Code

```
knitr::opts_chunk$set(echo = TRUE)
library(ggplot2)
library(corrplot)
library(glmnet)
library(FNN)
library(caret)
library(MASS)
library(dplyr)


train <- read.csv("~/Desktop/STA 160 Project/train.csv")
test <- read.csv("~/Desktop/STA 160 Project/test.csv")
#We want to see how many NAs we are dealing with or blanks as well
train[train == ""] <- NA
test[test == ""] <- NA

#Rows per
nrow(train) #n=1200000
nrow(test) #n=800000

#Factor it out
train$Gender <- as.factor(train$Gender)
train$Marital.Status <- as.factor(train$Marital.Status)
train$Education.Level <- as.factor(train$Education.Level)
train$Occupation <- as.factor(train$Occupation)
train$Location <- as.factor(train$Location)
train$Policy.Type <- as.factor(train$Policy.Type)
train$Customer.Feedback <- as.factor(train$Customer.Feedback)
train$Smoking.Status <- as.factor(train$Smoking.Status)
train$Exercise.Frequency <- as.factor(train$Exercise.Frequency)
train$Property.Type <- as.factor(train$Property.Type)


#We see a lot of NAs in certain regions
train_na <- round(colSums(is.na(train))/nrow(train),3)*100
test_na <- round(colSums(is.na(test))/nrow(test),3)*100
train_na
test_na

#We can NA Omit everything and still have lots of data
training_set <- na.omit(train)
testing_set <- na.omit(test)

names(training_set)
names(testing_set) #No response here


nrow(training_set)


names(training_set)
```

```r
training_set <- training_set %>%
  dplyr::select(-id, -Policy.Start.Date)

testing_set <- testing_set %>%
  dplyr::select(-id, -Policy.Start.Date)




#Let's do a 70-30 split
set.seed(123)

split_idx <- sample(nrow(training_set),size=0.7*nrow(training_set))
train_df <- training_set[split_idx,]
test_df <- training_set[-split_idx,]
names(train_df)



#Our new size to work with
nrow(train_df)
nrow(test_df)
#Overall a pretty even split
table(train_df$Gender)
table(train_df$Marital.Status)
table(train_df$Location)
table(train_df$Smoking.Status)

#Graphs show this too
ggplot(train_df,aes(x = Gender,fill=Gender)) +
  geom_bar() +
  scale_y_continuous(labels=scales::comma) +
  labs(title = "Gender Distribution") +
  theme_minimal()

ggplot(train_df, aes(x = Marital.Status,fill=Marital.Status)) +
  geom_bar() +
  scale_y_continuous(labels=scales::comma) +
  labs(title = "Marital Status Distribution") +
  theme_minimal()

ggplot(train_df, aes(x = Location,fill=Location)) +
  geom_bar() +
  labs(title = "Location Distribution") +
  scale_y_continuous(labels=scales::comma) +
  theme_minimal()

ggplot(train_df, aes(x = Smoking.Status,fill=Smoking.Status)) +
  geom_bar() +
  labs(title = "Smoking Status Distribution") +
  scale_y_continuous(labels=scales::comma) +
  theme_minimal()
```

```r
ggplot(train_df, aes(x = Age)) +
  geom_histogram(bins = 30, fill = "darkgreen", alpha = 0.7) +
  scale_y_continuous(labels=scales::comma) +
  labs(title = "Age Distribution",xlab="Age") +
  theme_minimal()

ggplot(train_df, aes(x = Annual.Income)) +
  geom_histogram(bins = 50, fill = "navy", alpha = 0.7) +
  scale_y_continuous(labels=scales::comma) +
  labs(title = "Annual Income Distribution",xlab="Annual Income") +
  scale_x_continuous(labels =scales::dollar)+
  theme_minimal()

ggplot(train_df, aes(x = Credit.Score)) +
  geom_histogram(bins = 30, fill = "purple1", alpha = 0.7) +
  scale_y_continuous(labels=scales::comma) +
  labs(title = "Credit Score Distribution",xlab="Credit Score") +
