Advanced Medical Insurance Cost Prediction Model II

Dr. Michael Adu

2024-12-18

Contents

1.0 Background of the Study

The cost of medical care significantly impacts both healthcare providers and patients. This project aims to explore the predictive utility of patient features captured by an insurance firm to estimate the annual cost of medical care. The dataset used is the publicly available Medical Cost Personal dataset from Kaggle, containing information on 1338 beneficiaries and 7 variables, including the target variable: medical costs billed by health insurance in a year.In this study, we aim to build upon previous work by applying advanced techniques to improve the accuracy of predictions and enhance model interpretability.

1.1 Overview of Features

- **age:** Age of the primary beneficiary.
- **sex:** Gender of the insurance contractor.
- **bmi:** Body mass index of the beneficiary.
- **children:** Number of children covered by health insurance.
- **smoker:** Smoking status of the beneficiary.
- **region:** Residential area of the beneficiary in the US.
- **charges:** Individual medical costs per beneficiary billed by health insurance in a year.

```
df = read.csv('insurance.csv', header=TRUE) #loading data
str(df) #examining structure of the dataset
```

```
## 'data.frame':    1338 obs. of  7 variables:
##  $ age     : int  19 18 28 33 32 31 46 37 37 60 ...
##  $ sex     : chr  "female" "male" "male" "male" ...
##  $ bmi     : num  27.9 33.8 33 22.7 28.9 ...
##  $ children: int  0 1 3 0 0 0 1 3 2 0 ...
##  $ smoker  : chr  "yes" "no" "no" "no" ...
```

```
##  $ region  : chr  "southwest" "southeast" "southeast" "northwest" ...
##  $ charges : num  16885 1726 4449 21984 3867 ...
```

```
summary(df)
```

```
##       age            sex                 bmi           children
##  Min.   :18.00   Length:1338        Min.   :15.96   Min.   :0.000
##  1st Qu.:27.00   Class :character   1st Qu.:26.30   1st Qu.:0.000
##  Median :39.00   Mode  :character   Median :30.40   Median :1.000
##  Mean   :39.21                      Mean   :30.66   Mean   :1.095
##  3rd Qu.:51.00                      3rd Qu.:34.69   3rd Qu.:2.000
##  Max.   :64.00                      Max.   :53.13   Max.   :5.000
##     smoker             region             charges
##  Length:1338        Length:1338        Min.   : 1122
##  Class :character   Class :character   1st Qu.: 4740
##  Mode  :character   Mode  :character   Median : 9382
##                                        Mean   :13270
##                                        3rd Qu.:16640
##                                        Max.   :63770
```

1.2 Significance

Developing a robust predictive model for medical costs is crucial for assisting healthcare providers, insurers, and policymakers. This study aims to demonstrate the practical application of such a model.

1.3 Objective

The objective is to develop a predictive model using advanced regression techniques, establishing relationships between predictor variables (e.g., age, BMI, location) and the target variable (medical cost).

1.4 Scope and Limitations

While the model provides valuable insights based on historical data, it assumes observed relationships will continue in the future. External factors not in the dataset may influence medical costs in the real world.

1.5 Disclaimer

The objective of this study is to demonstrate the development of a linear regression model for the purpose of learning and research only and does not necessarily reflect the real-world situation for predicting cost of insurance or medical care for any individual patient. The study findings are not intended to be used for any commercial or diagnostic purposes.

2.0 Methodology

2.1 Study Design

This study employs advanced statistical modelling and machine learning techniques to predict cost of medical insurance based on patient characteristics.

2.2 Data Preprocessing

```
# Load the dataset
df <- read.csv('insurance.csv')

# Data preprocessing
df$sex <- as.factor(df$sex)
```

```r
df$smoker <- as.factor(df$smoker)
df$region <- as.factor(df$region)

# Feature Engineering: Add interaction terms and transformations
df$smoker_bmi <- ifelse(df$smoker == "yes", df$bmi, 0)
df$age_squared <- df$age^2

head(df)
```

```
##   age    sex    bmi children smoker    region   charges smoker_bmi age_squared
## 1  19 female 27.900        0    yes southwest 16884.924       27.9         361
## 2  18   male 33.770        1     no southeast  1725.552        0.0         324
## 3  28   male 33.000        3     no southeast  4449.462        0.0         784
## 4  33   male 22.705        0     no northwest 21984.471        0.0        1089
## 5  32   male 28.880        0     no northwest  3866.855        0.0        1024
## 6  31 female 25.740        0     no southeast  3756.622        0.0         961
```

2.3 Splitting the Dataset

A train-test split ensures that the model is evaluated on unseen data.

