Practicum 1

Aditya Elayavalli

2023-10-03

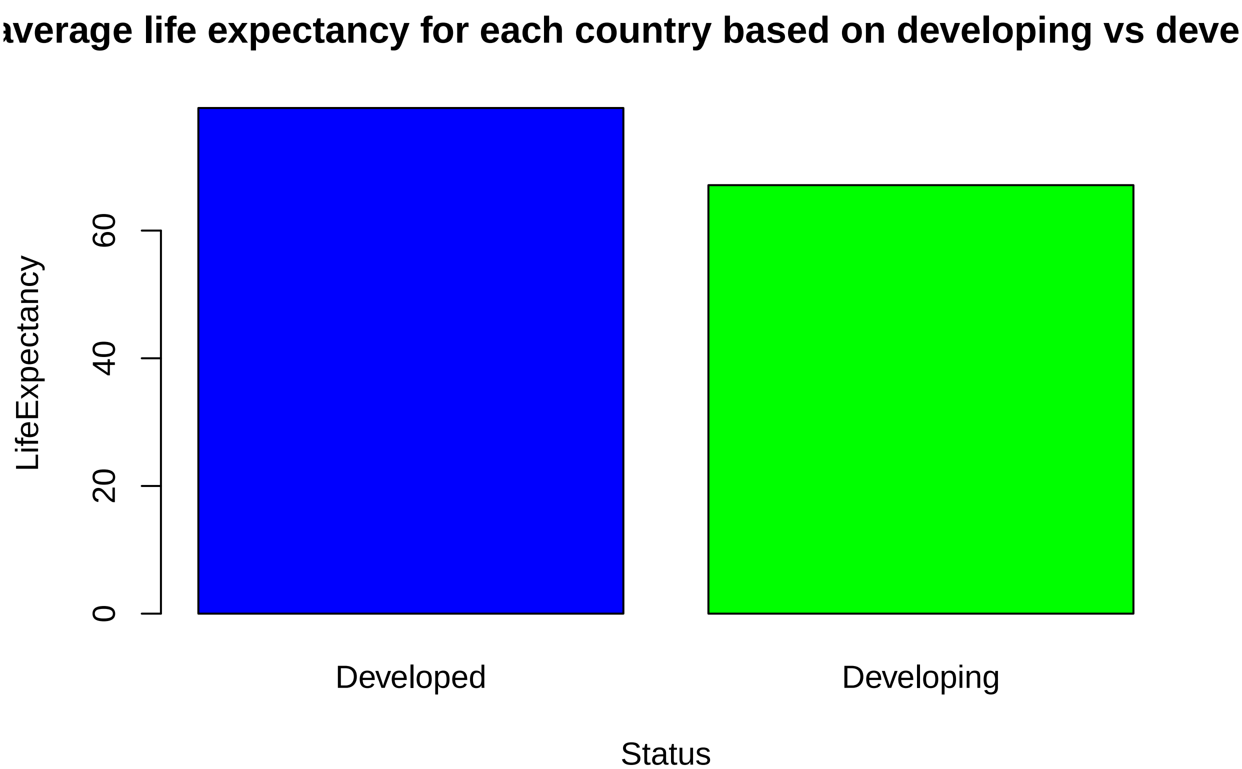
Question 1

```
## 'data.frame':    2938 obs. of  20 variables:
##  $ Country              : Factor w/ 193 levels "Afghanistan",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Year                 : int  2015 2014 2013 2012 2011 2010 2009 2008 2007 2006 ...
##  $ Status               : Factor w/ 2 levels "Developed","Developing": 2 2 2 2 2 2 2 2 2 2 ...
##  $ LifeExpectancy       : num  65 59.9 59.9 59.5 59.2 58.8 58.6 58.1 57.5 57.3 ...
##  $ AdultMortality       : int  263 271 268 272 275 279 281 287 295 295 ...
##  $ NumInfantDeaths      : int  62 64 66 69 71 74 77 80 82 84 ...
##  $ Alcohol              : num  0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.03 0.02 0.03 ...
##  $ PercentageExpenditure: num  71.3 73.5 73.2 78.2 7.1 ...
##  $ HepB                 : int  65 62 64 67 68 66 63 64 63 64 ...
##  $ Measles              : int  1154 492 430 2787 3013 1989 2861 1599 1141 1990 ...
##  $ BMI                  : num  19.1 18.6 18.1 17.6 17.2 16.7 16.2 15.7 15.2 14.7 ...
##  $ Under5Deaths         : int  83 86 89 93 97 102 106 110 113 116 ...
##  $ Polio                : int  6 58 62 67 68 66 63 64 63 58 ...
##  $ TotalExpenditure     : num  8.16 8.18 8.13 8.52 7.87 9.2 9.42 8.33 6.73 7.43 ...
##  $ Diphtheria           : int  65 62 64 67 68 66 63 64 63 58 ...
##  $ HIV                  : num  0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 ...
##  $ GDP                  : num  584.3 612.7 631.7 670 63.5 ...
##  $ thinness1.19y        : num  17.2 17.5 17.7 17.9 18.2 18.4 18.6 18.8 19 19.2 ...
##  $ thinness5.9y         : num  17.3 17.5 17.7 18 18.2 18.4 18.7 18.9 19.1 19.3 ...
##  $ Schooling            : num  10.1 10 9.9 9.8 9.5 9.2 8.9 8.7 8.4 8.1 ...
```

```
##       Country Year     Status LifeExpectancy AdultMortality NumInfantDeaths
## 1 Afghanistan 2015 Developing           65.0            263              62
## 2 Afghanistan 2014 Developing           59.9            271              64
## 3 Afghanistan 2013 Developing           59.9            268              66
## 4 Afghanistan 2012 Developing           59.5            272              69
## 5 Afghanistan 2011 Developing           59.2            275              71
##   Alcohol PercentageExpenditure HepB Measles  BMI Under5Deaths Polio
## 1    0.01             71.279624   65    1154 19.1           83     6
## 2    0.01             73.523582   62     492 18.6           86    58
## 3    0.01             73.219243   64     430 18.1           89    62
## 4    0.01             78.184215   67    2787 17.6           93    67
## 5    0.01              7.097109   68    3013 17.2           97    68
##   TotalExpenditure Diphtheria HIV       GDP thinness1.19y thinness5.9y
## 1             8.16         65 0.1 584.25921          17.2         17.3
## 2             8.18         62 0.1 612.69651          17.5         17.5
## 3             8.13         64 0.1 631.74498          17.7         17.7
## 4             8.52         67 0.1 669.95900          17.9         18.0
```

```
## 5             7.87         68 0.1   63.53723          18.2         18.2
##   Schooling
## 1      10.1
## 2      10.0
## 3       9.9
## 4       9.8
## 5       9.5
```

