Practicum 1

Merim Tojaga
Northeastern University
DA 5030: Intro to Data Mining/Machine Learning

October 07, 2024

Libraries Used in Practicum.

```r
library(dplyr)
library(tidyverse)
library(scales)
library(ggplot2)
library(mosaic)
library(kableExtra)
library(class)
library(gmodels)
library(TTR)
library(forecast)
```

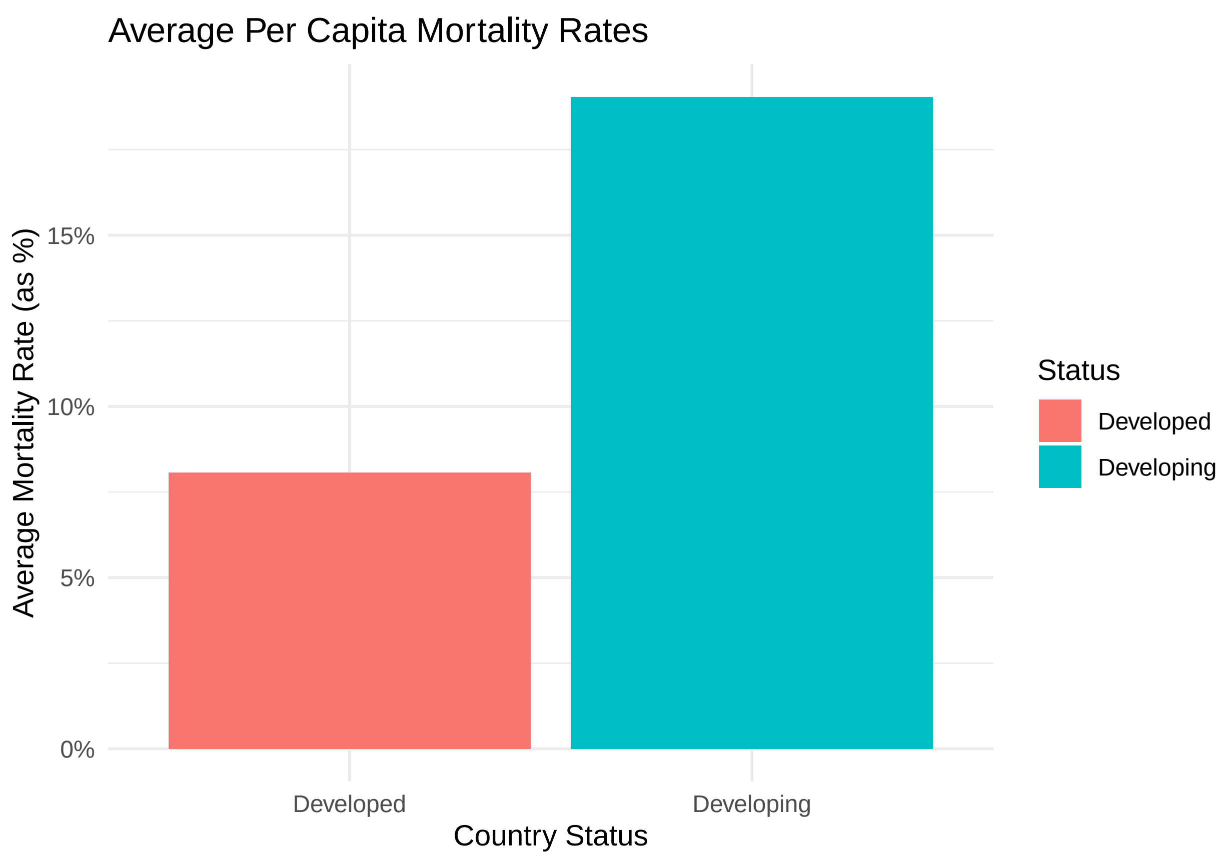
1 / Predicting Life Expectancy.

Importing Data.

```r
url <- "https://s3.us-east-2.amazonaws.com/artificium.us/datasets/LifeExpectancyData.csv"

df <- read.csv(url)
```

1.1 / Analysis of Per Capita Mortality.

- Question 2: “Developing” vs “Developed” countries Bar Chart.



We can see the massive difference in developing nations, as they have a much higher mortality rate than developed nations. Some of the values are very high making me assume I made a mistake in data.