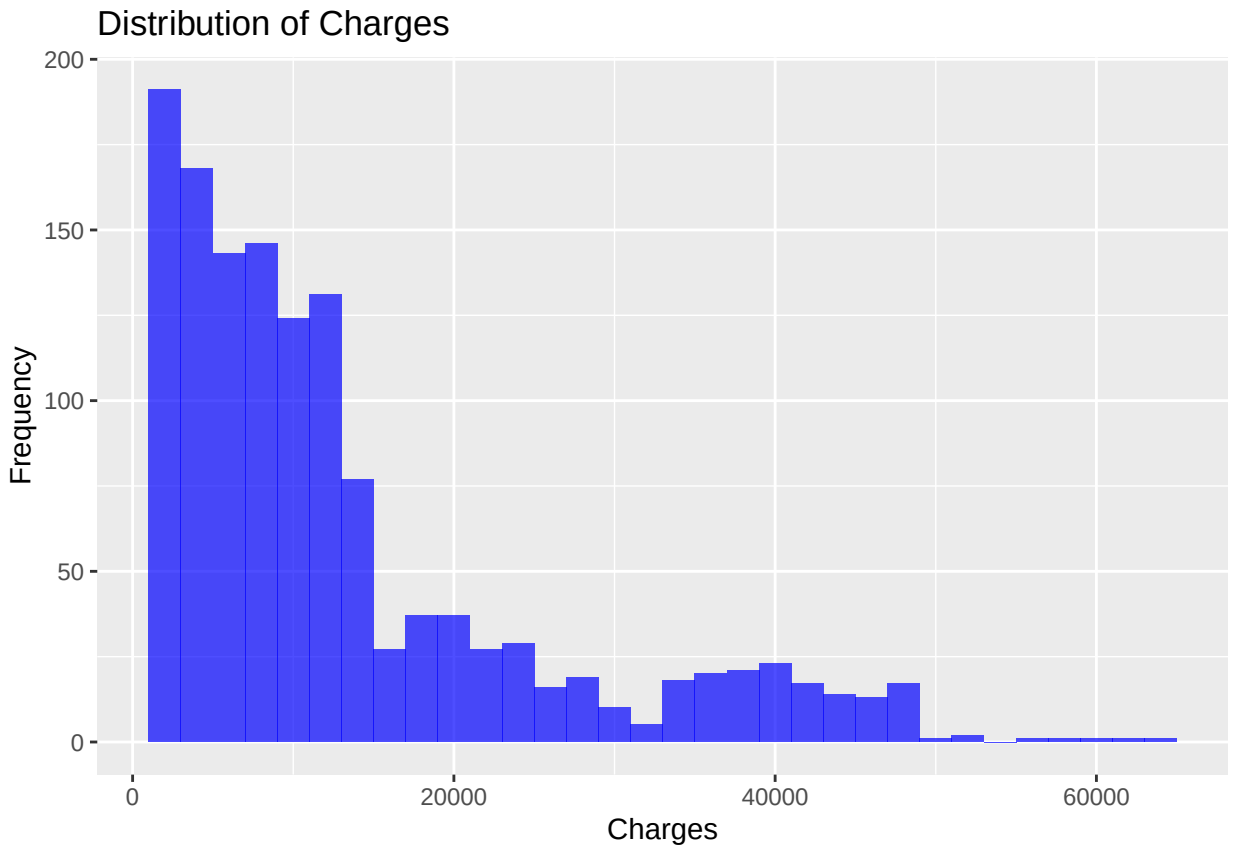
```r
set.seed(42)
train_index <- createDataPartition(df$charges, p = 0.8, list = FALSE)
train_data <- df[train_index, ]
test_data <- df[-train_index, ]
```