1.1 / Analysis of Data Distribution



From the graph, it's evident that developed countries have a higher average life expectancy at 79.19, compared to 67.11 for developing countries.

```
mean_life_w <- aggregate(df$LifeExpectancy,list(df$Country,df$Status),FUN = mean)
colnames(mean_life_w) <- c("Country","Status","LifeExpectancy")
developed <- mean_life_w$LifeExpectancy[which(mean_life_w$Status=="Developed")]
developing <- mean_life_w$LifeExpectancy[which(mean_life_w$Status=="Developing")]
a <- t.test(developed,developing)
a
```

```
##
##  Welch Two Sample t-test
##
## data:  developed and developing
## t = 13.362, df = 133.27, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
```

```
## 95 percent confidence interval:
##  10.29725 13.87552
## sample estimates:
## mean of x mean of y
##  79.19785  67.11147
```

Based on the t-test results, the obtained p-value is $2.2862506 \times 10^{-26}$. Given that this value is notably close to zero, we reject the null hypothesis which posited that there is no significant difference in the mean life expectancy between Developed and Developing countries. Since the p-value is less than the 0.05 threshold, we conclude that the difference in mean life expectancy between Developed and Developing countries is statistically significant

```r
df1 <- df[,4:20]
shapiro <- shapiro.test(df1$LifeExpectancy)
shapiro
```

```
##
##  Shapiro-Wilk normality test
##
## data:  df1$LifeExpectancy
## W = 0.95605, p-value < 2.2e-16
```

Based on the Shapiro test, the obtained p-value is $7.3604623 \times 10^{-29}$. Given its value is less than 0.05 and near zero, we reject the null hypothesis of normality. Therefore, we can conclude that the data does not adhere to a normal distribution.

1.2 / Identification of Outliers

```r
# Identify numeric columns
numeric_cols <- sapply(df, is.numeric)

# Initialize an empty list to store the row numbers of outliers for each column
outliers_list <- list()

# Loop through each numeric column and identify outliers
for(col_name in names(df)[numeric_cols]) {

  m <- mean(df[[col_name]], na.rm = TRUE)
  sd_col <- sd(df[[col_name]], na.rm = TRUE)

  z_scores <- abs((df[[col_name]] - m) / sd_col)

  outliers_list[[col_name]] <- which(z_scores > 3)
}

# Print the number of outliers and their rows for each numeric column
for(col_name in names(outliers_list)) {
  cat("\nColumn:", col_name, "\n")
  cat("Number of outliers:", length(outliers_list[[col_name]]), "\n")
  if(length(outliers_list[[col_name]]) > 0) {
    cat("Outlier positions:", outliers_list[[col_name]], "\n")
```

```
  } else {
    cat("No outliers detected\n")
  }
}
```

```
##
## Column: Year
## Number of outliers: 0
## No outliers detected
##
## Column: LifeExpectancy
## Number of outliers: 2
## Outlier positions: 1128 2313
##
## Column: AdultMortality
## Number of outliers: 40
## Outlier positions: 347 348 349 350 351 352 866 1128 1481 1482 1483 1484 1485 1486 1487 1488 1489 149
##
## Column: NumInfantDeaths
## Number of outliers: 37
## Outlier positions: 573 574 575 576 1187 1188 1189 1190 1191 1192 1193 1194 1195 1196 1197 1198 1199 
##
## Column: Alcohol
## Number of outliers: 3
## Outlier positions: 229 874 875
##
## Column: PercentageExpenditure
## Number of outliers: 84
## Outlier positions: 114 115 116 117 118 119 120 130 132 133 134 135 136 137 242 248 499 500 502 504 5
##
## Column: HepB
## Number of outliers: 18
## Outlier positions: 156 158 205 447 462 532 836 1389 1435 1439 1492 1785 1826 1833 1834 2663 2746 287
##
## Column: Measles
## Number of outliers: 48
## Outlier positions: 407 561 562 566 567 568 569 570 571 572 573 574 575 576 724 725 726 730 731 732 7
##
## Column: BMI
## Number of outliers: 0
## No outliers detected
##
## Column: Under5Deaths
## Number of outliers: 34
## Outlier positions: 575 576 1187 1188 1189 1190 1191 1192 1193 1194 1195 1196 1197 1198 1199 1200 120
##
## Column: Polio
## Number of outliers: 172
## Outlier positions: 1 12 49 60 61 64 105 128 154 158 205 214 221 233 276 279 300 310 326 327 404 405 
##
## Column: TotalExpenditure
## Number of outliers: 25
## Outlier positions: 1387 1497 1604 1651 1701 1704 1705 1706 2313 2714 2796 2797 2798 2799 2800 2801 2
```

```
##
## Column: Diphtheria
## Number of outliers: 170
## Outlier positions: 12 55 60 61 107 128 237 276 283 327 334 404 432 435 453 463 464 474 479 487 491 5
##
## Column: HIV
## Number of outliers: 69
## Outlier positions: 347 348 349 350 351 352 1378 1481 1482 1483 1484 1485 1486 1487 1488 1489 1490 15
##
## Column: GDP
## Number of outliers: 69
## Outlier positions: 113 114 115 116 117 118 130 133 136 499 500 739 740 742 744 745 746 1172 1178 117
##
## Column: thinness1.19y
## Number of outliers: 53
## Outlier positions: 5 6 7 8 9 10 11 12 13 14 195 196 197 198 199 200 201 202 203 301 302 303 304 1187
##
## Column: thinness5.9y
## Number of outliers: 57
## Outlier positions: 6 7 8 9 10 11 12 13 194 195 196 197 198 199 200 206 207 208 299 300 301 302 303 3
##
## Column: Schooling
## Number of outliers: 28
## Outlier positions: 75 76 77 78 79 80 336 850 1651 1715 1745 1746 1747 1748 2415 2416 2417 2418 2419
```

```
# Print the outliers' rows for each numeric column
```