  theme_minimal()

ggplot(train_df, aes(x = Annual.Income)) +
  geom_histogram(bins = 50, fill = "blue") +
  scale_y_continuous(labels = scales::comma) +
  scale_x_continuous(
    labels = scales::dollar,
    breaks = seq(0, 150000, 25000)
  ) +
  labs(
    title = "Annual Income Distribution",
    x = "Annual Income"
  ) +
  theme_minimal()


#Looking at means of numerical values
cat("Age summary \n")
summary(train_df$Age)
cat("Annual Income Summary\n")
summary(train_df$Annual.Income)
cat("Credit Score Summary\n")
summary(train_df$Credit.Score)
cat("Premium Amount Summary\n")
summary(train_df$Premium.Amount)



numeric_columns <- train_df %>% dplyr::select(where(is.numeric))
corr_matrix <- cor(numeric_columns, use = "complete.obs")


corrplot(corr_matrix,
         method = "color",
         tl.col = "black",
```

```
         tl.srt = 90,
         addCoef.col = "black",
         number.cex = 0.4,
         diag = FALSE)

#Our inagural model
first.model <- lm(Premium.Amount~.,data=train_df)
summary(first.model)
AIC(first.model)

train_model_AIC <- step(
  lm(Premium.Amount ~ 1, data = train_df),
  scope = list(
    lower = ~1,
    upper = formula(lm(Premium.Amount ~ .^2, data = train_df))
  ),trace = 0
)

summary(train_model_AIC)
AIC(train_model_AIC)
summary(train_model_AIC)
par(mfrow = c(2, 2))
plot(train_model_AIC)



bc_result <- boxcox(train_model_AIC)

best_lambda <- bc_result$x[which.max(bc_result$y)]
print(best_lambda)


train_df$Premium.BC <- if(best_lambda == 0) {
  log(train_df$Premium.Amount)
} else {
  (train_df$Premium.Amount^best_lambda - 1) / best_lambda
}

#Same Model from Box Cox
lm_bc_model <- lm(Premium.BC ~ Previous.Claims + Credit.Score +
                    Annual.Income + Health.Score + Age + Gender + Customer.Feedback +
                    Previous.Claims:Annual.Income + Previous.Claims:Credit.Score +
                    Credit.Score:Annual.Income + Annual.Income:Health.Score +
                    Previous.Claims:Health.Score + Credit.Score:Health.Score +
                    Credit.Score:Age + Gender:Customer.Feedback +
                    Credit.Score:Customer.Feedback + Annual.Income:Age +
                    Previous.Claims:Customer.Feedback, data = train_df)

# Summary of Box-Cox model
summary(lm_bc_model)
```

```r
num_cols <- intersect(
  names(train_df)[sapply(train_df, is.numeric)],
  names(test_df)[sapply(test_df, is.numeric)]
)


x_train <- scale(train_df[, num_cols]) %>% as.data.frame()
x_test <- scale(test_df[, num_cols]) %>% as.data.frame()


pred_knn <- knn.reg(train = x_train, test = x_test, y = train_df$Premium.Amount,
                    k = 10)$pred

rmse_knn <- sqrt(mean((test_df$Premium.Amount - pred_knn)^2))
r2_knn <- 1 - sum((test_df$Premium.Amount - pred_knn)^2) /
  sum((test_df$Premium.Amount - mean(test_df$Premium.Amount))^2)


cat("RSME (Numeric Columns):", round(rmse_knn, 2), "\n")
cat( "R² (Numeric Columns):", round(r2_knn, 4), "\n")

plot(test_df$Premium.Amount, pred_knn,
     xlab = "Actual Premium Amount ($)", ylab = "Predicted Premium Amount ($)",
     main = "KNN: Actual vs Predicted (Numerical Predictors Only)",
     col = rgb(0, 0, 1, 0.3), pch = 16)



#Building model matricies
train_df <- train_df %>% dplyr::select(-Premium.BC)

x_train <- model.matrix(Premium.Amount ~ ., data = train_df)[, -1]