2.4 Exploratory Data Analysis

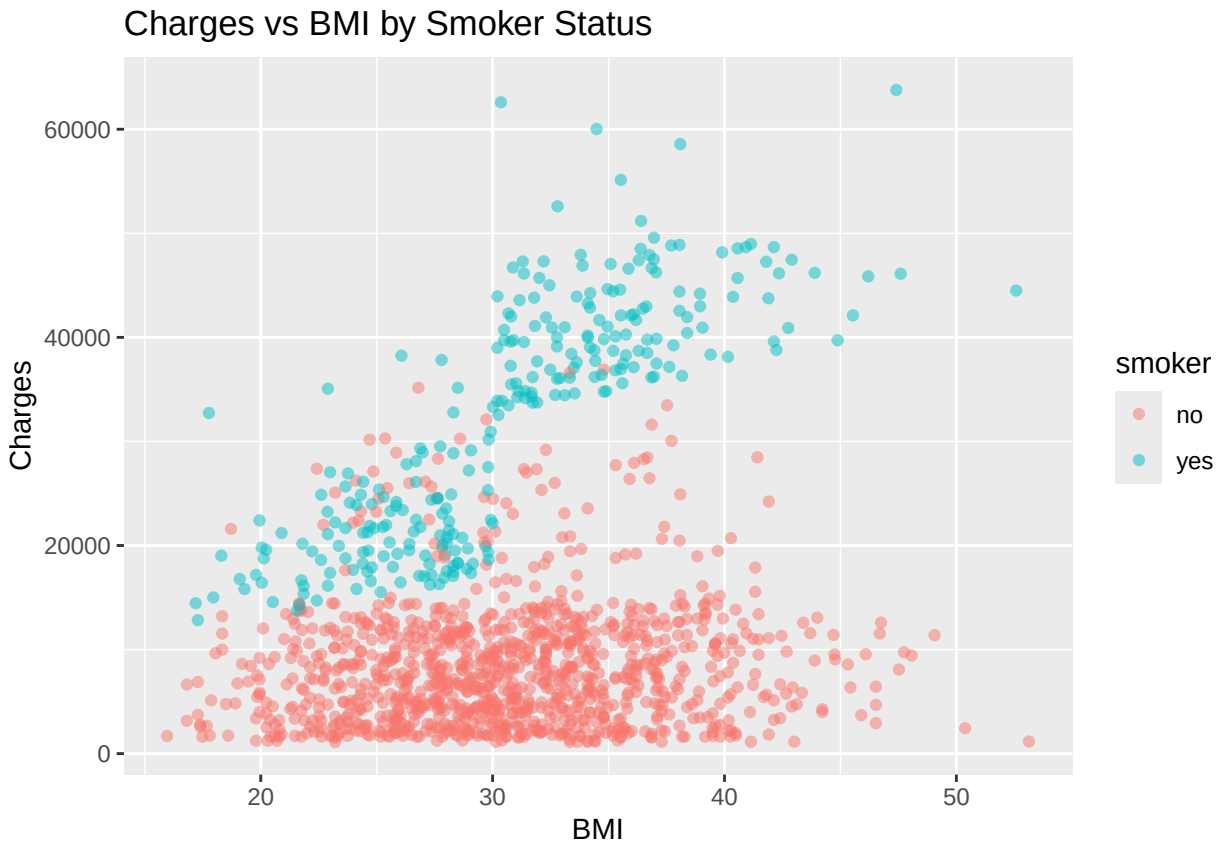
```r
# Charges Distribution
summary(df$charges)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1122    4740    9382   13270   16640   63770
```

```r
ggplot(df, aes(x = charges)) +
  geom_histogram(binwidth = 2000, fill = "blue", alpha = 0.7) +
  labs(title = "Distribution of Charges", x = "Charges", y = "Frequency")
```




```r
# Charges vs BMI
ggplot(df, aes(x = bmi, y = charges, color = smoker)) +
  geom_point(alpha = 0.5) +
  labs(title = "Charges vs BMI by Smoker Status", x = "BMI", y = "Charges")
```



3.0 Model Development and Evaluation

3.1 Linear Regression

```
#Define function to calculate RMSE
calc_rmse <- function(actual, predicted) {
  sqrt(mean((actual - predicted)^2))
}

lm_model <- lm(charges ~ age + age_squared + bmi + children + smoker + region + smoker_bmi, data = trai
summary(lm_model)
```

```
##
## Call:
## lm(formula = charges ~ age + age_squared + bmi + children + smoker +
##     region + smoker_bmi, data = train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14572.3  -1549.4  -1292.3   -859.2  22923.6
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      2.369e+03  1.502e+03   1.577   0.1150
## age             -1.134e+01  7.077e+01  -0.160   0.8727
## age_squared      3.554e+00  8.846e-01   4.017 6.30e-05 ***
## bmi              6.057e+00  2.777e+01   0.218   0.8274
```

