Medical Premium Price Prediction Final Project

Erika, Jason, Fiona

2025-04-24

# Import the libraries

rm(list=ls())  
#install.packages("MASS")  
#install.packages("tidymodels")  
# install.packages("glmnet")  
# install.packages("rpart.plot")  
# install.packages("yardstick")  
# install.packages("rpart")  
library(readr)  
library(tidyverse)

## Warning: package 'ggplot2' was built under R version 4.3.3

## Warning: package 'purrr' was built under R version 4.3.3

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.4 ✔ purrr 1.0.4  
## ✔ forcats 1.0.0 ✔ stringr 1.5.1  
## ✔ ggplot2 3.5.2 ✔ tibble 3.2.1  
## ✔ lubridate 1.9.3 ✔ tidyr 1.3.1  
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

library(ggplot2)  
library(reshape2)

##   
## Attaching package: 'reshape2'  
##   
## The following object is masked from 'package:tidyr':  
##   
## smiths

library(glmnet)

## Warning: package 'glmnet' was built under R version 4.3.3

## Loading required package: Matrix  
##   
## Attaching package: 'Matrix'  
##   
## The following objects are masked from 'package:tidyr':  
##   
## expand, pack, unpack  
##   
## Loaded glmnet 4.1-8

library(dplyr)  
library(caret) #machine learning library

## Warning: package 'caret' was built under R version 4.3.3

## Loading required package: lattice  
##   
## Attaching package: 'caret'  
##   
## The following object is masked from 'package:purrr':  
##   
## lift

library(xgboost) #XGBoost library

## Warning: package 'xgboost' was built under R version 4.3.3

##   
## Attaching package: 'xgboost'  
##   
## The following object is masked from 'package:dplyr':  
##   
## slice

library(rpart)  
library(MASS)

##   
## Attaching package: 'MASS'  
##   
## The following object is masked from 'package:dplyr':  
##   
## select

library(yardstick)

## Warning: package 'yardstick' was built under R version 4.3.3

##   
## Attaching package: 'yardstick'  
##   
## The following objects are masked from 'package:caret':  
##   
## precision, recall, sensitivity, specificity  
##   
## The following object is masked from 'package:readr':  
##   
## spec

library(tidymodels)

## Warning: package 'tidymodels' was built under R version 4.3.3

## ── Attaching packages ────────────────────────────────────── tidymodels 1.3.0 ──  
## ✔ broom 1.0.8 ✔ recipes 1.2.1  
## ✔ dials 1.4.0 ✔ rsample 1.3.0  
## ✔ infer 1.0.8 ✔ tune 1.3.0  
## ✔ modeldata 1.4.0 ✔ workflows 1.2.0  
## ✔ parsnip 1.3.1 ✔ workflowsets 1.1.0

## Warning: package 'broom' was built under R version 4.3.3

## Warning: package 'dials' was built under R version 4.3.3

## Warning: package 'infer' was built under R version 4.3.3

## Warning: package 'modeldata' was built under R version 4.3.3

## Warning: package 'parsnip' was built under R version 4.3.3

## Warning: package 'recipes' was built under R version 4.3.3

## Warning: package 'rsample' was built under R version 4.3.3

## Warning: package 'tune' was built under R version 4.3.3

## Warning: package 'workflows' was built under R version 4.3.3

## ── Conflicts ───────────────────────────────────────── tidymodels\_conflicts() ──  
## ✖ scales::discard() masks purrr::discard()  
## ✖ Matrix::expand() masks tidyr::expand()  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ recipes::fixed() masks stringr::fixed()  
## ✖ dplyr::lag() masks stats::lag()  
## ✖ caret::lift() masks purrr::lift()  
## ✖ Matrix::pack() masks tidyr::pack()  
## ✖ yardstick::precision() masks caret::precision()  
## ✖ dials::prune() masks rpart::prune()  
## ✖ yardstick::recall() masks caret::recall()  
## ✖ MASS::select() masks dplyr::select()  
## ✖ yardstick::sensitivity() masks caret::sensitivity()  
## ✖ xgboost::slice() masks dplyr::slice()  
## ✖ yardstick::spec() masks readr::spec()  
## ✖ yardstick::specificity() masks caret::specificity()  
## ✖ recipes::step() masks stats::step()  
## ✖ Matrix::unpack() masks tidyr::unpack()  
## ✖ recipes::update() masks Matrix::update(), stats::update()

library(rpart.plot)

## Warning: package 'rpart.plot' was built under R version 4.3.3

library(cluster)  
library(factoextra) # for visualizing clusters

## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa

# Load the Medical Premium Data

df <- read\_csv("~/Desktop/Medicalpremium.csv")

## Rows: 986 Columns: 11  
## ── Column specification ────────────────────────────────────────────────────────  
## Delimiter: ","  
## dbl (11): Age, Diabetes, BloodPressureProblems, AnyTransplants, AnyChronicDi...  
##   
## ℹ Use `spec()` to retrieve the full column specification for this data.  
## ℹ Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

# Creation of a New Variable

df <- df %>%  
 mutate(BMI = Weight/((Height/100)^2))   
  
# view(df)  
glimpse(df)

## Rows: 986  
## Columns: 12  
## $ Age <dbl> 45, 60, 36, 52, 38, 30, 33, 23, 48, 38, 60, 66…  
## $ Diabetes <dbl> 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0…  
## $ BloodPressureProblems <dbl> 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0…  
## $ AnyTransplants <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0…  
## $ AnyChronicDiseases <dbl> 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…  
## $ Height <dbl> 155, 180, 158, 183, 166, 160, 150, 181, 169, 1…  
## $ Weight <dbl> 57, 73, 59, 93, 88, 69, 54, 79, 74, 93, 74, 67…  
## $ KnownAllergies <dbl> 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1…  
## $ HistoryOfCancerInFamily <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…  
## $ NumberOfMajorSurgeries <dbl> 0, 0, 1, 2, 1, 1, 0, 0, 0, 0, 2, 0, 1, 0, 1, 1…  
## $ PremiumPrice <dbl> 25000, 29000, 23000, 28000, 23000, 23000, 2100…  
## $ BMI <dbl> 23.72529, 22.53086, 23.63403, 27.77031, 31.934…

head(df)

## # A tibble: 6 × 12  
## Age Diabetes BloodPressureProblems AnyTransplants AnyChronicDiseases Height  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 45 0 0 0 0 155  
## 2 60 1 0 0 0 180  
## 3 36 1 1 0 0 158  
## 4 52 1 1 0 1 183  
## 5 38 0 0 0 1 166  
## 6 30 0 0 0 0 160  
## # ℹ 6 more variables: Weight <dbl>, KnownAllergies <dbl>,  
## # HistoryOfCancerInFamily <dbl>, NumberOfMajorSurgeries <dbl>,  
## # PremiumPrice <dbl>, BMI <dbl>

#Summary of the data (checking if any NA)  
summary(df)

## Age Diabetes BloodPressureProblems AnyTransplants   
## Min. :18.00 Min. :0.0000 Min. :0.0000 Min. :0.00000   
## 1st Qu.:30.00 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.00000   
## Median :42.00 Median :0.0000 Median :0.0000 Median :0.00000   
## Mean :41.75 Mean :0.4199 Mean :0.4686 Mean :0.05578   
## 3rd Qu.:53.00 3rd Qu.:1.0000 3rd Qu.:1.0000 3rd Qu.:0.00000   
## Max. :66.00 Max. :1.0000 Max. :1.0000 Max. :1.00000   
## AnyChronicDiseases Height Weight KnownAllergies   
## Min. :0.0000 Min. :145.0 Min. : 51.00 Min. :0.000   
## 1st Qu.:0.0000 1st Qu.:161.0 1st Qu.: 67.00 1st Qu.:0.000   
## Median :0.0000 Median :168.0 Median : 75.00 Median :0.000   
## Mean :0.1805 Mean :168.2 Mean : 76.95 Mean :0.215   
## 3rd Qu.:0.0000 3rd Qu.:176.0 3rd Qu.: 87.00 3rd Qu.:0.000   
## Max. :1.0000 Max. :188.0 Max. :132.00 Max. :1.000   
## HistoryOfCancerInFamily NumberOfMajorSurgeries PremiumPrice BMI   
## Min. :0.0000 Min. :0.0000 Min. :15000 Min. :15.16   
## 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:21000 1st Qu.:23.39   
## Median :0.0000 Median :1.0000 Median :23000 Median :27.16   
## Mean :0.1176 Mean :0.6673 Mean :24337 Mean :27.46   
## 3rd Qu.:0.0000 3rd Qu.:1.0000 3rd Qu.:28000 3rd Qu.:30.76   
## Max. :1.0000 Max. :3.0000 Max. :40000 Max. :50.00

sapply(df, class)

## Age Diabetes BloodPressureProblems   
## "numeric" "numeric" "numeric"   
## AnyTransplants AnyChronicDiseases Height   
## "numeric" "numeric" "numeric"   
## Weight KnownAllergies HistoryOfCancerInFamily   
## "numeric" "numeric" "numeric"   
## NumberOfMajorSurgeries PremiumPrice BMI   
## "numeric" "numeric" "numeric"

sum(is.na(df))