The summary above lists the outliers within each column, along with their respective positions in the dataset. Outliers are data points that deviate significantly from other observations. Their presence can skew the analysis, potentially leading to erroneous predictions. It's crucial to address outliers correctly, either by normalization or removal, depending on the nature of the data.

Several techniques exist to detect outliers, including:

- **Standard Deviation**: Points more than three standard deviations from the mean are typically considered outliers.
- **Interquartile Range (IQR)**: In a box plot, any data point below $Q1 - 1.5 \times IQR$ or above $Q3 + 1.5 \times IQR$ is treated as an outlier, where $Q1$ and $Q3$ are the first and third quartiles, respectively.
- **Z-score**: A popular method where each data point's z-score (a measure in terms of standard deviation) is calculated. Points with z-scores greater than 3 are usually tagged as outliers.
- **DBSCAN Clustering**: This method forms clusters from core samples and marks non-core points as outliers.
- **Isolation Forest**: It distinguishes outliers by randomly selecting a feature and a split value between the maximum and minimum values of the selected feature.

For our analysis, we utilized the z-score technique to pinpoint outliers.

The table above provides a detailed count of outliers present in each column.

```
# checking to see if there is any NA
anyNA(df)
```

```
## [1] TRUE
```

```r
# Removing NAs from the life expectancy column in the data frame
df_L <- df$LifeExpectancy[which(!is.na(df$LifeExpectancy))]
```

The life expectancy column has a maximum value of 89 and a minimum of 36.3. Its standard deviation stands at 9.52.

The median represents the central value in a dataset, splitting it into two halves. Specifically, when data is sorted in ascending or descending order, the median is the middle number. For the life expectancy column, the median is 72.1.

A trimmed mean offers an average by excluding a predetermined percentage of extreme values (outliers) from both the top and bottom ends of the data. For instance, if we choose a 10% trim level, it will eliminate the top and bottom 10% of data before calculating the mean. This approach mitigates the influence of outliers.

Considering the life expectancy column, calculating a trimmed mean may not be the most effective as there are only two outliers. However, for columns like Polio and Diphtheria, which have a more substantial number of outliers, the trimmed mean becomes more relevant. Different trim percentages can be experimented with across columns to gauge how much outliers impact the average. It's pivotal to strike a balance between obtaining a reliable estimate and ensuring the precision of any derived model.

1.3 / Data Preparation

```r
# Create a new dataset to ensure that all relevant numerical data is put in
df_accurate <- df[, !(names(df) %in% c("Country", "Status", "Year"))]

z_score_normalization <- function(col) {
  if(is.numeric(col)) {
    mean_col <- mean(col, na.rm = TRUE)
    std_col <- sd(col, na.rm = TRUE)
    return ((col - mean_col) / std_col)
  } else {
    return(col)
  }
}

df_z_score_normalized <- as.data.frame(lapply(df_accurate, z_score_normalization))
```