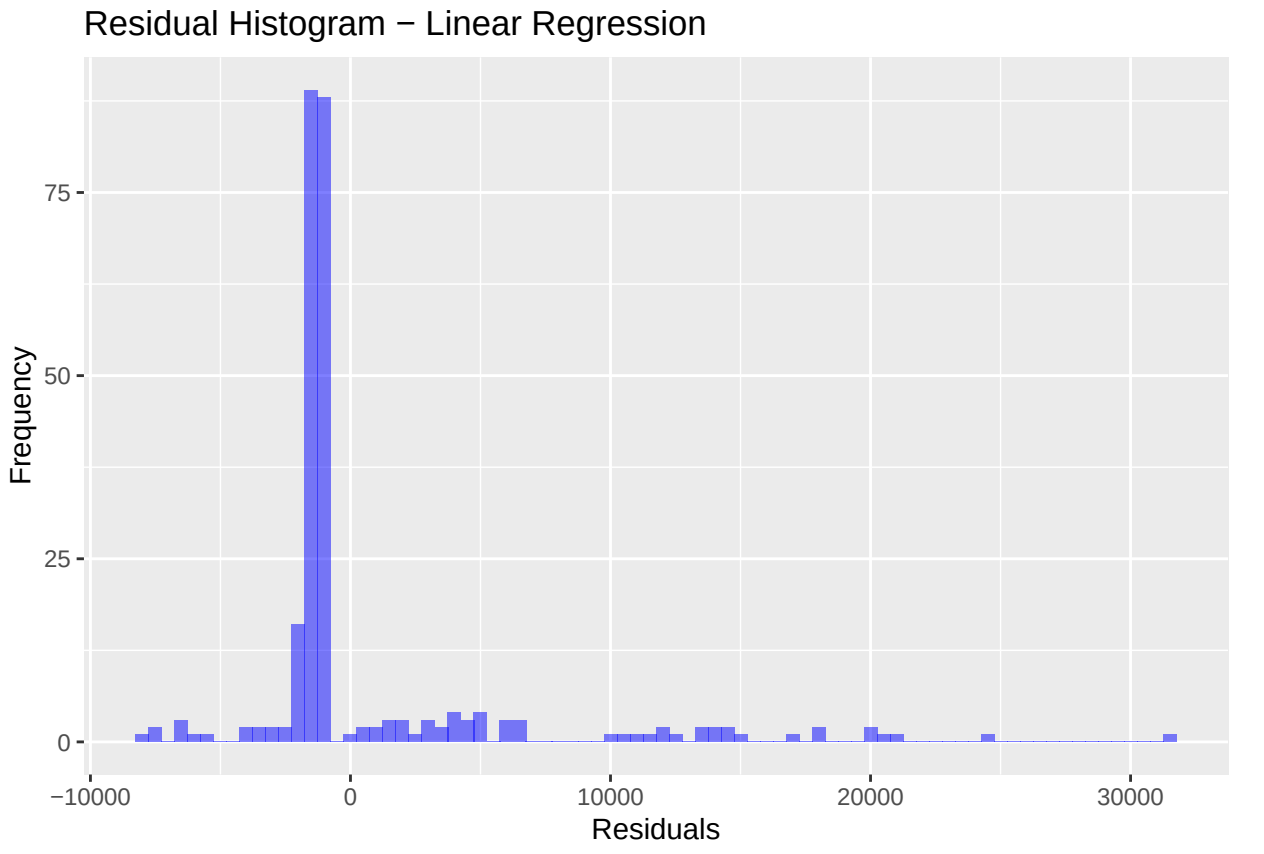
```
## children          6.418e+02  1.248e+02   5.141 3.25e-07 ***
## smokeryes        -2.094e+04  1.758e+03 -11.910  < 2e-16 ***
## regionnorthwest -4.727e+02  4.093e+02  -1.155   0.2484
## regionsoutheast -1.065e+03  4.126e+02  -2.580   0.0100 *
## regionsouthwest -1.063e+03  4.178e+02  -2.545   0.0111 *
## smoker_bmi        1.451e+03  5.572e+01  26.036  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4659 on 1062 degrees of freedom
## Multiple R-squared:  0.8537, Adjusted R-squared:  0.8525
## F-statistic: 688.8 on 9 and 1062 DF,  p-value: < 2.2e-16
```

```r
# Predict and evaluate
lm_predictions <- predict(lm_model, test_data)
lm_rmse <- calc_rmse(test_data$charges, lm_predictions)
cat("Linear Regression RMSE:", lm_rmse, "\n")
```

```
## Linear Regression RMSE: 5408.644
```