## [1] 0

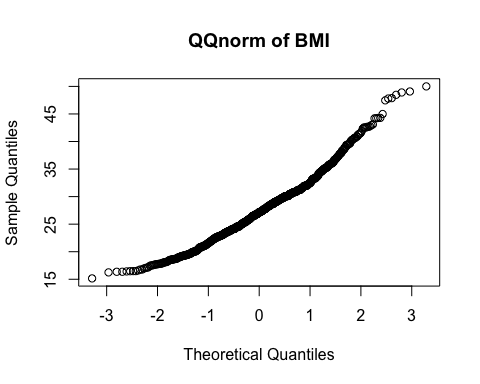
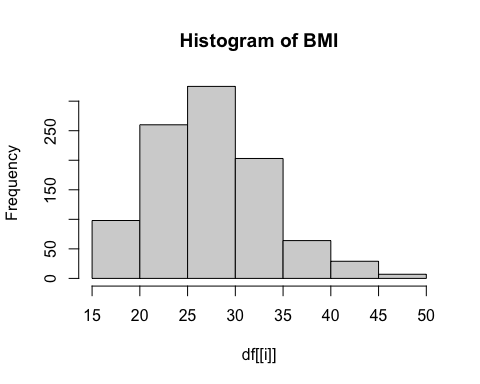
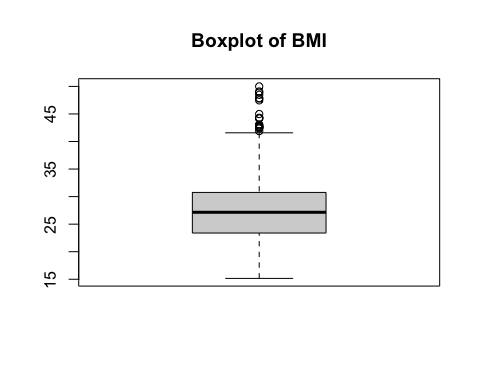
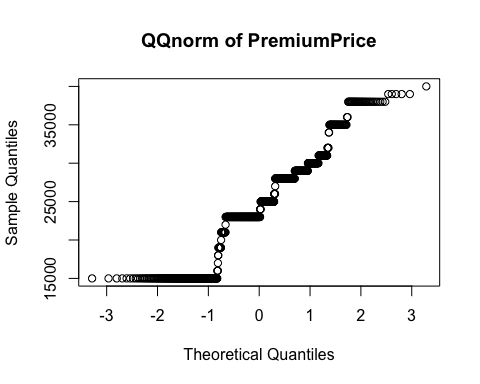
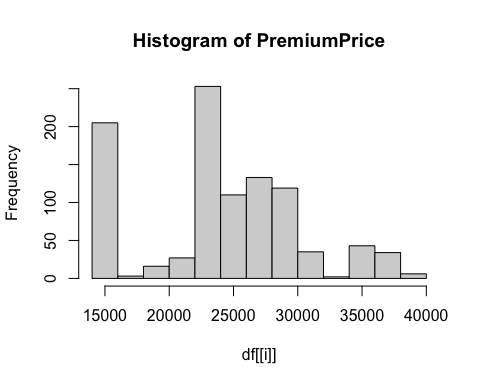
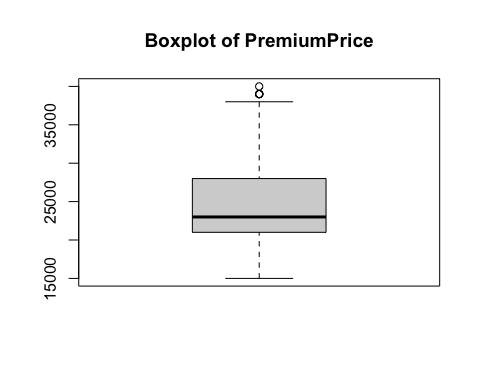
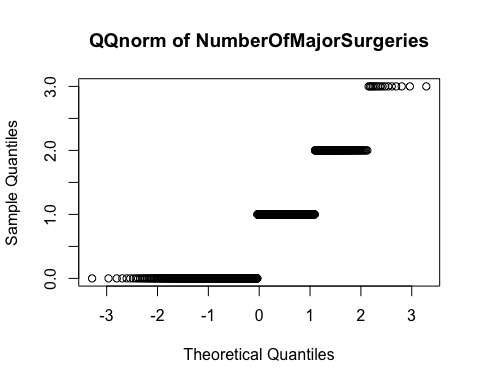
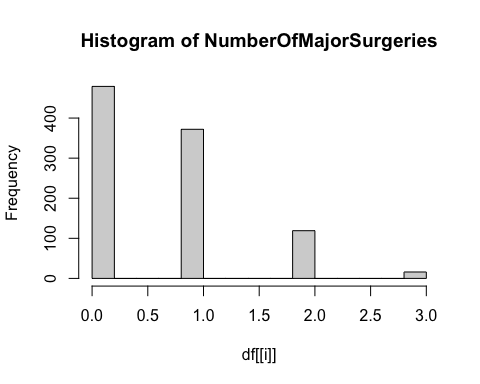
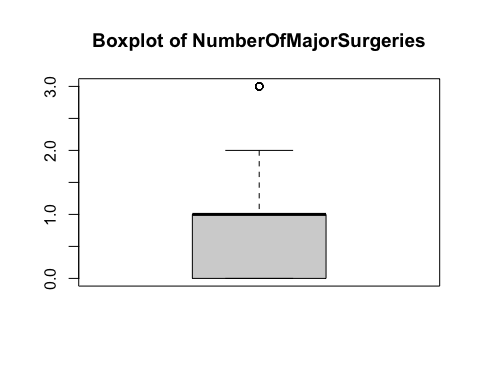
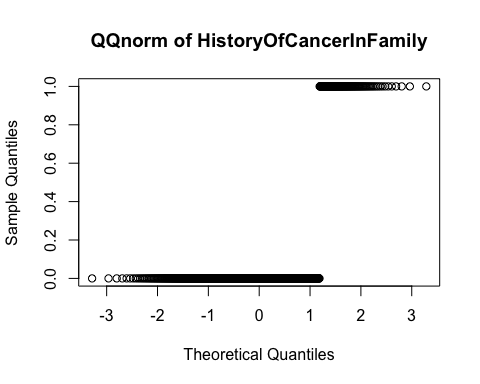
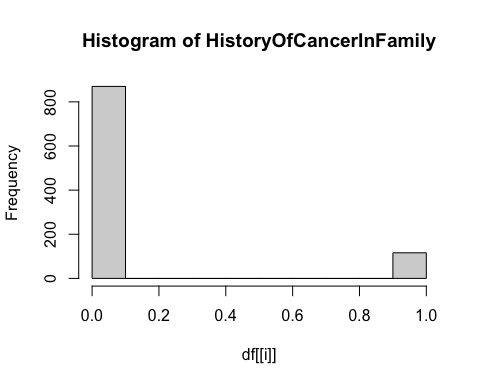
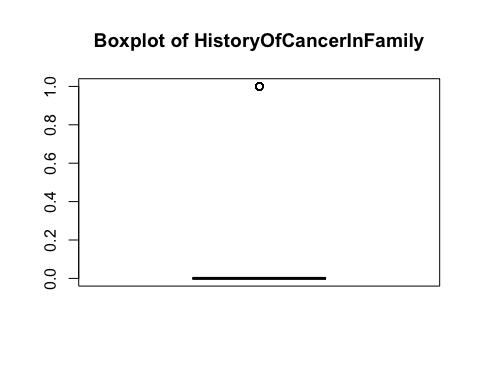
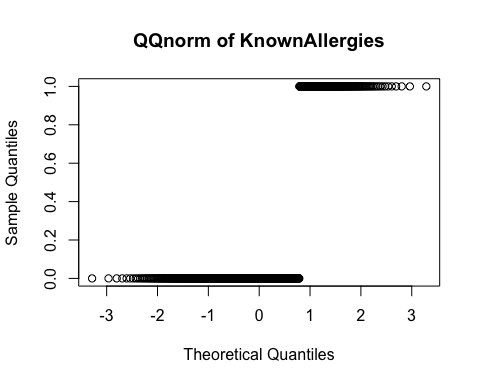
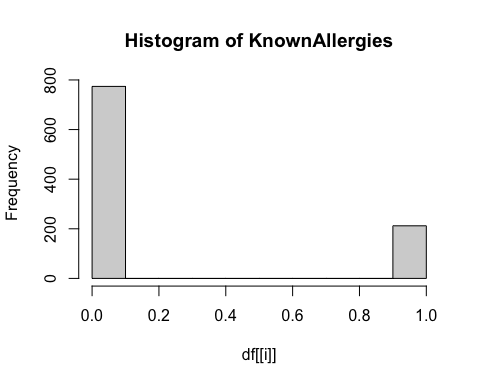
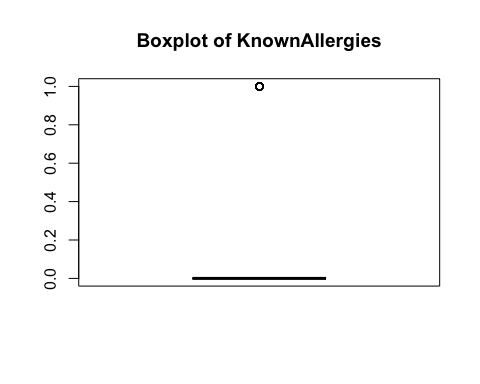
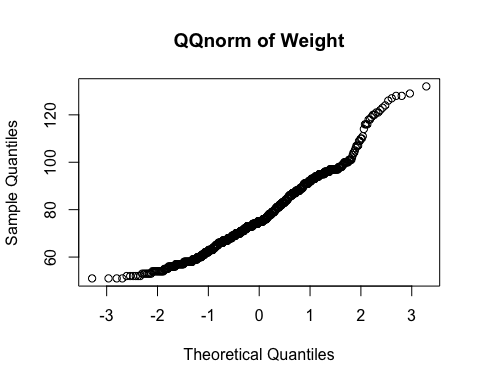
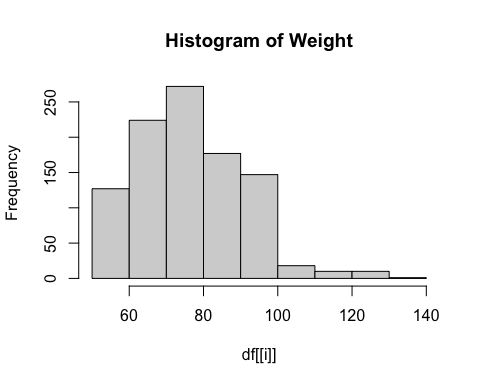
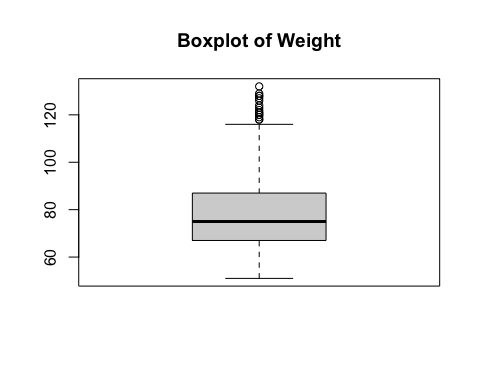
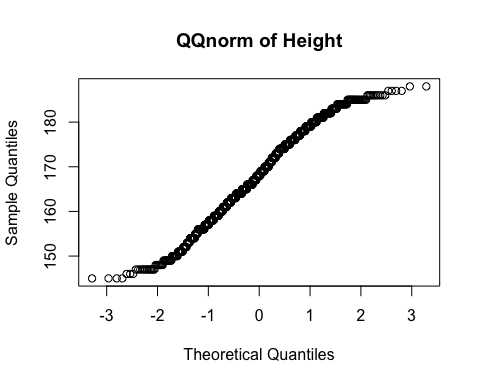