In the provided code, we employ the equation $z = (x - average)/stddeviation$ or each column. By doing so, we're standardizing the features to exhibit characteristics of a standard normal distribution. This step ensures that data across columns are presented uniformly, eliminating potential biases and establishing a consistent basis for comparison. The primary objective is to ensure that every feature has a proportional influence on predictions, enhancing the overall accuracy of the model.

```r
df_z_score_normalized <- as.data.frame(df_z_score_normalized)
df_z_score_normalized$Disease <- df_z_score_normalized$HepB + df_z_score_normalized$Measles + df_z_score
summary(df_z_score_normalized)
```

```
##  LifeExpectancy    AdultMortality    NumInfantDeaths       Alcohol
##  Min.   :-3.4571   Min.   :-1.3178   Min.   :-0.25697   Min.   :-1.1334
##  1st Qu.:-0.6431   1st Qu.:-0.7305   1st Qu.:-0.25697   1st Qu.:-0.9193
```

```
##  Median : 0.3019   Median :-0.1673   Median :-0.23153   Median :-0.2092
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.0000
##  3rd Qu.: 0.6799   3rd Qu.: 0.5085   3rd Qu.:-0.07042   3rd Qu.: 0.7649
##  Max.   : 2.0764   Max.   : 4.4911   Max.   :15.00677   Max.   : 3.2739
##  NA's   :10        NA's   :10                           NA's   :194
##  PercentageExpenditure      HepB           Measles             BMI
##  Min.   :-0.3714       Min.   :-3.1887   Min.   :-0.2110   Min.   :-1.8620
##  1st Qu.:-0.3690       1st Qu.:-0.1572   1st Qu.:-0.2110   1st Qu.:-0.9490
##  Median :-0.3387       Median : 0.4411   Median :-0.2095   Median : 0.2584
##  Mean   : 0.0000       Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.:-0.1493       3rd Qu.: 0.6406   3rd Qu.:-0.1796   3rd Qu.: 0.8920
##  Max.   : 9.4278       Max.   : 0.7204   Max.   :18.2924   Max.   : 2.4436
##                        NA's   :553                          NA's   :34
##   Under5Deaths          Polio          TotalExpenditure     Diphtheria
##  Min.   :-0.26204   Min.   :-3.3944   Min.   :-2.22877   Min.   :-3.3868
##  1st Qu.:-0.26204   1st Qu.:-0.1937   1st Qu.:-0.67173   1st Qu.:-0.1823
##  Median :-0.23711   Median : 0.4464   Median :-0.07333   Median : 0.4501
##  Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.0000
##  3rd Qu.:-0.08755   3rd Qu.: 0.6171   3rd Qu.: 0.62214   3rd Qu.: 0.6188
##  Max.   :15.31709   Max.   : 0.7025   Max.   : 4.66786   Max.   : 0.7031
##  NA's   :1          NA's   :21        NA's   :226        NA's   :19
##       HIV                GDP          thinness1.19y      thinness5.9y
##  Min.   :-0.3234   Min.   :-0.5243   Min.   :-1.0723   Min.   :-1.0580
##  1st Qu.:-0.3234   1st Qu.:-0.4919   1st Qu.:-0.7329   1st Qu.:-0.7475
##  Median :-0.3234   Median :-0.4006   Median :-0.3483   Median :-0.3483
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.:-0.1855   3rd Qu.:-0.1102   3rd Qu.: 0.5340   3rd Qu.: 0.5167
##  Max.   : 9.6219   Max.   : 7.8268   Max.   : 5.1718   Max.   : 5.2629
##                    NA's   :448       NA's   :34        NA's   :34
##    Schooling           Disease
##  Min.   :-3.57043   Min.   :-9.5600
##  1st Qu.:-0.56351   1st Qu.:-0.7337
##  Median : 0.09146   Median : 1.1684
##  Mean   : 0.00000   Mean   : 0.1630
##  3rd Qu.: 0.68689   3rd Qu.: 1.6688
##  Max.   : 2.59226   Max.   :13.1334
##  NA's   :163        NA's   :555
```

1.4 / Sampling Training and Validation Data

```r
# Attach 'Status' to the normalized dataset
df_z_score_normalized <- cbind(df_z_score_normalized, df$Status)
colnames(df_z_score_normalized)[which(names(df_z_score_normalized) == "df$Status")] <- "Status"

# Determine indices for 'Developing' and 'Developed' countries
developing_countries <- which(df_z_score_normalized$Status == "Developing")
developed_countries <- which(df_z_score_normalized$Status == "Developed")

# Randomly select 15% from each country type
set.seed(13846) # Ensure reproducibility
developing_countries_sample <- sample(developing_countries, ceiling(0.15 * length(developing_countries)
developed_countries_sample <- sample(developed_countries, ceiling(0.15 * length(developed_countries)),
```

```
# Combine indices to form the total sample
total_samples <- c(developing_countries_sample, developed_countries_sample)

# Create training and validation datasets
df_val <- df_z_score_normalized[total_samples, ]
df_train <- df_z_score_normalized[-total_samples, ]

# Extract 'Status' labels from training and validation datasets
df_val_labels <- df_val$Status
df_train_labels <- df_train$Status

# Drop the 'Status' column from both datasets
df_val <- df_val[, -which(names(df_val) == "Status")]
df_train <- df_train[, -which(names(df_train) == "Status")]
```

1.5 / Predictive Modeling

```
library(class)

normalize <- function(data_point, training_data){
  means <- colMeans(training_data, na.rm = TRUE)
  std_devs <- apply(training_data, 2, sd, na.rm = TRUE)

  normalized_pt <- (data_point - means) / std_devs

  return(normalized_pt)
}

# Impute the median values to all columns for missing values in df_train
for (i in colnames(df_train)){
  miss <- is.na(df_train[,i])
  if (any(miss)){
    med <- median(df_train[,i], na.rm = TRUE)
    df_train[,i][miss] <- med
  }
}


# Create new data point
new_pt <- data.frame(
  LifeExpectancy = 66.4,
  AdultMortality = 275,
  NumInfantDeaths = 1,
  Alcohol = 0.01,
  PercentageExpenditure = 10,
  HepB = 40,
  Measles = 400,
  BMI = 17,
  Under5Deaths = 106,
  Polio = 10,
  TotalExpenditure = median(df_train$TotalExpenditure, na.rm = TRUE),
```

```r
  Diphtheria = 66,
  HIV = median(df_train$HIV, na.rm = TRUE),
  GDP = 620,
  thinness1.19y = median(df_train$thinness1.19y, na.rm = TRUE),
  thinness5.9y = median(df_train$thinness5.9y, na.rm = TRUE),
  Schooling = median(df_train$Schooling, na.rm = TRUE)
)

# Sum of diseases
new_pt$Disease <- new_pt$HepB + new_pt$Measles + new_pt$Polio + new_pt$Diphtheria

# Ensure that the column structure and ordering match between new_pt and df_train
new_pt <- new_pt[, colnames(df_train)]

# Normalize new_pt
normalized_pt <- normalize(new_pt, training_data = df_train)

# Apply kNN
model <- knn(train = df_train, test = normalized_pt, cl = df_train_labels, k = 5)
```

We initiated our process by standardizing each column using the z-score normalization technique. After this, the data was divided into a training set and a validation set. Using the kNN function from the `class` package, we predicted the category of a new data point. The prediction for this specific data point was determined as such.

kNN, or k-Nearest Neighbors, is a machine learning algorithm predominantly utilized for classification tasks. This method gauges the distance between a new, unlabelled data point and its surrounding data points, basing its prediction on the majority classification of its neighbors. The initial step entails computing the distance between the new data point and every other data point in the training set. Once computed, these distances are organized in ascending order.

