June, July, August
- Autumn: September, October, November

The `seasons` vector provided is as follows:

```
seasons <- c("Winter", "Winter", "Spring", "Spring", "Spring", "Summer",
             "Summer", "Summer", "Autumn", "Autumn", "Autumn", "Winter")
```

Use `lapply` or `map` functions from R to compute the average waste generation for each material (`Plastic`, `Metal`, `Glass`, `Paper`, `Organic`) for each `season`. The expected output should be a structured data object (like a `list` or a `data frame`) where each entry or row corresponds to a season, showing the average waste generation for each material during that season.

```
df_waste$season <- rep(seasons, each = nrow(df_waste) / length(seasons))
```

```r
average_waste_by_season <- map_dfr(names(df_waste)[2:6], function(material) {
  df_waste %>%
    group_by(season) %>%
    summarize(Material = material,
              Average = mean(.data[[material]], na.rm = TRUE)) %>%
    ungroup()
})

average_waste_by_season
## # A tibble: 20 x 3
##    season Material Average
##    <chr>  <chr>      <dbl>
##  1 Autumn plastic     260
##  2 Spring plastic     180
##  3 Summer plastic     197.
##  4 Winter plastic     183.
##  5 Autumn metal       117.
##  6 Spring metal       130
##  7 Summer metal       133.
##  8 Winter metal       115
##  9 Autumn glass       185
## 10 Spring glass       170
## 11 Summer glass       160
## 12 Winter glass       195
## 13 Autumn paper       130
## 14 Spring paper       100
## 15 Summer paper       115
## 16 Winter paper       105
## 17 Autumn organic     390
## 18 Spring organic     330
## 19 Summer organic     360
## 20 Winter organic     340
```

Q5 String Manipulations

a. (5 points)

Given a vector of words, write a function named `find_vowel_5_letter_words` that identifies all five-letter words starting and ending with a vowel. The function should return the indices of these words within the original vector. For this problem, assume vowels are "a", "e", "i", "o", and "u", and consider case insensitivity.

```r
find_vowel_5_letter_words <- function(words) {
  pattern <- "^[aeiouAEIOU].{3}[aeiouAEIOU]$"
  matches <- str_detect(words, pattern)
  return(which(matches))
}

find_vowel_5_letter_words(c("Ananya", "Elena", "Aya"))
## [1] 2
```

b. (5 points)

```
comments <- c("Loving the new #season!", "Can't wait for #Friday!")
```

Write a command to remove all instances of # from the `comments` vector.

```
clean_comments <- str_replace_all(comments, "#", "")
print(clean_comments)
## [1] "Loving the new season!" "Can't wait for Friday!"
```

c. (5 points)

```
dates <- c("01012023", "15032024")
```

Write a command to insert dashes (-) into the dates vector to achieve the desired format.

```
formatted_dates <- str_c(
  str_sub(dates, 1, 2), "-",
  str_sub(dates, 3, 4), "-",
  str_sub(dates, 5, 8)
)
print(formatted_dates)
## [1] "01-01-2023" "15-03-2024"
```

d. (10 points)

```
info <- "John Doe:john.doe@example.com, Jane Smith:jane.smith@example.com"
```

Use the `stringr` functions to separate the names and email addresses into two vectors, `names` and `emails`.

```
pairs <- str_split(info, ",\\s*")[[1]]

# Further split each pair into name and email
separated_info <- str_split(pairs, ":")
names <- sapply(separated_info, `[`, 1)
emails <- sapply(separated_info, `[`, 2)

print(names)
## [1] "John Doe"   "Jane Smith"
print(emails)
## [1] "john.doe@example.com"   "jane.smith@example.com"
```

Q6 Shiny (10 points)

The Shiny application below contains three errors. Identify and correct these errors. Additionally, describe the purpose and functionality of this Shiny app in broad terms.

```
library(shiny)
library(gapminder)
dplyr::glimpse(gapminder)
## Rows: 1,704
## Columns: 6
## $ country   <fct> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanistan", ~
## $ continent <fct> Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, ~
## $ year      <int> 1952, 1957, 1962, 1967, 1972, 1977, 1982, 1987, 1992, 1997, ~
## $ lifeExp   <dbl> 28.801, 30.332, 31.997, 34.020, 36.088, 38.438, 39.854, 40.8~
```

```
## $ pop       <int> 8425333, 9240934, 10267083, 11537966, 13079460, 14880372, 12~
## $ gdpPercap <dbl> 779.4453, 820.8530, 853.1007, 836.1971, 739.9811, 786.1134, ~
```

```
ui <- fluidPage(
  titlePanel("Gapminder Data for Selected Country"),
  selectInput("countryInput", "Select a Country:", choices = unique(gapminder$country)),
  actionButton("actionButton", "Show Latest Data"),
  htmlOutput("countryInfo")
)

server <- function(input, output) {
  latestData <- eventReactive(input$showButton, {
    selectedCountry <- isolate(input$countryInput)
    gapminder %>%
      filter(country == selectedCountry) %>%
      tail(1)
  })

  output$countryInfo <- renderUI({
    HTML(stringr::str_c(
      "Country:", latestData$country, " ",
      "Life Expectancy:", round(latestData$lifeExp, 2), "years"
    ))
  })
}

shinyApp(ui, server)
```