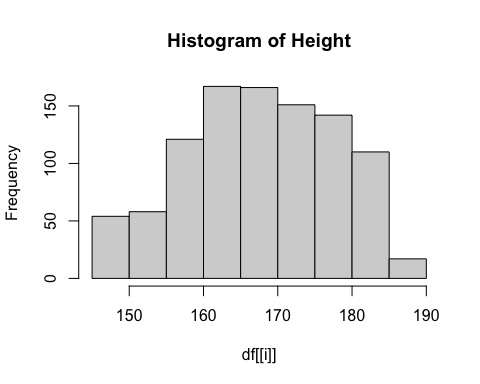
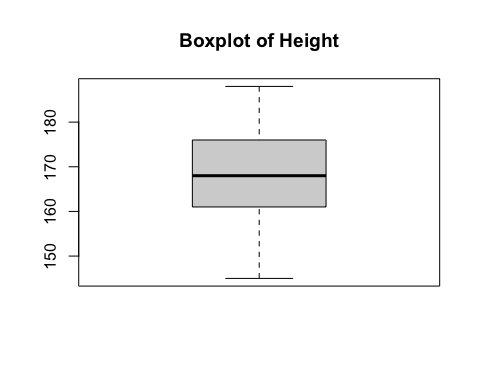
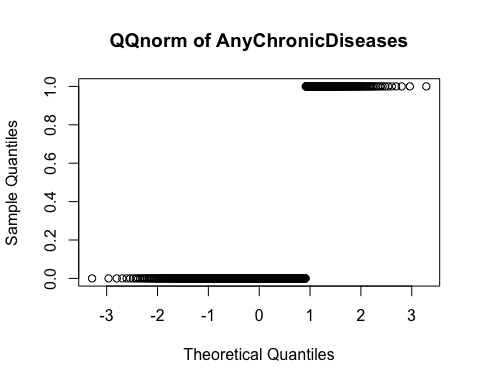
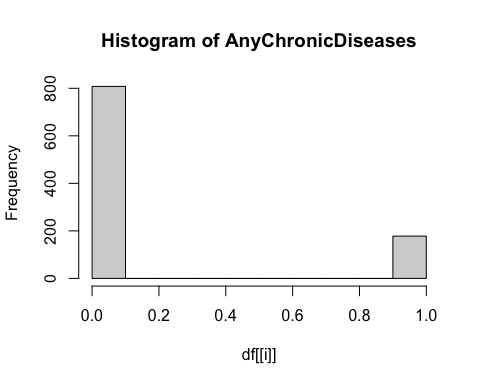
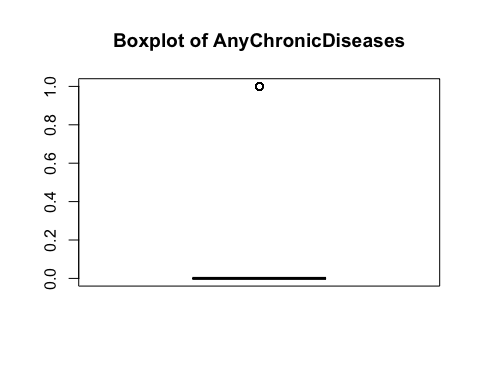
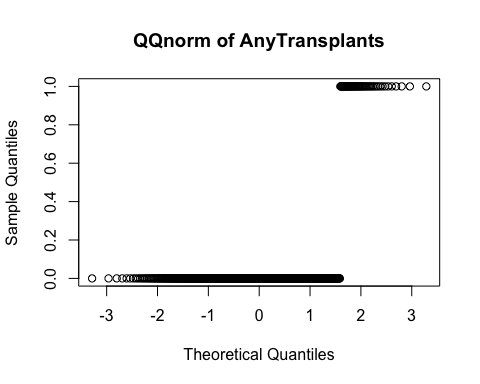
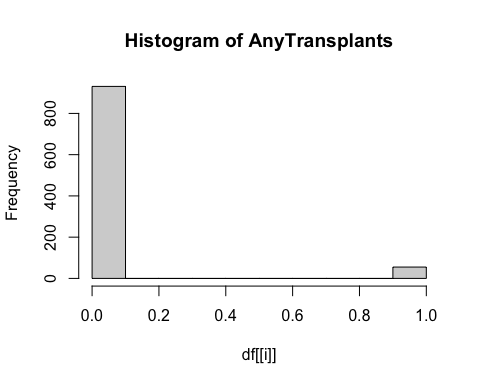
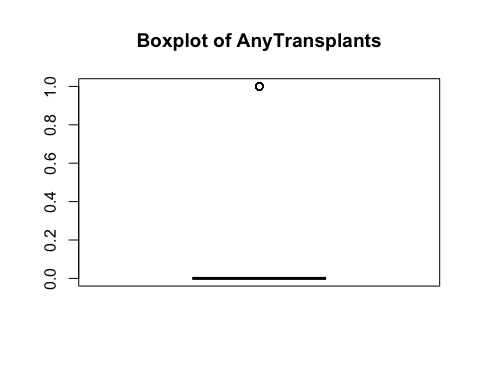
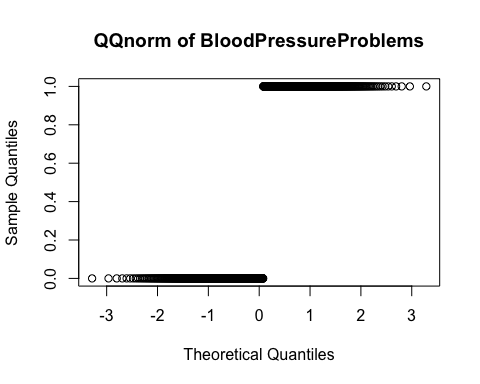
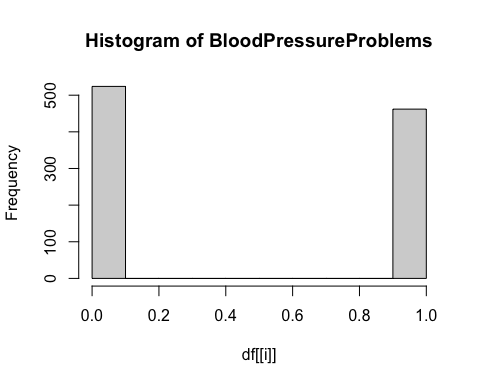
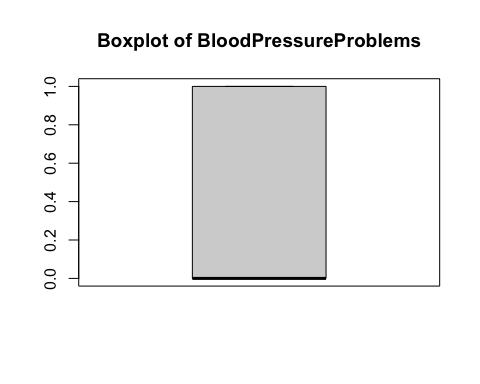
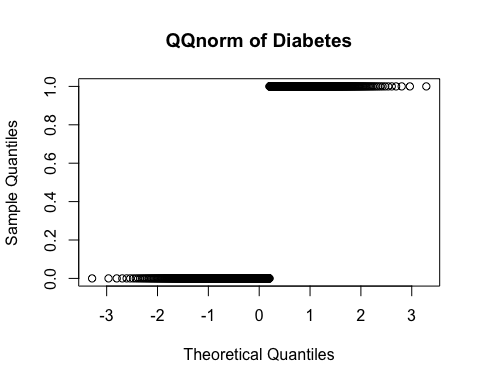
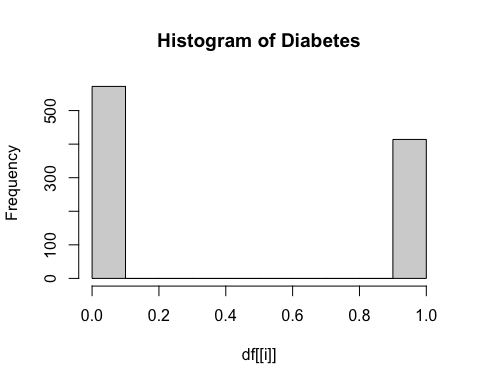
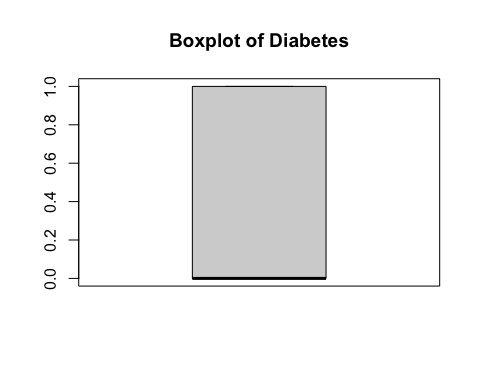
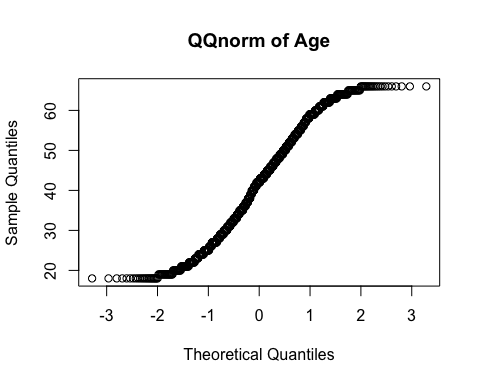
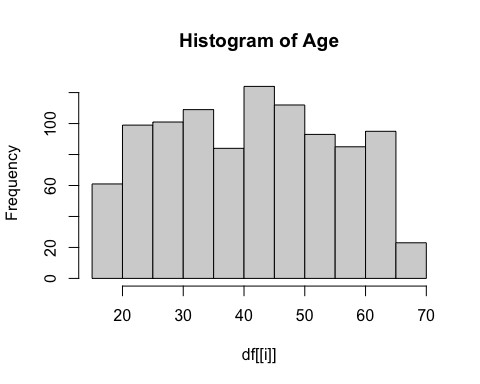
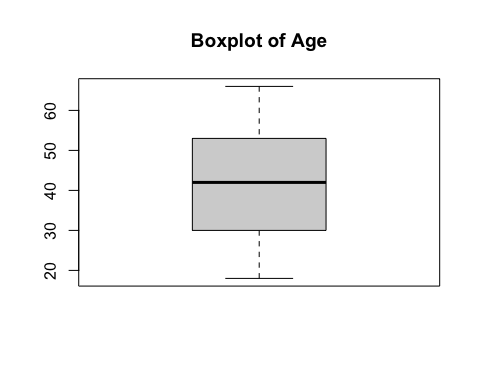
#ensuring that it is a numerical data  
df[] <- sapply(df, as.numeric)