A predetermined number, denoted as 'k', defines how many nearest neighbors should be considered. In this instance, we've chosen k as 5, so the nearest 5 data points are evaluated. The final prediction is determined by the predominant classification among these neighbors.

Upon concluding steps like data preprocessing and column normalization, we also standardized the new data point using the same method applied to the training set. Following these steps, our prediction categorized the provided country as Developing.

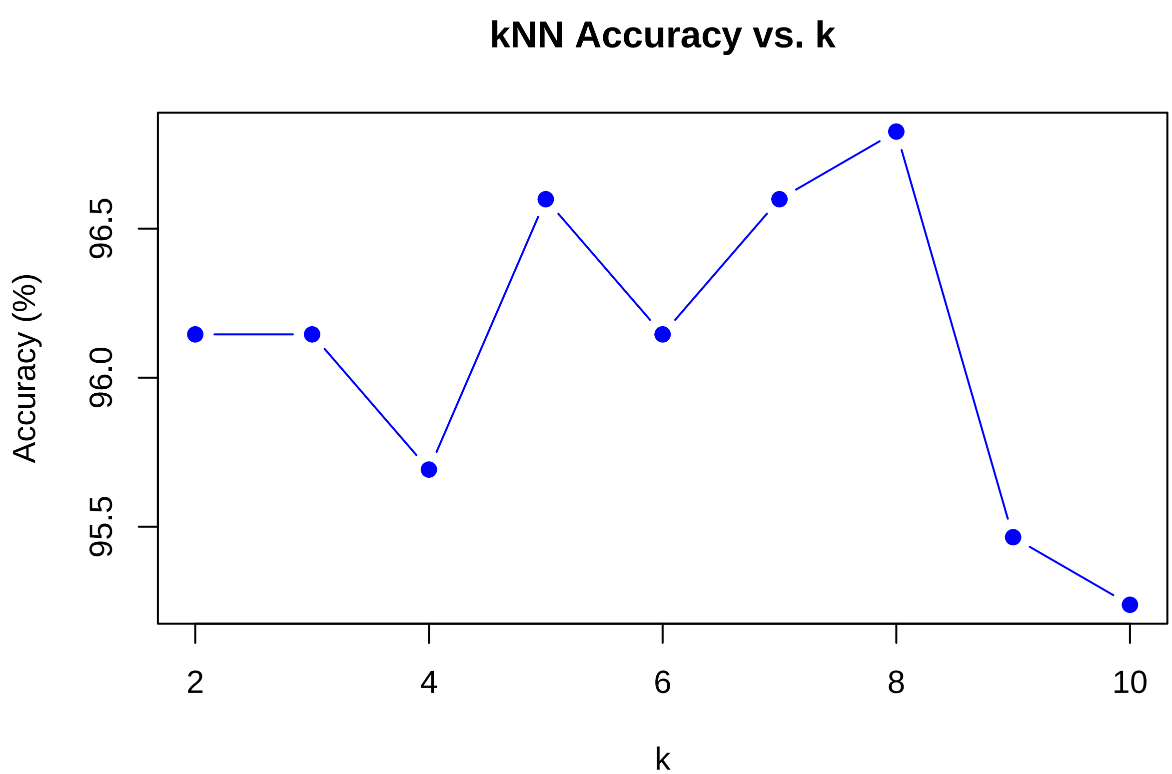
1.6 / Model Accuracy

```r
# Impute the data to median for validation dataset
for (i in colnames(df_val)){
  miss <- is.na(df_val[,i])
  if (any(miss)){
    med <- median(df_val[,i], na.rm = TRUE)
    df_val[,i][miss] <- med
  }
}

predictions <- data.frame(k = integer(), pred_labels = character())

# Predict using kNN for each observation in validation set
```

```r
for (i in 1:nrow(df_val)){
  unk <- df_val[i,]

  for (k in 2:10){
    knn_output <- knn(train = df_train, test = unk, cl = df_train_labels, k = k)
    predictions <- rbind(predictions, data.frame(k = k, pred_labels = knn_output))
  }
}

# Calculate accuracies
accuracies <- sapply(2:10, function(k) {
  predicted_labels <- predictions[predictions$k == k,]$pred_labels
  mean(predicted_labels == df_val_labels) * 100
})

print(accuracies)
```

```
## [1] 96.14512 96.14512 95.69161 96.59864 96.14512 96.59864 96.82540 95.46485
## [9] 95.23810
```

```r
# Plotting accuracies
plot(2:10, accuracies, type = "b", pch = 19, col = "blue",
     xlab = "k", ylab = "Accuracy (%)",
     main = "kNN Accuracy vs. k")
```



We visualized the outcomes of the k-Nearest Neighbors (kNN) method from the 'class' package to assess the model's precision. The graph depicts the relationship between different k values, spanning from 2 to 10, and their respective accuracy levels. Upon analyzing the graph, it became evident that the model's accuracy fluctuates with different k values. For our purposes, we aim to choose the k value that offers the best accuracy, balancing both bias and variance, depending on the specific dataset and its needs. Notably, the best accuracy was 96.8253968% achieved when k= 8. It's vital to normalize new data, akin to the training data, to ensure meaningful distance calculations and enhance the precision of the model.

Question 2

This problem requires analysis of a data set about abalones– a type of marine snail We are going to use various measurement to estimate the shucked weight from other measurements, which are easier to obtain. To do that, we will build a predictive model using a regression kNN algorithm.