- Question 3: Determine the difference in mortality using appropriate test.

```
#first we check if they are normally distributed

developed <- df_clean[df_clean$Status == "Developed",]
dped_mort <- developed$total_mort
developing <- df_clean[df_clean$Status == "Developing",]
dping_mort <- developing$total_mort

#Both groups are normally distributed.
shapiro_val_dped <- shapiro.test(dped_mort)
pval_dped <- shapiro_val_dped$p.value
shapiro_val_dping <- shapiro.test(dping_mort)
pval_dping <- shapiro_val_dping$p.value
wilcox_results <- wilcox.test(dped_mort, dping_mort)
wil_pval <- wilcox_results$p.value
```

To check the difference in mortality rate between the countries we first have to check if our data is normally distributed. We can see Developed Nations have a p-value of $3.5628687 \times 10^{-14}$, & Developing Nations with a p-value of $2.4812273 \times 10^{-27}$ which are both below 0.05 meaning our data is not normally distributed. The appropriate t-test for this data is the Wilicox rank-sum test. Our results from the wilicox test were a p-value $3.4956717 \times 10^{-69}$ which indicates a significant difference in distribution in mortality rate between Developed and Developing Nations.

- Question 4: Test the normality of the column "Life expectancy".

Conducting a shapiro test on the column "Life expectancy" in our data frame we can see that the p-value is $7.3604623 \times 10^{-29}$ which is far under 0.05 ultimately meaning the data in not normally distributed.

1.2 / Identification of Outliers.

```r
calculate_z_score <- function(col) {
  mean_col <- mean(col, na.rm = TRUE)
  sd_col <- sd(col, na.rm = TRUE)
  z_scores <- (col - mean_col) / sd_col
}
```

We can apply the above function to all columns

Column	OutlierCount
Year	0
Life.expectancy	5
Adult.Mortality	52
infant.deaths	50
Alcohol	5
percentage.expenditure	97
Hepatitis.B	159
Measles	51
BMI	0
under.five.deaths	36
Polio	172
Total.expenditure	28
Diphtheria	170
HIV.AIDS	73
GDP	88
Population	18
thinness..1.19.years	66
thinness.5.9.years	71
Income.composition.of.resources	130
Schooling	28

The Table above showcases the number of outliers each column has. These outliers are more than 2.8 Standard deviations from the mean. Hepatitus.B defeinetly contains a lot of outliers.

min	Q1	median	Q3	max	mean	sd	n	missing
36.3	39	41	41.5	42.3	40.02	2.409772	5	10

The Table above goes into descriptive statistics on the outliers for Life Expectancy. We can see that all of the outliers are very low, indicating an average age of 40. The values like min, max and median can be observed above in the kable table.

I would not use a trimmed mean for Life Expectancy because we are only talking about 5 values that are all low, if we used a trimmed mean we would cut into values that are high as well. In this case, we need to be careful about removing outliers about life expectancy when we are trying to calculate mortallity. I would remove all 5 of the outliers as they could potentially skew the data. In this case a country with a life expectancy in 40's is definitely going to skew data on mortality.

1.3 / Data Preparation.

- Question 1: Normalize all numeric columns using z-score standardization.

Normalizing the data above allows us to compare the data easily. As it takes things that are very far apart numerically and standardize them, making them useful for algorithms.

- Question 2: Add a new, derived feature to the dataframe called "disease"

```
sd_cleaned_df <- sd_cleaned_df %>%
  group_by(Country) %>%
  mutate(Disease = rowSums(pick(Hepatitis.B, Measles, Polio, HIV.AIDS, Diphtheria), na.rm = TRUE)) %>%
  ungroup()
```

1.4 / Sampling Training and Validation Data.

```
shuffled_data <- z_cleaned_df %>% sample_frac(size = 1)

validation_data <- shuffled_data %>%
  group_by(Status) %>%
  slice_sample(prop = .15) %>%
  ungroup()

training_data <- shuffled_data %>%
  filter(!row_number() %in% row_number(validation_data))
```