There is no NA values

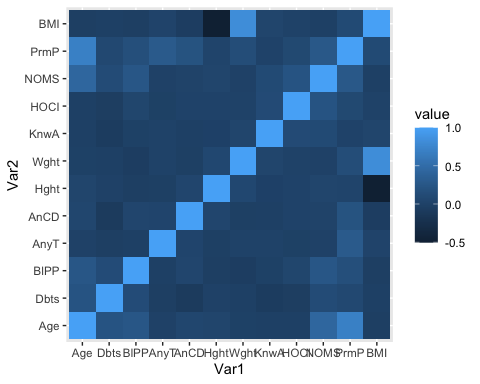
# Data Preprocessing

## Viewing the variables of the data using boxplot, qqnorms, correlation matrix, histogram and violen plots

#Boxplots and Histograms  
# Create lists to store plots  
boxplot\_list <- list()  
hist\_list <- list()  
qqnorm\_list <- list()  
  
for (i in 1:ncol(df)) {  
 column\_name <- names(df)[i]  
   
 # Create boxplot and histogram for each column  
 boxplot(df[[i]], main = paste("Boxplot of", column\_name))  
 hist(df[[i]], main = paste("Histogram of", column\_name))  
 qqnorm(df[[i]], main = paste("QQnorm of", column\_name))  
   
 # Save plots (optional - storing as plot objects or images)  
 boxplot\_list[[i]] <- recordPlot()  
 hist\_list[[i]] <- recordPlot()  
 qqnorm\_list[[i]] <- recordPlot()  
}

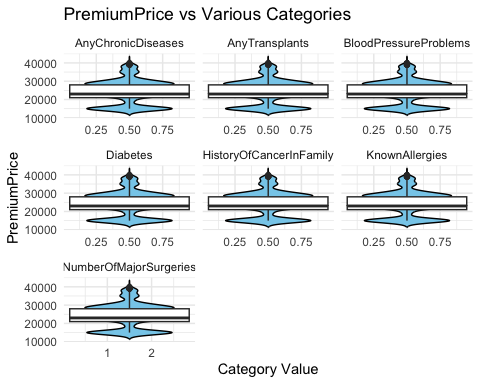


#Correlation matrix   
data <- cor(df[sapply(df,is.numeric)])  
data1 <- melt(data)  
  
ggplot(data1,aes(x= Var1, y=Var2,fill=value)) +  
 geom\_tile() +  
 scale\_x\_discrete(labels=abbreviate)+  
 scale\_y\_discrete(labels=abbreviate)



#Violin plots - categorical variables   
#Categorical variables include Diabetes, BloodPressureProblems, AnyTransplants, AnyChronicDiseases, KnownAllergies, HistoryOfCancerInFamily, NumberOfMajorSurgeries (x variables)  
#Premium price as y variable  
  
cat\_labels <- c("Diabetes", "BloodPressureProblems", "AnyTransplants", "AnyChronicDiseases", "KnownAllergies", "HistoryOfCancerInFamily", "NumberOfMajorSurgeries")  
  
df\_long <- df %>%  
 pivot\_longer(cols = all\_of(cat\_labels), names\_to = "x\_var", values\_to = "x\_value")  
  
  
ggplot(df\_long, aes(x = x\_value, y = PremiumPrice)) +  
 geom\_violin(trim = FALSE, fill = "skyblue", color = "black") +  
 geom\_boxplot(width = 0.1, fill = "white") +  
 facet\_wrap(~ x\_var, scales = "free\_x") +  
 theme\_minimal() +  
 labs(title = "PremiumPrice vs Various Categories", x = "Category Value", y = "PremiumPrice")

## Warning: Continuous x aesthetic  
## ℹ did you forget `aes(group = ...)`?



**1. QQ Plots (Quantile-Quantile Plots)**

Purpose: Check whether each variable (e.g., Age, Weight, PremiumPrice) follows a normal distribution.

What We Observed: Most variables do not align perfectly with the diagonal line, meaning they are not normally distributed. For example, PremiumPrice is right-skewed, indicating outliers and a long tail of higher prices.

Business Interpretation:

* Health costs (premiums) vary widely, and this non-normality supports using nonlinear models (e.g., XGBoost).
* Stakeholders should be cautious using statistical methods assuming normality.

**2. Histograms**

Purpose: Visualize the distribution of each feature to spot skewness, outliers, or groupings.

Key Insights:

* PremiumPrice has a right-skewed distribution, confirming that most customers pay mid-range premiums, but a few pay much higher.
* BMI and Weight also show broad spread — useful for pricing based on body metrics.

Business Interpretation:

* Premium pricing needs to handle a wide range of cases — standardized pricing may not be fair or profitable.
* Emphasizes the need for tiered risk-based premiums.

**3. Violin Plots (PremiumPrice vs Categorical Features)**

Purpose: Show how PremiumPrice varies by binary health conditions, such as: Diabetes, Blood Pressure, Chronic Disease, Transplants, Allergies, etc.

What We Saw: Clear upward shifts in premium for those with: Chronic disease, Transplant history, Cancer in family, Allergies showed minimal effect.

Business Interpretation:

* Visual validation that risk-based pricing is justified.
* Helps actuaries identify which medical conditions drive costs up most.
* Supports development of health screening questionnaires.

**4. Correlation Matrix (Heatmap)**

Purpose: Explore linear relationships between all numeric variables.

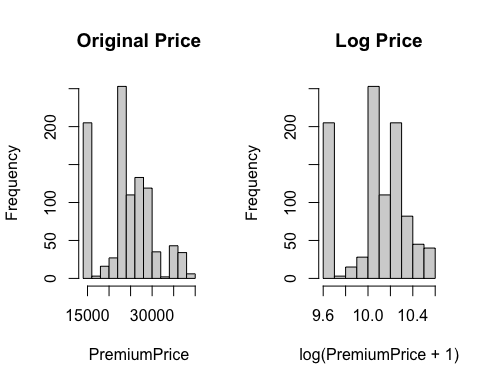
Observations: PremiumPrice is moderately correlated with: Age, Weight, Number of Surgeries, Chronic Disease, Correlations are not extremely high → supports use of flexible models.

Business Interpretation:

* No single factor dictates price → pricing models must consider multiple interactions.
* Feature engineering (e.g., Age × Weight) is important for predictive accuracy.

##Comparison to LogPremiumPrice and PremiumPrice

df\_log <- df  
df\_log$logPrice <- log(df$PremiumPrice + 1)  
# Check histogram of the original and log price  
par(mfrow = c(1, 2))  
hist(df\_log$PremiumPrice, main = "Original Price", xlab = "PremiumPrice")  
hist(log(df\_log$PremiumPrice + 1), main = "Log Price", xlab = "log(PremiumPrice + 1)")



The results:

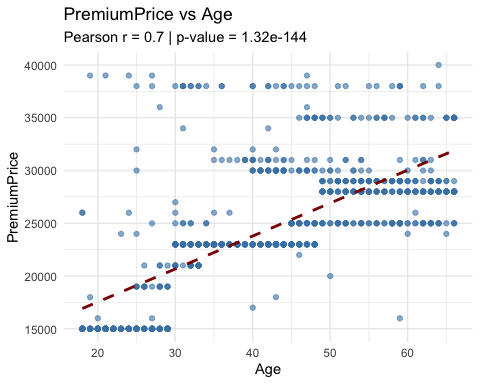
* Original PremiumPrice (left plot): The distribution is right-skewed, with a sharp peak around 20,000 and a long tail toward higher prices.
* Log(PremiumPrice + 1) (right plot): The distribution is much more symmetric and closer to normal — which is typically better for many regression models.

## Scatter plot with hypothesis test (Pearson correlation)

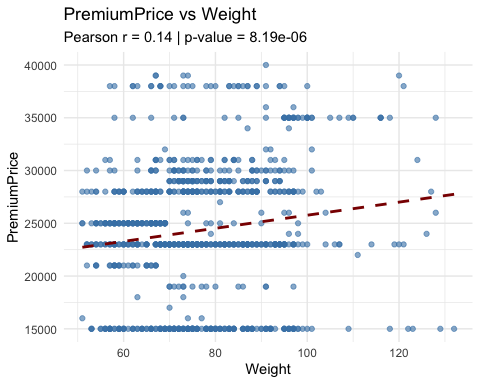
#Discrete variables include Age, Height, Weight, BMI (x variables)   
#Premium price as y variable   
  
num\_labels <- c("Age", "Weight", "Height", "BMI")  
  
for (x in num\_labels) {  
   
 # Run Pearson correlation test  
 test <- cor.test(df[[x]], df$PremiumPrice, method = "pearson")  
 r <- round(test$estimate, 2)  
 p <- signif(test$p.value, 3)  
   
 #Scatter Plot  
 print(  
 ggplot(df, aes\_string(x = x, y = "PremiumPrice")) +  
 geom\_point(color = "steelblue", alpha = 0.6) +  
 geom\_smooth(method = "lm", se = FALSE, color = "darkred", linetype = "dashed") +  
 theme\_minimal() +  
 labs(  
 title = paste("PremiumPrice vs", x),  
 subtitle = paste("Pearson r =", r, "| p-value =", p),  
 x = x,  
 y = "PremiumPrice"  
 )  
 )  
}

## Warning: `aes\_string()` was deprecated in ggplot2 3.0.0.  
## ℹ Please use tidy evaluation idioms with `aes()`.  
## ℹ See also `vignette("ggplot2-in-packages")` for more information.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.