The below code will give us the data frame along with inspecting the data

```
url2 <- "https://s3.us-east-2.amazonaws.com/artificium.us/datasets/abalone.csv"
df2 <- read.csv(url2,
               header = T,
               stringsAsFactors = F)

str(df2)
```

```
## 'data.frame':    4178 obs. of  9 variables:
##  $ Length       : num  0.455 0.35 0.53 0.44 0.33 0.425 0.53 0.545 0.475 0.55 ...
##  $ Diameter     : num  0.365 0.265 0.42 0.365 0.255 0.3 0.415 0.425 0.37 0.44 ...
##  $ Height       : num  0.095 0.09 0.135 0.125 0.08 0.095 0.15 0.125 0.125 0.15 ...
##  $ ShuckedWeight: num  0.2245 0.0995 0.2565 0.2155 0.0895 ...
##  $ VisceraWeight: num  0.101 0.0485 0.1415 0.114 0.0395 ...
##  $ ShellWeight  : num  0.15 0.07 0.21 0.155 0.055 0.12 0.33 0.26 0.165 0.32 ...
##  $ WholeWeight  : num  0.514 0.226 0.677 0.516 0.205 ...
##  $ NumRings     : int  15 7 9 10 7 8 20 16 9 19 ...
##  $ Sex          : chr  "M" "M" "F" "F" ...
```

further inspecting the data

```
head(df2,4)
```

```
##   Length Diameter Height ShuckedWeight VisceraWeight ShellWeight WholeWeight
## 1  0.455    0.365  0.095        0.2245        0.1010       0.150      0.5140
## 2  0.350    0.265  0.090        0.0995        0.0485       0.070      0.2255
## 3  0.530    0.420  0.135        0.2565        0.1415       0.210      0.6770
## 4  0.440    0.365  0.125        0.2155        0.1140       0.155      0.5160
##   NumRings Sex
## 1       15   M
## 2        7   M
## 3        9   F
## 4       10   F
```

```
tail(df2,4)
```

```
##      Length Diameter Height ShuckedWeight VisceraWeight ShellWeight WholeWeight
## 4175  0.600    0.475  0.205        0.5255        0.2875       0.308      1.1760
## 4176  0.625    0.485  0.150        0.5310        0.2610       0.296      1.0945
## 4177  0.710    0.555  0.195        0.9455        0.3765       0.495      1.9485
## 4178  0.710    0.515  0.162        0.8550        0.3750       0.450      1.7820
##      NumRings Sex
## 4175        9   M
## 4176       10   F
## 4177       12   M
## 4178       11   M
```

checking for any missing data

```r
anyNA(df2)
```

```
## [1] FALSE
```

Does not appear to be any missing data. The below code gives me the summary of the data frame

```r
summary(df2)
```

```
##      Length         Diameter          Height       ShuckedWeight
##  Min.   :0.075   Min.   :0.0550   Min.   :0.0000   Min.   :0.0010
##  1st Qu.:0.450   1st Qu.:0.3500   1st Qu.:0.1150   1st Qu.:0.1861
##  Median :0.545   Median :0.4250   Median :0.1400   Median :0.3360
##  Mean   :0.524   Mean   :0.4079   Mean   :0.1395   Mean   :0.3595
##  3rd Qu.:0.615   3rd Qu.:0.4800   3rd Qu.:0.1650   3rd Qu.:0.5020
##  Max.   :0.815   Max.   :0.6500   Max.   :1.1300   Max.   :1.4880
##  VisceraWeight     ShellWeight      WholeWeight        NumRings
##  Min.   :0.0005   Min.   :0.0015   Min.   :0.0020   Min.   : 1.000
##  1st Qu.:0.0935   1st Qu.:0.1300   1st Qu.:0.4416   1st Qu.: 8.000
##  Median :0.1710   Median :0.2340   Median :0.7997   Median : 9.000
##  Mean   :0.1806   Mean   :0.2389   Mean   :0.8290   Mean   : 9.934
##  3rd Qu.:0.2530   3rd Qu.:0.3290   3rd Qu.:1.1538   3rd Qu.:11.000
##  Max.   :0.7600   Max.   :1.0050   Max.   :2.8255   Max.   :29.000
##      Sex
##  Length:4178
##  Class :character
##  Mode  :character
##
##
##
```

2.1 / Predicting Shucked Weight of Abalones using Regression kNN

```r
# Save the values of the "Shucked Weight"" column in a separate vector called target_data
target_data <- df2$ShuckedWeight

# Create a new dataset called train_data containing all the training features
train_data <- df2[, !(names(df2) == "Shucked Weight")]
```

2.2 / Encoding

```r
sex <- as.factor(train_data$Sex)
levels(sex)
```

```
## [1] "F" "I" "M"
```

One-Hot Encoding is an effective way to transform nominal categorical variables into a format that can be provided to machine learning algorithms to do a better job in prediction. This method creates binary columns for each category and indicates the presence of the categories with a 1 or 0. Since "Sex" is a nominal categorical variable (categories that don't have a natural order), one-hot encoding is suitable.

```r
# initialize the columns First and Second to 0
train_data$Male <- 0
train_data$Female <- 0
train_data$Intersex <- 0

# set the columns to 1 if it's the corresponding class
train_data$Male <- ifelse(train_data$Sex == "M", 1, 0)
train_data$Female <- ifelse(train_data$Sex == "F", 1, 0)
train_data$Intersex <- ifelse(train_data$Sex == "I", 1, 0)
head(train_data[c(10,11,12)], 10)
```

```
##    Male Female Intersex
## 1     1      0        0
## 2     1      0        0
## 3     0      1        0
## 4     0      1        0
## 5     0      0        1
## 6     0      0        1
## 7     0      1        0
## 8     0      1        0
## 9     1      0        0
## 10    0      1        0
```