1.5 / Predictive Modeling.

```
anyNA(z_cleaned_df)
```

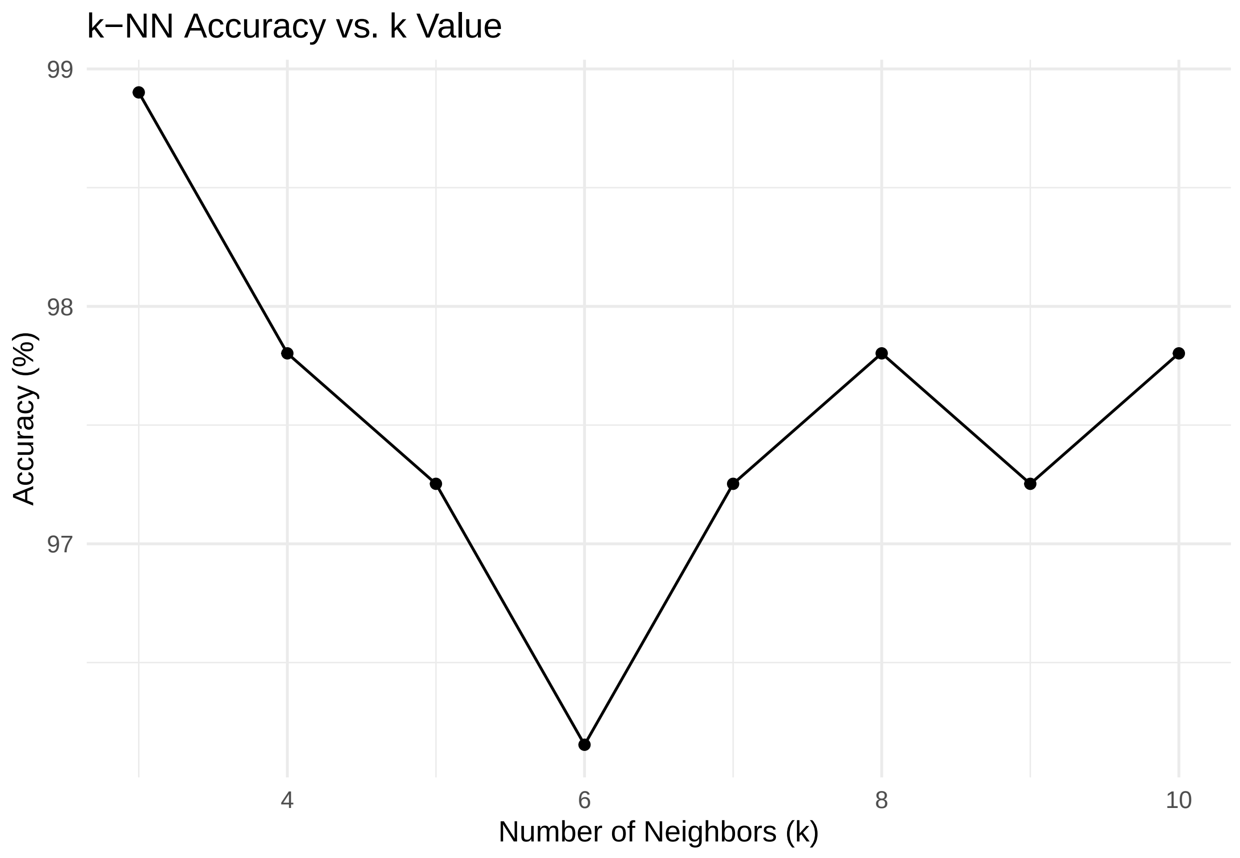
```
## [1] FALSE
```

I used the kNN algorithm to make the prediction analysis. I had to prepare my data using the z score standardization (In order to give the algorithm values to make better predictions with), and then make sure I had no NA values. From there I split the data set randomly into a 85% Training data and 15% Validation data. I used kNN because of my experience with it in a previous assignment for this class, I understand that it is an easy to use analysis tool that can help me in making predictions for future datasets. My prediction based on this data would be a value of "Developing" soley based on under 5 deaths, as I think I remember seeing that pediatric care is something many developing nations lack.

After kNN: The predicted status is Developing. Which definitely did not surpise me, I looked at the factor of infant deaths and thought it would be pretty hard to see that in a developed country.