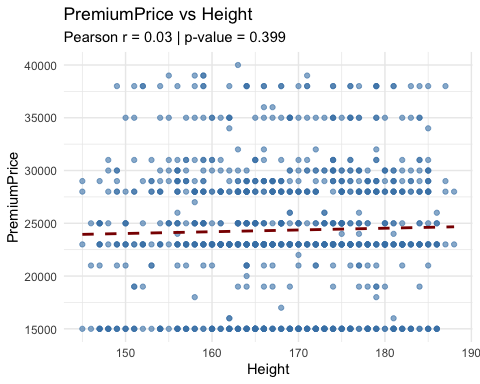
## `geom\_smooth()` using formula = 'y ~ x'



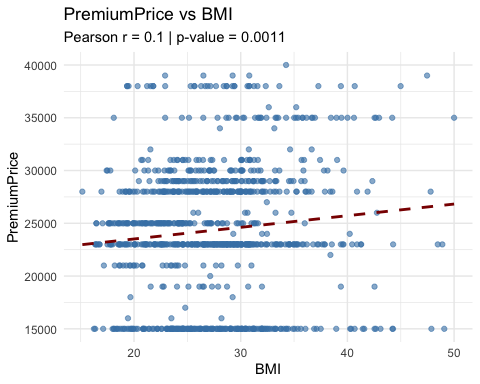
## `geom\_smooth()` using formula = 'y ~ x'



## `geom\_smooth()` using formula = 'y ~ x'



## `geom\_smooth()` using formula = 'y ~ x'



## Conducting T-test and ANOVA test to quantity if the difference is significant

for (x in cat\_labels) {  
 df[[x]] <- as.factor(df[[x]]) # ensure it's a factor  
 formula <- as.formula(paste("PremiumPrice ~", x))  
   
 if (length(unique(df[[x]])) == 2) {  
 test <- t.test(formula, data = df)  
 cat("\n", x, "- t-test\n")  
 } else {  
 test <- aov(formula, data = df)  
 cat("\n", x, "- ANOVA\n")  
 print(summary(test))  
 next  
 }  
   
 print(test)  
}

##   
## Diabetes - t-test  
##   
## Welch Two Sample t-test  
##   
## data: PremiumPrice by Diabetes  
## t = -2.4489, df = 949.09, p-value = 0.01451  
## alternative hypothesis: true difference in means between group 0 and group 1 is not equal to 0  
## 95 percent confidence interval:  
## -1737.0795 -191.5546  
## sample estimates:  
## mean in group 0 mean in group 1   
## 23931.82 24896.14   
##   
##   
## BloodPressureProblems - t-test  
##   
## Welch Two Sample t-test  
##   
## data: PremiumPrice by BloodPressureProblems  
## t = -5.3703, df = 982.76, p-value = 9.813e-08  
## alternative hypothesis: true difference in means between group 0 and group 1 is not equal to 0  
## 95 percent confidence interval:  
## -2855.331 -1327.033  
## sample estimates:  
## mean in group 0 mean in group 1   
## 23356.87 25448.05   
##   
##   
## AnyTransplants - t-test  
##   
## Welch Two Sample t-test  
##   
## data: PremiumPrice by AnyTransplants  
## t = -6.2599, df = 56.504, p-value = 5.545e-08  
## alternative hypothesis: true difference in means between group 0 and group 1 is not equal to 0  
## 95 percent confidence interval:  
## -10382.274 -5349.081  
## sample estimates:  
## mean in group 0 mean in group 1   
## 23897.96 31763.64   
##   
##   
## AnyChronicDiseases - t-test  
##   
## Welch Two Sample t-test  
##   
## data: PremiumPrice by AnyChronicDiseases  
## t = -7.7077, df = 311.78, p-value = 1.728e-13  
## alternative hypothesis: true difference in means between group 0 and group 1 is not equal to 0  
## 95 percent confidence interval:  
## -4251.769 -2522.455  
## sample estimates:  
## mean in group 0 mean in group 1   
## 23725.25 27112.36   
##   
##   
## KnownAllergies - t-test  
##   
## Welch Two Sample t-test  
##   
## data: PremiumPrice by KnownAllergies  
## t = -0.36669, df = 320.56, p-value = 0.7141  
## alternative hypothesis: true difference in means between group 0 and group 1 is not equal to 0  
## 95 percent confidence interval:  
## -1171.0486 803.0997  
## sample estimates:  
## mean in group 0 mean in group 1   
## 24297.16 24481.13   
##   
##   
## HistoryOfCancerInFamily - t-test  
##   
## Welch Two Sample t-test  
##   
## data: PremiumPrice by HistoryOfCancerInFamily  
## t = -2.3568, df = 139.3, p-value = 0.01983  
## alternative hypothesis: true difference in means between group 0 and group 1 is not equal to 0  
## 95 percent confidence interval:  
## -2963.4186 -259.5699  
## sample estimates:  
## mean in group 0 mean in group 1   
## 24147.13 25758.62   
##   
##   
## NumberOfMajorSurgeries - ANOVA  
## Df Sum Sq Mean Sq F value Pr(>F)   
## NumberOfMajorSurgeries 3 2.843e+09 947765071 26.14 2.87e-16 \*\*\*  
## Residuals 982 3.561e+10 36263662   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

The results shows:

* Diabetes is associated with higher premiums.
* Blood pressure issues are strongly linked to higher premiums.
* Transplant history is highly associated with much higher premiums.
* Chronic disease status is strongly linked to increased premiums.
* Allergies do not significantly impact PremiumPrice.
* Family cancer history is linked to slightly higher premiums.

More analysis: do a stepwise regression (forward and combination of forward and backwards), then remove the one that is not significant. Then we look at the multi-collinearity.

We standardized the model using the log transformation where In(Price) for the linear regression and regularized models.

## Stepwise regression for feature selection

intercept\_only <- lm(PremiumPrice ~ 1, data=df)  
  
allVars <- lm(PremiumPrice ~., data=df)  
  
  
#forward stepwise regression  
forward <- stats::step(intercept\_only, direction='forward', scope=formula(allVars), trace=0)  
forward

##   
## Call:  
## lm(formula = PremiumPrice ~ Age + AnyTransplants + AnyChronicDiseases +   
## Weight + NumberOfMajorSurgeries + HistoryOfCancerInFamily,   
## data = df)  
##   
## Coefficients:  
## (Intercept) Age AnyTransplants1   
## 3914.37 334.56 7866.76   
## AnyChronicDiseases1 Weight NumberOfMajorSurgeries1   
## 2686.40 71.14 253.61   
## NumberOfMajorSurgeries2 NumberOfMajorSurgeries3 HistoryOfCancerInFamily1   
## -1672.82 -3261.89 1844.16

summary(forward)

##   
## Call:  
## lm(formula = PremiumPrice ~ Age + AnyTransplants + AnyChronicDiseases +   
## Weight + NumberOfMajorSurgeries + HistoryOfCancerInFamily,   
## data = df)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -13143.5 -2181.2 -271.3 1939.5 23708.9   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 3914.370 771.100 5.076 4.61e-07 \*\*\*  
## Age 334.559 9.726 34.400 < 2e-16 \*\*\*  
## AnyTransplants1 7866.756 517.918 15.189 < 2e-16 \*\*\*  
## AnyChronicDiseases1 2686.400 310.249 8.659 < 2e-16 \*\*\*  
## Weight 71.142 8.338 8.533 < 2e-16 \*\*\*  
## NumberOfMajorSurgeries1 253.612 283.545 0.894 0.371312   
## NumberOfMajorSurgeries2 -1672.823 429.444 -3.895 0.000105 \*\*\*  
## NumberOfMajorSurgeries3 -3261.892 981.706 -3.323 0.000925 \*\*\*  
## HistoryOfCancerInFamily1 1844.163 403.509 4.570 5.49e-06 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 3727 on 977 degrees of freedom  
## Multiple R-squared: 0.6471, Adjusted R-squared: 0.6442   
## F-statistic: 224 on 8 and 977 DF, p-value: < 2.2e-16

intercept\_only\_log <- lm(logPrice ~ 1, data=df\_log)  
  
allVars\_log <- lm(logPrice ~., data=df\_log)  
forward\_log <- stats::step(intercept\_only, direction='forward', scope=formula(allVars), trace=0)  
forward\_log

##   
## Call:  
## lm(formula = PremiumPrice ~ Age + AnyTransplants + AnyChronicDiseases +   
## Weight + NumberOfMajorSurgeries + HistoryOfCancerInFamily,   
## data = df)  
##   
## Coefficients:  
## (Intercept) Age AnyTransplants1   
## 3914.37 334.56 7866.76   
## AnyChronicDiseases1 Weight NumberOfMajorSurgeries1   
## 2686.40 71.14 253.61   
## NumberOfMajorSurgeries2 NumberOfMajorSurgeries3 HistoryOfCancerInFamily1   
## -1672.82 -3261.89 1844.16

summary(forward\_log)

##   
## Call:  
## lm(formula = PremiumPrice ~ Age + AnyTransplants + AnyChronicDiseases +   
## Weight + NumberOfMajorSurgeries + HistoryOfCancerInFamily,   
## data = df)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -13143.5 -2181.2 -271.3 1939.5 23708.9   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 3914.370 771.100 5.076 4.61e-07 \*\*\*  
## Age 334.559 9.726 34.400 < 2e-16 \*\*\*  
## AnyTransplants1 7866.756 517.918 15.189 < 2e-16 \*\*\*  
## AnyChronicDiseases1 2686.400 310.249 8.659 < 2e-16 \*\*\*  
## Weight 71.142 8.338 8.533 < 2e-16 \*\*\*  
## NumberOfMajorSurgeries1 253.612 283.545 0.894 0.371312   
## NumberOfMajorSurgeries2 -1672.823 429.444 -3.895 0.000105 \*\*\*  
## NumberOfMajorSurgeries3 -3261.892 981.706 -3.323 0.000925 \*\*\*  
## HistoryOfCancerInFamily1 1844.163 403.509 4.570 5.49e-06 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 3727 on 977 degrees of freedom  
## Multiple R-squared: 0.6471, Adjusted R-squared: 0.6442   
## F-statistic: 224 on 8 and 977 DF, p-value: < 2.2e-16