2.3 / Min Max-Normalization

```r
# need to create a function for the Normalization
# Define the normalization function
normalization <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}

# Function to apply normalization selectively to numeric columns and avoid the 'Sex' column
normalize_cols_except_sex <- function(data) {
  is_numeric <- sapply(data, is.numeric)
  is_not_sex <- names(data) != "Sex"

  # Columns that are numeric and not 'Sex'
  to_normalize <- is_numeric & is_not_sex
```

```r
  data[to_normalize] <- lapply(data[to_normalize], normalization)
  return(data)
}

# Apply the normalization
train_data <- normalize_cols_except_sex(train_data)
summary(train_data)
```

```
##      Length          Diameter          Height       ShuckedWeight
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.5068   1st Qu.:0.4958   1st Qu.:0.1018   1st Qu.:0.1245
##  Median :0.6351   Median :0.6218   Median :0.1239   Median :0.2253
##  Mean   :0.6068   Mean   :0.5931   Mean   :0.1235   Mean   :0.2411
##  3rd Qu.:0.7297   3rd Qu.:0.7143   3rd Qu.:0.1460   3rd Qu.:0.3369
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##  VisceraWeight     ShellWeight      WholeWeight        NumRings
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.1224   1st Qu.:0.1281   1st Qu.:0.1557   1st Qu.:0.2500
##  Median :0.2245   Median :0.2317   Median :0.2825   Median :0.2857
##  Mean   :0.2372   Mean   :0.2366   Mean   :0.2929   Mean   :0.3191
##  3rd Qu.:0.3325   3rd Qu.:0.3264   3rd Qu.:0.4079   3rd Qu.:0.3571
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##      Sex                 Male            Female          Intersex
##  Length:4178        Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  Class :character   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
##  Mode  :character   Median :0.0000   Median :0.0000   Median :0.0000
##                     Mean   :0.3657   Mean   :0.3131   Mean   :0.3212
##                     3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000
##                     Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
```

2.4 / KNN

The below code will feature a function called "knn.reg" that implements a regression version of kNN. that averages the value of the "Shucked Weight" of the "k"nearest neighbors using a weighted average where the weight is 3 for the closest neighbor, 2 for the second closest, and 1 for the remaining neighbors

```r
# Calculate euclidean distance
euclidean_dist <- function(x, y) {
    dist <- 0
    for (i in 1:length(x)) {
        dist <- dist + (x[i] - y[i])^2
    }
    dist <- sqrt(dist)
    return(dist)
}

Neighbor_dist <- function(training_pts, unknown_pt) {
    pts <- nrow(training_pts)
    dist_2_pt <- numeric(pts)
    for (i in 1:pts) {
        training_pt <- training_pts[i,]
        dist_2_u <- euclidean_dist(training_pt[,1], unknown_pt[,1])
```

```
        dist_2_pt[i] <- dist_2_u
    }
    return(dist_2_pt)
}

Neighbor_select <- function(dists, k) {
    arrange <- order(dists)
    closest_pts <- arrange[1:k]
    return(closest_pts)
}

wma <- function(w, closest_pts, target_data) {
    wma_avg <- sum(w * target_data[closest_pts]) / sum(w)
    return(wma_avg)
}

knn.reg <- function(new_data, target_data, train_data, k) {
    distances <- Neighbor_dist(training_pts = train_data, unknown_pt = new_data)
    closest_pts <- Neighbor_select(distances, k)

    # If more than one row in new_data, compute the weighted average for each row and then take the ave
    if (nrow(new_data) > 1) {
        all_predictions <- sapply(1:nrow(new_data), function(row_num) {
            wma(w = c(3, 2, 1), closest_pts, target_data)
        })
        return(mean(all_predictions))
    } else {
        weighted_avg <- wma(w = c(3, 2, 1), closest_pts, target_data)
        return(weighted_avg)
    }
}

n <- nrow(train_data)
set.seed(876587)
training_points <- sample(1:n, ceiling(0.8 * n), replace = F)
training_dataset <- train_data[training_points,]
training_labels <- target_data[training_points]
val_dataset <- train_data[-training_points,]
val_labels <- target_data[-training_points]

# Test cases
prediction_1 <- knn.reg(val_dataset[1,], training_labels, training_dataset, k=3)
print(prediction_1)
```

```
## [1] 0.2745
```

```
prediction_2 <- knn.reg(val_dataset, training_labels, training_dataset, k=3)
print(prediction_2)
```

```
## [1] 0.2745
```

2.5 / Forecasting

```
# Create new abalone data without the Sex column
new_point <- data.frame(
  Length = 0.34,
  Diameter = 0.491,
  Height = 0.245,
  VisceraWeight = 0.0887,
  ShellWeight = 0.19,
  WholeWeight = 0.4853,
  NumRings = 10
)

# Define columns to be normalized
columns_to_normalize <- c("Length", "Diameter", "Height", "VisceraWeight", "ShellWeight", "WholeWeight"

# Ensure the new_point and training_dataset columns are numeric for these columns
new_point[columns_to_normalize] <- lapply(new_point[columns_to_normalize], as.numeric)
training_dataset[columns_to_normalize] <- lapply(training_dataset[columns_to_normalize], as.numeric)

# Normalize the data using min-max normalization
min_vals_cols <- sapply(training_dataset[columns_to_normalize], min)
max_vals_cols <- sapply(training_dataset[columns_to_normalize], max)

normalize_data <- function(data_point, min_vals, max_vals) {
  return(sweep(sweep(data_point, 2, min_vals, "-"), 2, max_vals - min_vals, "/"))
}

new_point_normalized <- normalize_data(new_point, min_vals_cols, max_vals_cols)

# Ensure there are no NAs in the normalized data
if (sum(is.na(new_point_normalized)) > 0) {
  stop("There are NA values in the normalized data!")
}

# Apply the knn.reg function
pred <- knn.reg(new_data = new_point_normalized, target_data = training_labels, train_data = training_da

# Print the result
print(paste("The predicted value for the given new point is:", round(pred, 4)))
```

```
## [1] "The predicted value for the given new point is: 0.058"
```