1.6 / Model Accuracy.

```r
k_values <- 3:10
accuracies <- numeric(length(k_values))


for (i in seq_along(k_values)) {
  k <- k_values[i]
  predicted_labels <- knn(train = predict_train, test = predict_val, cl = training_data$Status, k = k)
  accuracies[i] <- mean(predicted_labels == validation_data$Status) * 100 #
}

results <- data.frame(k = k_values, accuracy = accuracies)


ggplot(results, aes(x = k, y = accuracy)) +
  geom_line() +
  geom_point() +
  labs(title = "k-NN Accuracy vs. k Value",
       x = "Number of Neighbors (k)",
       y = "Accuracy (%)") +
  theme_minimal()
```



2 / Predicting Shucked Weight of Abalones using Regression kNN

Getting Dataset

2.2 / Encoding Categorical Variables

```
target_data <- abalone_df$ShuckedWeight

training_data <- abalone_df %>%
  select(-ShuckedWeight)
```

- Question 2: Encode all categorical columns using an encoding scheme of your choice.

```
# Dummy Code Sex
training_data$sexID_F <- ifelse(training_data$Sex == "F",1,0)
training_data$sexID_I <- ifelse(training_data$Sex == "I",1,0)
```

I chose to use Dummy Encoding (also known as One Hot Encoding) as I have experience with it in DA5020 and I know it is very simple to manage. I only saw one candidate for a categorical type of encoding like this in the "Sex" column. The number of rings column seemed like it was connected to age, and had a lot of values so I assumed it was better to normalize this column. The Dummy encoding in this code asigns a zero or one based on the column value, because we have 3 levels in the "Sex" column I only needed two ;dummy encodings' because the computer is smart enough to recognize that values that aren't taken by "F" or "I" are taken by "M".

- Question 3: Normalize appropriate columns in train_data using min-max normalization.

```
# Introduce my min-max normalization function.
calculate_min_max <- function(col) {
  min_col <- min(col, na.rm = TRUE)
  max_col <- max(col, na.rm = TRUE)
  min_max_norm <- (col - min_col) / (max_col - min_col)
}
```

- Question 4: Build a function called knn.reg that averages the value of the "Shucked Weight" of the 'k' nearest neighbors using a simple average.

We have to create a random test sample, so we need to complete the process of separating the the shucked weight and normalizing/encoding the values.

```
# Function Running
k <- 5
predictions <- knn.reg(new_data_sample, target_data, training_data, k)

mae <- mean(abs(predictions - new_target_sample))
rmse <- sqrt(mean((predictions - new_target_sample)^2))


mae <- percent(mae/mean(new_target_sample))
rmse <- percent(rmse/mean(new_target_sample))
```

Original KNN Regression function took an hour to run, so I researched techniques on efficiency and was able to create a frankenstien of code from the professor, the text book, and editing the code. The new code can run efficiently and effectively having a Root Mean Squared Error value of 22% and a Mean Absolute Error of 15% these values are pretty good considering that we are measuring something that can have a lot of variation like weight.

- Question 5: Forecast the Shucked Weight of this new abalone using your regression kNN using k= 3

```
prediction_new_sample <- knn.reg(new_abalone, target_data, training_data, 3)
samp <- round(prediction_new_sample, 4)
```

Given the information about the sample our model predict a Shucked Weight of 0.2212. I am resonalby confident in this answer given our low RMSE and MAE.

- Question 6: Calculate the Mean Squared Error (MSE) using a random sample of 20% of the data set as test data.

```
#Running kNN function on the data
predictions_two <- knn.reg(second_data_sample, target_data, training_data, 3)
mse <- mean((second_data_target - predictions_two)^2)
```

For a random sample of 20% of the original data we got a MSE 0.0062996. This value is low even with our values being decimals. It indicates our model is fairly accurate in making predictions. Hopefully it is not over fitting.