#forward stepwise regression  
forward <- stats::step(intercept\_only, direction='forward', scope=formula(allVars), trace=0)  
forward

##   
## Call:  
## lm(formula = PremiumPrice ~ Age + AnyTransplants + AnyChronicDiseases +   
## Weight + NumberOfMajorSurgeries + HistoryOfCancerInFamily,   
## data = df)  
##   
## Coefficients:  
## (Intercept) Age AnyTransplants1   
## 3914.37 334.56 7866.76   
## AnyChronicDiseases1 Weight NumberOfMajorSurgeries1   
## 2686.40 71.14 253.61   
## NumberOfMajorSurgeries2 NumberOfMajorSurgeries3 HistoryOfCancerInFamily1   
## -1672.82 -3261.89 1844.16

#Mixture of both forward and backward selection  
  
both <- stats::step(intercept\_only, direction='both', scope=formula(allVars), trace=0)  
  
summary(both)

##   
## Call:  
## lm(formula = PremiumPrice ~ Age + AnyTransplants + AnyChronicDiseases +   
## Weight + NumberOfMajorSurgeries + HistoryOfCancerInFamily,   
## data = df)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -13143.5 -2181.2 -271.3 1939.5 23708.9   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 3914.370 771.100 5.076 4.61e-07 \*\*\*  
## Age 334.559 9.726 34.400 < 2e-16 \*\*\*  
## AnyTransplants1 7866.756 517.918 15.189 < 2e-16 \*\*\*  
## AnyChronicDiseases1 2686.400 310.249 8.659 < 2e-16 \*\*\*  
## Weight 71.142 8.338 8.533 < 2e-16 \*\*\*  
## NumberOfMajorSurgeries1 253.612 283.545 0.894 0.371312   
## NumberOfMajorSurgeries2 -1672.823 429.444 -3.895 0.000105 \*\*\*  
## NumberOfMajorSurgeries3 -3261.892 981.706 -3.323 0.000925 \*\*\*  
## HistoryOfCancerInFamily1 1844.163 403.509 4.570 5.49e-06 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 3727 on 977 degrees of freedom  
## Multiple R-squared: 0.6471, Adjusted R-squared: 0.6442   
## F-statistic: 224 on 8 and 977 DF, p-value: < 2.2e-16

Due to the forward stepwise regression, the following selected variables are: Age, AnyTransplants, AnyChronicDiseases, Weight, HistoryofCancerInFamily, NumberofMajorSurgeries.

Forward selection had the same results when using either premium price or log transformed premium price as a target.

df\_new <- df |>  
 dplyr::select(PremiumPrice, Age, AnyTransplants, AnyChronicDiseases, Weight, HistoryOfCancerInFamily, NumberOfMajorSurgeries)  
  
#ensuring that it is a numerical data  
df\_new[] <- sapply(df\_new, as.numeric)

# Regression Analysis (Baseline, Interaction, Lasso, Ridge) - need to be updated to logPremiumPrice and the df\_regression

#New dataframe for regression models using the logPremiumPrice   
df\_regression <- df\_log %>%  
 dplyr::select(logPrice, Age, AnyTransplants, AnyChronicDiseases, Weight, HistoryOfCancerInFamily, NumberOfMajorSurgeries)  
  
#Sort out the remaining linear regression equations using the df\_regression   
no\_int <- aov(logPrice ~ Age + Weight , data = df\_regression)  
  
interaction <- aov(logPrice ~ Age + Weight + Age:Weight, data = df\_regression)  
  
no\_int

## Call:  
## aov(formula = logPrice ~ Age + Weight, data = df\_regression)  
##   
## Terms:  
## Age Weight Residuals  
## Sum of Squares 39.67477 1.15014 31.57187  
## Deg. of Freedom 1 1 983  
##   
## Residual standard error: 0.1792146  
## Estimated effects may be unbalanced

interaction

## Call:  
## aov(formula = logPrice ~ Age + Weight + Age:Weight, data = df\_regression)  
##   
## Terms:  
## Age Weight Age:Weight Residuals  
## Sum of Squares 39.67477 1.15014 0.32378 31.24809  
## Deg. of Freedom 1 1 1 982  
##   
## Residual standard error: 0.178384  
## Estimated effects may be unbalanced

#2  
  
summary(no\_int)

## Df Sum Sq Mean Sq F value Pr(>F)   
## Age 1 39.67 39.67 1235.29 < 2e-16 \*\*\*  
## Weight 1 1.15 1.15 35.81 3.04e-09 \*\*\*  
## Residuals 983 31.57 0.03   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

summary(interaction)

## Df Sum Sq Mean Sq F value Pr(>F)   
## Age 1 39.67 39.67 1246.82 < 2e-16 \*\*\*  
## Weight 1 1.15 1.15 36.14 2.58e-09 \*\*\*  
## Age:Weight 1 0.32 0.32 10.18 0.00147 \*\*   
## Residuals 982 31.25 0.03   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

#3  
  
summary(lm(logPrice ~ Age+ Weight + Age:Weight, data = df\_regression))

##   
## Call:  
## lm(formula = logPrice ~ Age + Weight + Age:Weight, data = df\_regression)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.53252 -0.12782 -0.01096 0.08313 0.83733   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 9.565e+00 9.669e-02 98.927 < 2e-16 \*\*\*  
## Age 7.553e-03 2.191e-03 3.448 0.000589 \*\*\*  
## Weight -1.303e-03 1.226e-03 -1.063 0.288203   
## Age:Weight 8.874e-05 2.782e-05 3.190 0.001469 \*\*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.1784 on 982 degrees of freedom  
## Multiple R-squared: 0.5684, Adjusted R-squared: 0.5671   
## F-statistic: 431 on 3 and 982 DF, p-value: < 2.2e-16

summary(lm(logPrice ~ Age + Weight, data = df\_regression))

##   
## Call:  
## lm(formula = logPrice ~ Age + Weight, data = df\_regression)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.57081 -0.12918 -0.01055 0.08249 0.85853   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 9.2783344 0.0359558 258.049 < 2e-16 \*\*\*  
## Age 0.0144185 0.0004090 35.252 < 2e-16 \*\*\*  
## Weight 0.0023958 0.0004004 5.984 3.04e-09 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.1792 on 983 degrees of freedom  
## Multiple R-squared: 0.5639, Adjusted R-squared: 0.563   
## F-statistic: 635.5 on 2 and 983 DF, p-value: < 2.2e-16

#4  
  
categorical <- lm(logPrice~.,data=df\_regression)  
  
categorical

##   
## Call:  
## lm(formula = logPrice ~ ., data = df\_regression)  
##   
## Coefficients:  
## (Intercept) Age AnyTransplants   
## 9.218850 0.014962 0.261872   
## AnyChronicDiseases Weight HistoryOfCancerInFamily   
## 0.123655 0.002492 0.088991   
## NumberOfMajorSurgeries   
## -0.026997

summary(categorical)

##   
## Call:  
## lm(formula = logPrice ~ ., data = df\_regression)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.54829 -0.10244 -0.00912 0.09323 0.92823   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 9.2188505 0.0322125 286.188 < 2e-16 \*\*\*  
## Age 0.0149616 0.0004058 36.865 < 2e-16 \*\*\*  
## AnyTransplants 0.2618718 0.0220759 11.862 < 2e-16 \*\*\*  
## AnyChronicDiseases 0.1236553 0.0131944 9.372 < 2e-16 \*\*\*  
## Weight 0.0024924 0.0003553 7.016 4.27e-12 \*\*\*  
## HistoryOfCancerInFamily 0.0889912 0.0162312 5.483 5.33e-08 \*\*\*  
## NumberOfMajorSurgeries -0.0269974 0.0077272 -3.494 0.000497 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.1589 on 979 degrees of freedom  
## Multiple R-squared: 0.6584, Adjusted R-squared: 0.6563   
## F-statistic: 314.5 on 6 and 979 DF, p-value: < 2.2e-16

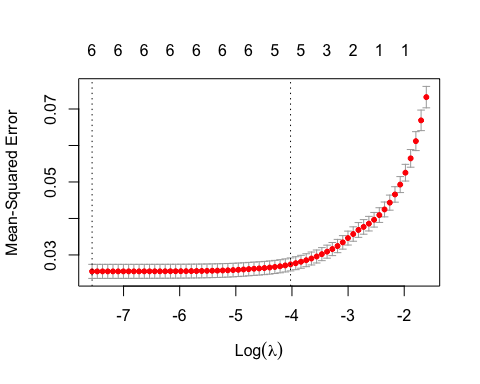
#Lasso regression model  
  
y <- df\_regression$logPrice  
  
X <- data.matrix(df\_regression[,c("Age","AnyTransplants","Weight","AnyChronicDiseases","HistoryOfCancerInFamily","NumberOfMajorSurgeries")])  
  
#Finding optimal lambda value  
  
cross\_validation <- cv.glmnet(X,y,alpha=1)  
cross\_validation

##   
## Call: cv.glmnet(x = X, y = y, alpha = 1)   
##   
## Measure: Mean-Squared Error   
##   
## Lambda Index Measure SE Nonzero  
## min 0.000521 65 0.0255 0.001927 6  
## 1se 0.017857 27 0.0274 0.001778 5

min\_mse <- cross\_validation$lambda.min  
min\_mse

## [1] 0.0005205472

plot(cross\_validation)



#final\_model  
  
final\_model <- glmnet(X,y,alpha=1,lambda=min\_mse)  
coef(final\_model)

## 7 x 1 sparse Matrix of class "dgCMatrix"  
## s0  
## (Intercept) 9.224470905  
## Age 0.014888649  
## AnyTransplants 0.259594630  
## Weight 0.002454203  
## AnyChronicDiseases 0.122412203  
## HistoryOfCancerInFamily 0.086528084  
## NumberOfMajorSurgeries -0.025491272

#calculating r-squared  
predicitons <- predict(final\_model, s=min\_mse,newx=X)  
  
sst <- sum((y-mean(y))^2)  
sse <- sum((predicitons-y)^2)  
  
rsq <- 1 - sse/sst  
  
rsq

## [1] 0.6583464

#ridge regression model  
  
ridge\_model <- glmnet(X,y,alpha=0, lambda=min\_mse)  
coef(ridge\_model)

## 7 x 1 sparse Matrix of class "dgCMatrix"  
## s0  
## (Intercept) 9.220703029  
## Age 0.014923564  
## AnyTransplants 0.261354117  
## Weight 0.002486957  
## AnyChronicDiseases 0.123484357  
## HistoryOfCancerInFamily 0.088574406  
## NumberOfMajorSurgeries -0.026603954

ridge\_predicitons <- predict(ridge\_model, s=min\_mse,newx=X)  
  
ridge\_sst <- sum((y-mean(y))^2)  
ridge\_sse <- sum((ridge\_predicitons-y)^2)  
  
ridge\_rsq <- 1 - ridge\_sse/ridge\_sst  
  
ridge\_rsq

## [1] 0.6583747

**Ridge and Lasso regression**

The ridge regression model produced and R-Squared value of 0.658 indicating that approximately 65.8 percent in the variation in premium prices can be explained by the model.