x_test <- model.matrix(Premium.Amount ~ ., data = test_df)[, -1]




#Scaling and ENcoding
x_train_scaled <- scale(x_train)
x_test_scaled <- scale(x_test)

#KNN
pred_knn_all <- knn.reg(
  x_train_scaled,
  x_test_scaled,
  y = train_df$Premium.Amount,
  k = 10
)$pred

rmse_knn_all <- sqrt(mean((test_df$Premium.Amount - pred_knn_all)^2))
r2_knn_all <- 1 - sum((test_df$Premium.Amount - pred_knn_all)^2) /
                  sum((test_df$Premium.Amount - mean(test_df$Premium.Amount))^2)
```

```r
cat("RSME (All Columns):", round(rmse_knn_all, 2), "\n")
cat( "R² (All Columns):", round(r2_knn_all, 4), "\n")

plot(test_df$Premium.Amount, pred_knn_all,
     xlab = "Actual Premium Amount ($)", ylab = "Predicted Premium Amount ($)",
     main = "KNN: Actual vs Predicted (All Predictors)",
     col = rgb(0, 0, 1, 0.5), pch = 16)


#Log transformations
train_df$log_premium <- log1p(train_df$Premium.Amount)
test_df$log_premium <- log1p(test_df$Premium.Amount)

x_train <- model.matrix(log_premium ~ ., data = train_df)[, -1]
x_test <- model.matrix(log_premium ~ ., data = test_df)[, -1]

#Scaling it
x_train_scaled <- scale(x_train)
x_test_scaled <- scale(x_test)

pred_log_knn <- knn.reg(train = x_train_scaled,
                        test = x_test_scaled,
                        y = train_df$log_premium,
                        k = 10)$pred

pred_knn_final <- expm1(pred_log_knn)

rmse_log <- sqrt(mean((test_df$Premium.Amount - pred_knn_final)^2))
r2_log <- 1 - sum((test_df$Premium.Amount - pred_knn_final)^2) /
              sum((test_df$Premium.Amount - mean(test_df$Premium.Amount))^2)


plot(test_df$Premium.Amount, pred_knn_final,
     xlab = "Actual Premium Amount ($)", ylab = "Predicted Premium Amount ($)",
     main = "KNN: Actual vs Predicted ",
     col = c("blue"), pch = 16)
cat("RSME (Log):", round(rmse_log, 2), "\n")
cat( "R² (Log):", round(r2_log, 4), "\n")




# Log transform
train_df$log_premium <- log1p(train_df$Premium.Amount)

# Design matrix
x <- model.matrix(log_premium ~ ., data = train_df)[, -1]
x <- scale(x)

#Combine into one
full_df <- as.data.frame(x)
full_df$log_premium <- train_df$log_premium
```

```r
full_df$premium <- train_df$Premium.Amount

# 5 Fold
set.seed(123)
folds <- createFolds(full_df$log_premium, k = 5, list = TRUE)

rmse_vals <- c()
r2_vals <- c()

for (i in 1:5) {
  test_idx <- folds[[i]]
  train_idx <- setdiff(seq_len(nrow(full_df)), test_idx)

  x_train <- as.matrix(full_df[train_idx, !(names(full_df) %in%
                                              c("log_premium", "premium"))])
  y_train <- full_df$log_premium[train_idx]
  x_test <- as.matrix(full_df[test_idx, !(names(full_df) %in%
                                            c("log_premium", "premium"))])
  y_test_true <- full_df$premium[test_idx]

  #Prediction versus true errir
  pred_log <- knn.reg(train = x_train, test = x_test, y = y_train, k = 10)$pred
  pred_dollar <- expm1(pred_log)

  rmse <- sqrt(mean((y_test_true - pred_dollar)^2))
  r2 <- 1 - sum((y_test_true - pred_dollar)^2) / sum((y_test_true -
                                                        mean(y_test_true))^2)

  rmse_vals <- c(rmse_vals, rmse)
  r2_vals <- c(r2_vals, r2)
}

# Results
mean_rmse <- mean(rmse_vals)
mean_r2 <- mean(r2_vals)

cat("Average RMSE over 5 folds:", round(mean_rmse, 2), "\n")
cat("Average R² over 5 folds:", round(mean_r2, 4), "\n")
```