3 / Forecasting Future Sales Price

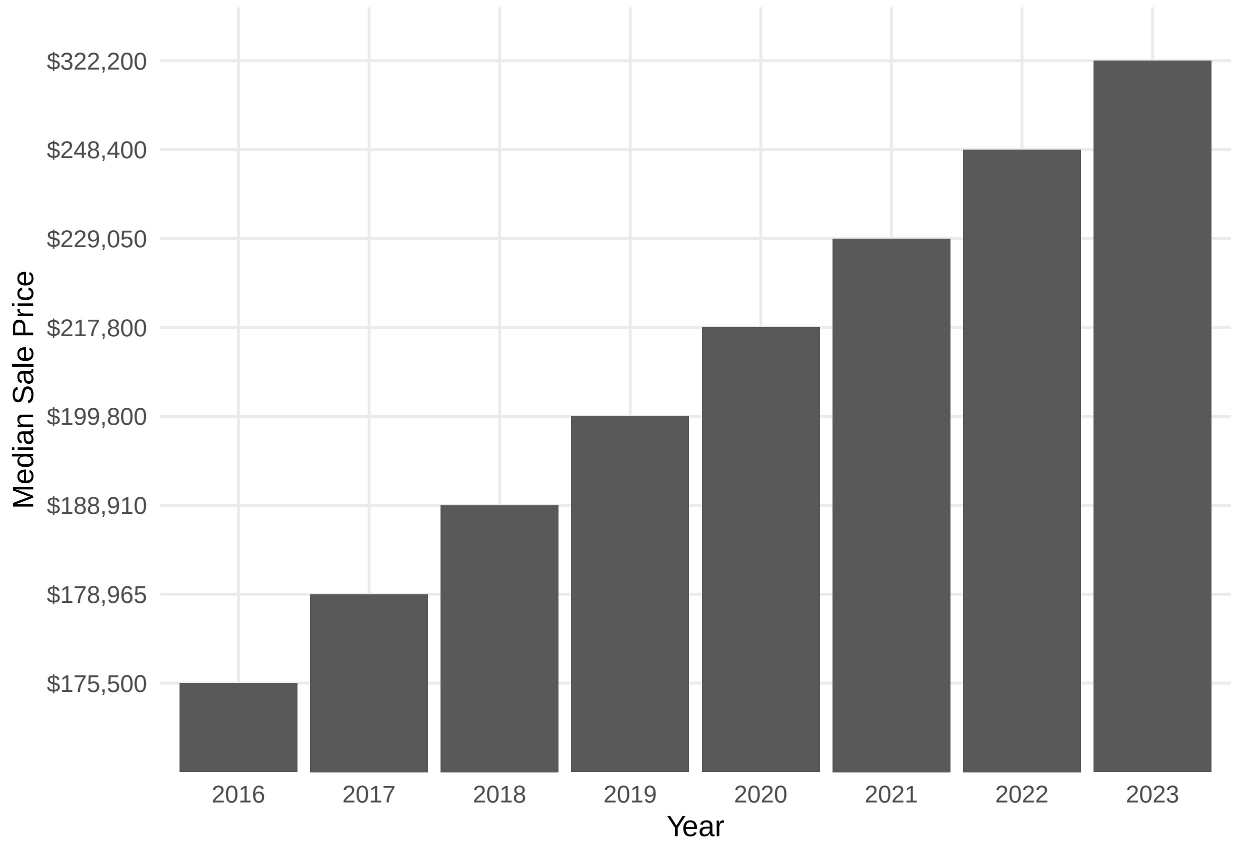
```
rm(list = ls())

url <- "https://s3.us-east-2.amazonaws.com/artificium.us/datasets/HomeSalesUFFIData.csv"

HomeSales_df <- read.csv(url)
```

We obtained a data set with a total of 99 sales transactions for the years from 2016 to 2023. The median sales price for the entire time frame was \$207,000, while the 10% trimmed mean was 2.12869×10^5 (sd = 7.2614×10^4). Broken down by year, we have the following number of sales, plus the 10% trimmed mean and median sales prices per year

YearSold	Homes Sold	Average Sales Price (10% Trim)	Median Sale Price
2016	6	\$169,170	\$175,500
2017	18	\$189,349	\$178,965
2018	20	\$180,810	\$188,910
2019	8	\$201,038	\$199,800
2020	9	\$220,500	\$217,800
2021	12	\$230,562	\$229,050
2022	13	\$252,327	\$248,400

2023	13	$310,909	$322,200

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



```
## [1] 298400.9
```

Using both a weighted moving average forecasting model that averages the prior 3 years (with weights of 0.7, 0.2, and 0.1) and a linear regression trend line model, we predict next year's average sales price to be around $294,779 (average of the two forecasts). The average home price of homes with both pools and air conditioning changed from $218,700 in 2017 (earliest year for which data is available) to $624,600 in 2023 (most recent year's sales data).