The lasso regression model produced and R-Squared value of 0.658 indicating that approximately 65.8 percent in the variation in premium prices can be explained by the model.

# Machine Learning Models: Tree-Based Methods

We use the PremiumPrice variables instead of the logs because:

* Decision trees, and Xgboost model don’t assume normality or linearity, so they handle skewed distributions well.
* They split the feature space based on thresholds, not based on mean or variance, so log-transforming the target is not essential.
* The end goal was to predict predict and interpret PremiumPrice in dollars, and see which model is accurate in predicting the Premium Prices.

## Decision Tree

set.seed(12345)  
  
# Split into training and test sets (60/40 split)  
# Split into features and label  
y <- df\_new$PremiumPrice  
X <- df\_new %>% dplyr::select(-PremiumPrice)  
  
train\_indices <- sample(1:nrow(df), 0.6 \* nrow(df))  
  
# Create training and testing data frames for tidymodels  
train\_data <- df\_new[train\_indices, ]  
test\_data <- df\_new[-train\_indices, ]  
  
# Model specification  
tree\_spec <- decision\_tree() %>%  
 set\_engine("rpart") %>%  
 set\_mode("regression")  
  
# Model fitting  
tree\_fit <- tree\_spec %>%  
 fit(PremiumPrice ~ ., data = train\_data)  
  
# Predict on test data  
predictions <- predict(tree\_fit, new\_data = test\_data) %>%  
 pull(.pred)  
  
# Evaluate model performance  
results\_df <- test\_data %>%  
 mutate(predictions = predictions)  
  
# Compute RMSE  
rmse\_result <- rmse(results\_df, truth = PremiumPrice, estimate = predictions)  
  
# Compute MAE  
mae\_result <- mae(results\_df, truth = PremiumPrice, estimate = predictions)  
  
# Compute R-squared  
rsq\_result <- rsq(results\_df, truth = PremiumPrice, estimate = predictions)  
  
# Print results  
print(rmse\_result)

## # A tibble: 1 × 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 rmse standard 2954.

print(mae\_result)

## # A tibble: 1 × 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 mae standard 1713.

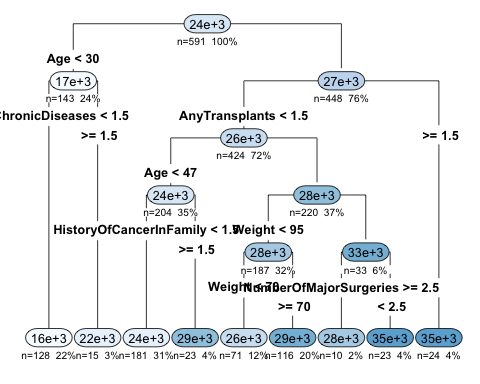
print(rsq\_result)

## # A tibble: 1 × 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 rsq standard 0.776

# Visualize the tree  
rpart.plot(tree\_fit$fit, type = 4, extra = 101, under = TRUE, cex = 0.8, box.palette = "auto")

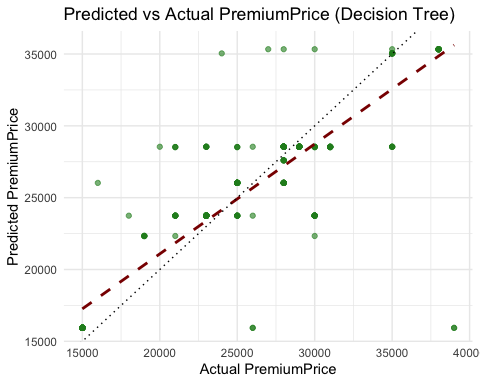
## Warning: Cannot retrieve the data used to build the model (model.frame: 'data' must be a data.frame, not a matrix or an array).  
## To silence this warning:  
## Call rpart.plot with roundint=FALSE,  
## or rebuild the rpart model with model=TRUE.

## Warning: labs do not fit even at cex 0.15, there may be some overplotting



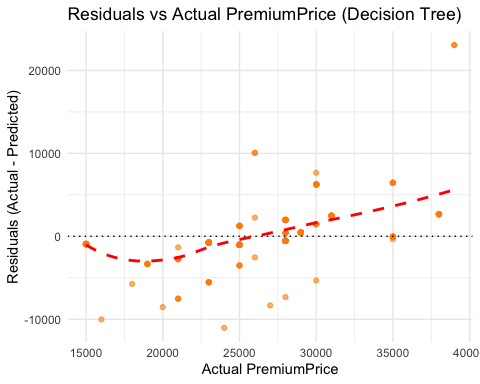
# Create a data frame with actual and predicted values  
results\_df\_tree <- data.frame(  
 Actual = test\_data$PremiumPrice,  
 Predicted = predictions  
)  
  
  
  
# Plot actual vs predicted  
ggplot(results\_df\_tree, aes(x = Actual, y = Predicted)) +  
 geom\_point(alpha = 0.6, color = "forestgreen") +  
 geom\_smooth(method = "lm", se = FALSE, linetype = "dashed", color = "darkred") +  
 geom\_abline(slope = 1, intercept = 0, color = "black", linetype = "dotted") +  
 theme\_minimal() +  
 labs(  
 title = "Predicted vs Actual PremiumPrice (Decision Tree)",  
 x = "Actual PremiumPrice",  
 y = "Predicted PremiumPrice"  
 )

## `geom\_smooth()` using formula = 'y ~ x'



# Calculate residuals  
residuals\_df <- data.frame(  
 Actual = test\_data$PremiumPrice,  
 Predicted = predictions,  
 Residuals = test\_data$PremiumPrice - predictions  
)  
  
# Plot residuals  
ggplot(residuals\_df, aes(x = Actual, y = Residuals)) +  
 geom\_point(alpha = 0.6, color = "darkorange") +  
 geom\_hline(yintercept = 0, linetype = "dotted", color = "black") +  
 geom\_smooth(method = "loess", se = FALSE, color = "red", linetype = "dashed") +  
 theme\_minimal() +  
 labs(  
 title = "Residuals vs Actual PremiumPrice (Decision Tree)",  
 x = "Actual PremiumPrice",  
 y = "Residuals (Actual - Predicted)"  
 )

## `geom\_smooth()` using formula = 'y ~ x'



Visual Analysis:

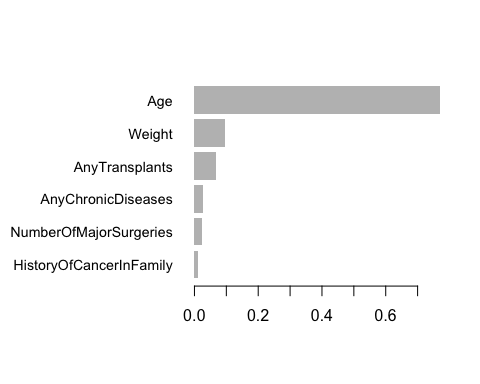
* Predicted vs Actual Plot: Shows a general trend following the diagonal, but with some flattening—indicating the tree predicts close to the average price for many cases (limited flexibility).
* Residual Plot: Shows systematic patterns (underprediction of high prices, overprediction of low prices). The loess curve is curved, not flat—indicating bias.

Interpretation:

* Strengths: Interpretable and simple; good for business users who need logic-based rules.
* Weaknesses: Tends to underfit complex relationships (e.g., interactions between Age × ChronicDisease). Not optimal for modeling premium price variation in detail.

## XGBoost Model

X\_train <- as.matrix(X[train\_indices, ])  
y\_train <- y[train\_indices]  
X\_test <- as.matrix(X[-train\_indices, ])  
y\_test <- y[-train\_indices]  
  
# Convert to DMatrix format for XGBoost  
dtrain <- xgb.DMatrix(data = X\_train, label = y\_train)  
dtest <- xgb.DMatrix(data = X\_test, label = y\_test)  
  
# Train XGBoost Regressor  
xgb\_model <- xgboost(data = dtrain,  
 objective = "reg:squarederror",  
 nrounds = 100,  
 max\_depth = 4,  
 eta = 0.1,  
 verbose = 0)  
  
# Predict  
preds <- predict(xgb\_model, dtest)  
  
# Feature Importance Plot  
importance <- xgb.importance(model = xgb\_model, feature\_names = colnames(X))  
xgb.plot.importance(importance\_matrix = importance)



# True labels  
y\_test <- getinfo(dtest, "label")  
  
# Compute evaluation metrics  
results\_caret <- postResample(pred = preds, obs = y\_test)  
  
# Create a clean results data frame  
resultstest <- data.frame(  
 Results = c("RMSE Test", "R Squared Test", "MAE Test"),  
 Score = as.numeric(results\_caret)  
)  
  
#Model Evaluation Metrics  
print(resultstest)

## Results Score  
## 1 RMSE Test 2777.653651  
## 2 R Squared Test 0.802637  
## 3 MAE Test 1415.892714

# --- Generate Submission Format for PredictedPremiumPrice ---  
submission\_xgb <- data.frame(  
 Id = 1:nrow(X\_test),  
 PredictedPremiumPrice = preds  
)  
  
print("Submission DataFrame:")

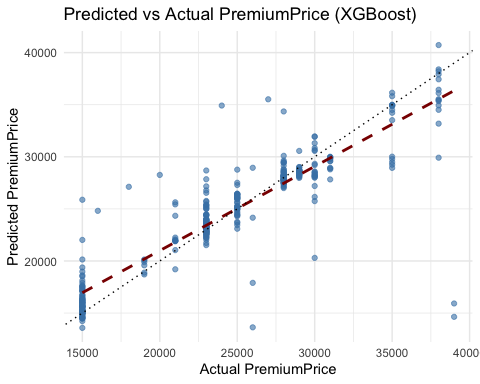
## [1] "Submission DataFrame:"

head(submission\_xgb, 10)