2.6 / MSE

```
set.seed(677687)
samp_size <- sample(nrow(df2), size = 0.15 * nrow(df2), replace = F)
testing <- df2[samp_size,]
training <- df2[-samp_size,]

# One-hot encoding
```

```
training$Male <- ifelse(training$Sex == "M", 1, 0)
training$Female <- ifelse(training$Sex == "F", 1, 0)
training$Intersex <- ifelse(training$Sex == "I", 1, 0)

testing$Male <- ifelse(testing$Sex == "M", 1, 0)
testing$Female <- ifelse(testing$Sex == "F", 1, 0)
testing$Intersex <- ifelse(testing$Sex == "I", 1, 0)

train_labels <- training$ShuckedWeight
test_labels <- testing$ShuckedWeight

# Drop the original 'Sex' and 'ShuckedWeight' columns
training <- training[, !(names(training) %in% c("Sex", "ShuckedWeight"))]
testing <- testing[, !(names(testing) %in% c("Sex", "ShuckedWeight"))]

start <- Sys.time()
predictions <- knn.reg(new_data = testing, train_data = training, target_data = train_labels, k = 3)
ends <- Sys.time()

mse <- mean((test_labels - predictions)^2)

print(paste("The mean squared error is, ", round(mse,4)))
```

```
## [1] "The mean squared error is,  0.0645"
```

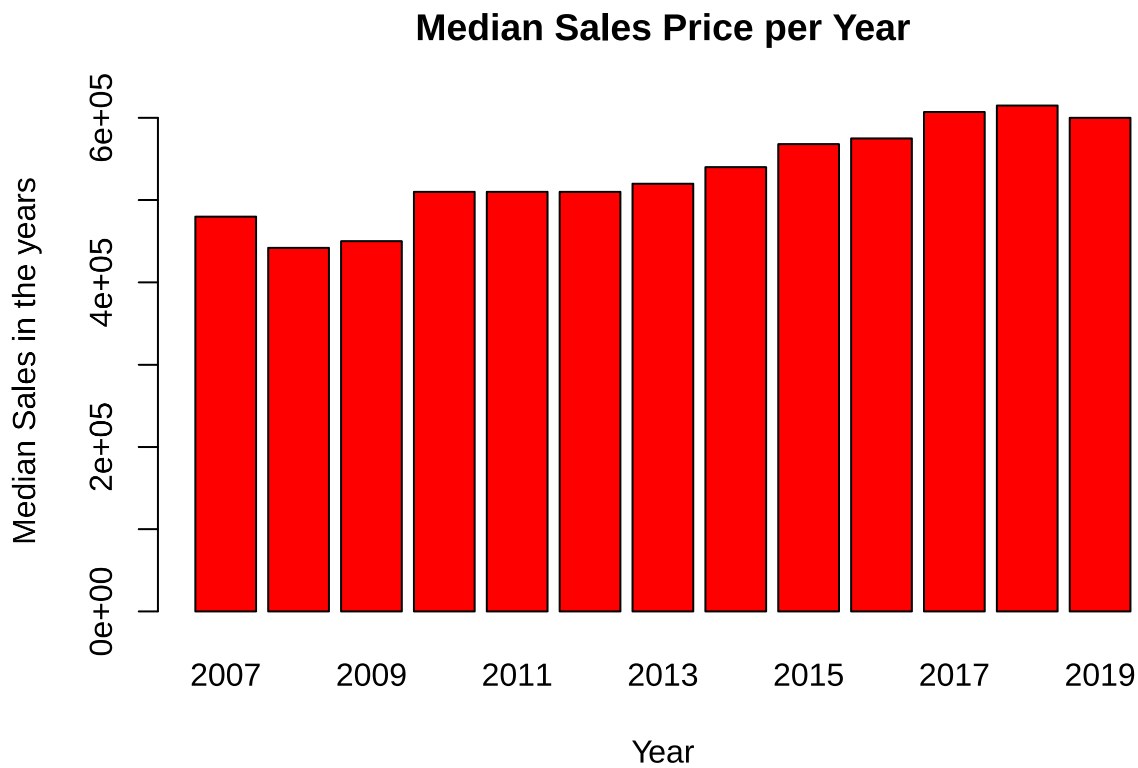
Question 3

3 / Forecasting Future Sales Price

We obtained a data set with a total of 29580 sales transactions for the years from 2007 to 2019. The median sales price for the entire time frame was 5.5×10^5, while the 10% trimmed mean was 5.7184458×10^5 (sd = 2.8170791×10^5). Broken down by year, we have the following 10% trimmed mean and median sales prices per year:

Year	AnnualMean	AnnualMedian
2007	489189.1	480000
2008	463226.3	442000
2009	470796.4	450000
2010	528650.1	510000
2011	528366.7	510000
2012	522672.9	510000
2013	532317.0	520000
2014	558258.4	540000
2015	586370.3	568000
2016	593631.5	575000
2017	628977.0	607000
2018	623654.4	615000
2019	608294.2	600000

As the graph below shows, the median sales price per year has been increasing



Using both a weighted moving average forecasting model that averages the prior 3 years (with weights of 4, 3, and 1) and a linear regression trend line model, we predict next year's average sales price to be around $583,878.6.