## Id PredictedPremiumPrice  
## 1 1 28685.75  
## 2 2 25763.14  
## 3 3 28218.32  
## 4 4 24989.82  
## 5 5 16117.57  
## 6 6 15048.99  
## 7 7 22589.79  
## 8 8 21930.01  
## 9 9 16336.82  
## 10 10 20070.57

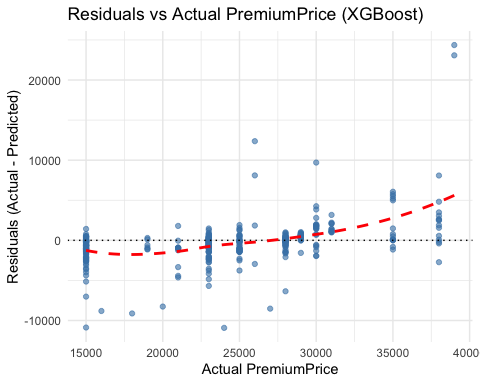
# Create a data frame for plotting  
results\_df <- data.frame(  
 Actual = getinfo(dtest, "label"),  
 Predicted = preds  
)  
  
# Plot actual vs predicted values  
ggplot(results\_df, aes(x = Actual, y = Predicted)) +  
 geom\_point(alpha = 0.6, color = "steelblue") +  
 geom\_smooth(method = "lm", se = FALSE, linetype = "dashed", color = "darkred") +  
 geom\_abline(slope = 1, intercept = 0, color = "black", linetype = "dotted") +  
 theme\_minimal() +  
 labs(  
 title = "Predicted vs Actual PremiumPrice (XGBoost)",  
 x = "Actual PremiumPrice",  
 y = "Predicted PremiumPrice"  
)

## `geom\_smooth()` using formula = 'y ~ x'



# Compute residuals for XGBoost  
residuals\_xgb <- data.frame(  
 Actual = getinfo(dtest, "label"),  
 Predicted = preds,  
 Residuals = y\_test - preds  
)  
  
# Plot residuals  
ggplot(residuals\_xgb, aes(x = Actual, y = Residuals)) +  
 geom\_point(alpha = 0.6, color = "steelblue") +  
 geom\_hline(yintercept = 0, linetype = "dotted", color = "black") +  
 geom\_smooth(method = "loess", se = FALSE, color = "red", linetype = "dashed") +  
 theme\_minimal() +  
 labs(  
 title = "Residuals vs Actual PremiumPrice (XGBoost)",  
 x = "Actual PremiumPrice",  
 y = "Residuals (Actual - Predicted)"  
 )

## `geom\_smooth()` using formula = 'y ~ x'



Visual Analysis:

* Predicted vs Actual Plot: Much closer fit to the diagonal.Better spread of predictions and more accuracy across all premium levels.
* Residual Plot: More centered around 0 and shows less bias than the decision tree. Some increasing error variance (heteroscedasticity), but fewer systematic patterns.

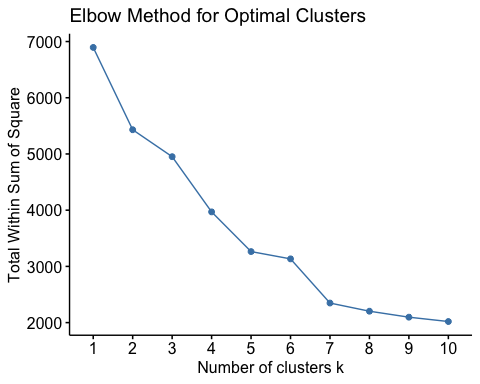
Interpretation:

* Strengths: Captures nonlinear patterns, interactions, and subtle effects. Stronger generalization. More accurate premium pricing.
* Weaknesses: Less interpretable.

# Clustering Methods: KMeans Clustering

We represented distinct groups such as low-risk customers, moderate-risk customers, high-risk customers, and very high-risk customers, each with varying premium levels and health characteristics.

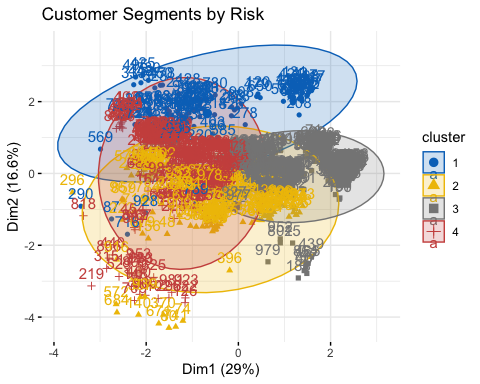
clustering\_df <- df\_new  
  
# Standardize the data (important for clustering)  
clustering\_scaled <- scale(clustering\_df)  
  
#Determine Optimal Number of Clusters  
  
fviz\_nbclust(clustering\_scaled, kmeans, method = "wss") +   
 labs(title = "Elbow Method for Optimal Clusters")



# There is a steep drop from k = 1 to k = 4.  
# After k = 4, the WSS reduction becomes less significant.  
# This makes k = 4 a good candidate for the optimal number of clusters.  
  
# Set number of clusters to 4 (low, moderate, high, very high risk)  
kmeans\_result <- kmeans(clustering\_scaled, centers = 4, nstart = 25)  
  
# Add cluster labels back to original data  
clustering\_df$RiskCluster <- as.factor(kmeans\_result$cluster)  
  
# View cluster counts  
table(clustering\_df$RiskCluster)

##   
## 1 2 3 4   
## 116 156 421 293

fviz\_cluster(kmeans\_result, data = clustering\_scaled,  
 ellipse.type = "norm",  
 palette = "jco",  
 ggtheme = theme\_minimal(),  
 main = "Customer Segments by Risk")



#Label Risk Tiers  
risk\_labels <- c("Low Risk", "Moderate Risk", "High Risk", "Very High Risk")  
clustering\_df$RiskTier <- factor(clustering\_df$RiskCluster,  
 levels = c("1", "2", "3", "4"),  
 labels = risk\_labels)  
  
# Preview results  
head(clustering\_df %>% dplyr::select(PremiumPrice, RiskTier))

## # A tibble: 6 × 2  
## PremiumPrice RiskTier   
## <dbl> <fct>   
## 1 25000 High Risk   
## 2 29000 Very High Risk  
## 3 23000 High Risk   
## 4 28000 Moderate Risk   
## 5 23000 Moderate Risk   
## 6 23000 High Risk

Interpretation:

| Cluster ID | Risk Tier | Count | Interpretation |
| --- | --- | --- | --- |
| 1 | Low Risk | 116 | Likely younger, healthier, fewer surgeries |
| 2 | Moderate Risk | 156 | Mild health conditions or aging effects |
| 3 | High Risk | 421 | Largest group — possibly middle-aged with issues |
| 4 | Very High Risk | 293 | Older or multiple chronic conditions |

## 

## Conclusion and Recommendations – non chat gpt

Our project aimed to uncover key driver of medical insurance premiums through a combination of statistical testing, regression modeling, tree-based machine learning (e.g., XGBoost), and unsupervised clustering. In our analyses, we find a significant prediction between medical premiums and factors like age, chronic conditions, transplant history, and major surgeries. With high levels of accuracy, our model is also able to segments customers into actionable risk tiers. This allows for personalized and data-driven medical premium pricing strategies.

There are numerous business applications for a medical premium analysis. First, insurers can integrate our model into their pricing engine to personalize premium pricing, allowing our suggested variables to adjust quoted premiums. This allows for fairer pricing for low-risk individuals while also improving profit margins for high-risk patients by reducing underpriced high-risk accounts. Patients can be ranked through our k-clustering and tree-based machine learning models. This allows for faster and more efficient evaluations, automating standard low-risk groups. Once high-risk patients are identified, insurance companies can offer targeted wellness programs to high-risk patients for insurance discounts. This reduces long-term claim costs while increasing customer retention.

For future model improvements, we recommend expanding the data source and parameters to include more factors. Key behavioral and lifestyle data is omitted from this dataset, including smoking patterns, and physical activity. Further, adding interaction terms, like Age x Chronic Conditions may better capture risk for older patients. Additionally, we recommend considering longitudinal health records that take into consideration patient health history over a series of time. For equitable and effective practices pricing strategies, insurers should incorporate socioeconomic data and fairness audits for model consideration. For deployment readiness, we recommend that the insurance company builds an API for live quoting that considers current data. Additionally, implementing drift detection will allow for the model to be constantly monitored overtime, allowing insurers to predict accuracy and feature distribution degradation.

## Conclusion and Recommendations – Chat GPT version

This project demonstrates the power of integrating data science, economics, and actuarial reasoning to uncover key drivers of medical insurance premium pricing. By leveraging a combination of statistical testing, regression modeling, tree-based machine learning (e.g., XGBoost), and unsupervised clustering, we effectively quantified how factors like age, chronic conditions, transplant history, and major surgeries influence premium costs. Our models not only forecast premium prices with strong accuracy (XGBoost R² ≈ 0.80), but also segment customers into actionable risk tiers, enabling more personalized and data-driven pricing strategies.

To translate these insights into practice, we recommend that insurers adopt a multi-tiered pricing strategy aligned with customer health profiles. Specifically, premiums can be dynamically adjusted using risk tiers derived from clustering (low to very high risk), with modular pricing for high-cost medical conditions such as chronic illnesses or transplant history. Additionally, integrating age-weight interactions into pricing formulas can improve fairness and accuracy, while incentive-based discounts—such as for stable health indicators or participation in wellness programs—can encourage preventive behavior. Lastly, ongoing monitoring of model fairness and the inclusion of broader socioeconomic features will help ensure that pricing remains both data-driven and socially responsible.

Looking ahead, predictive performance can be enhanced through feature interaction modeling, advanced ensemble techniques, and interpretability tools like SHAP for business transparency. Incorporating socioeconomic data and fairness audits can ensure pricing strategies remain both equitable and effective. This work provides a replicable framework for insurance companies and risk managers to automate premium forecasting, understand medical risk dynamics, and align pricing with customer profiles—all crucial capabilities in the future of personalized insurance analytics.