3.1.1 Residual Analysis

```r
residuals <- test_data$charges - lm_predictions
ggplot(data.frame(residuals), aes(x = residuals)) +
  geom_histogram(binwidth = 500, fill = "blue", alpha = 0.5) +
  labs(title = "Residual Histogram - Linear Regression", x = "Residuals", y = "Frequency")
```



3.2 Interpretations

3.2.1 Model Overview

- **Residuals:**

The spread of residuals indicates how well the model's predictions match the actual data. in this model, Min residual: -14572.3; Max residual: 22923.6 shows some large errors, especially on the higher end. However most of the residuals from the graph are very close to 0, which that for majority of the data points the model is predicting relatively accurately.

- **Outliers and Skewness:**

The histogram shows some large residuals (both positive and negative) far from zero. These represent instances where the model's predictions are far off the mark.The residuals extend significantly to the right (positive residuals), suggesting the presence of underestimation for certain data points.

- **Non-Normal Residual Distribution:**

Ideally, residuals in a well-fitted linear regression model should follow a normal distribution centered around zero.Here, the residuals are heavily skewed and far from symmetrical, indicating potential problems with model assumptions.

Understanding output of our model:

- **Residual Standard Error (RSE):**

Value: 4659 This Indicates the typical distance of observed data points from the regression line.

- **R-squared:**

Multiple R-squared:

Value: 0.8537 85.37% of the variance in charges is explained by the predictors.

Adjusted R-squared: 0.8525 adjusts for the number of predictors, still very high, indicating a good fit.

- **F-statistic:**

F = 688.8, p-value < 2.2e-16 The overall model is highly statistically significant.

- **RMSE:**

Value: 5408.644 Root Mean Square Error quantifies the average error magnitude, indicating a typical error of approximately 5409 units in predicted charges.

Coefficients Interpretation

- **(Intercept):**

Estimate: 2369 (not statistically significant at = 0.05). This is the baseline charge when all predictors are at their reference levels.

- **age:**

Estimate: -11.34, not significant (p = 0.8727). Suggests no linear relationship between age and charges after accounting for age_squared.

- **age_squared:**

Estimate: 3.554, highly significant (p = 6.3e-05). Indicates a significant quadratic effect of age on charges, implying costs increase more sharply for older individuals.

- **bmi:**

Estimate: 6.057, not significant (p = 0.8274). BMI alone doesn't significantly impact charges after accounting for smoker_bmi.

- **children:**

Estimate: 641.8, highly significant (p = 3.25e-07). Each additional child increases charges by approximately $641.8 on average.

- **smokeryes:**

Estimate: -20940, highly significant (p < 2e-16). Smokers, on average, have significantly lower base costs, but this effect is counteracted by the strong positive interaction with smoker_bmi.

- **region:**

northwest: Estimate -472.7 (not significant, p = 0.2484). southeast: Estimate -1065 (significant, p = 0.01). southwest: Estimate -1063 (significant, p = 0.011). Indicates some regional differences, with southeast and southwest having significantly lower charges compared to the baseline (northeast).

- **smoker_bmi:**

Estimate: 1451, highly significant (p < 2e-16). Suggests a very strong interaction between being a smoker and BMI; for smokers, higher BMI leads to dramatically increased costs.

Insights The model explains most of the variability in charges (R^2 85%).

- **Significant Predictors:** age_squared, children, smoker, smoker_bmi, and some region effects are highly statistically significant.
- **Non-Significant Predictors:** age, bmi (without interaction), regionnorthwest are not significant.
- **Practical Takeaways**
- Smoking combined with BMI has the largest impact on increasing charges.
- There is a quadratic effect of age on charges, indicating that costs increase non-linearly as people age.
- Living in southeast or southwest regions may slightly lower costs.

3.1.2 Refined Linear Model

```
# Refine the model by excluding non-significant predictors
refined_model <- lm(formula = charges ~ age_squared + children + smoker +  region + smoker_bmi, data =

# Summarize the refined model
summary(refined_model)
```

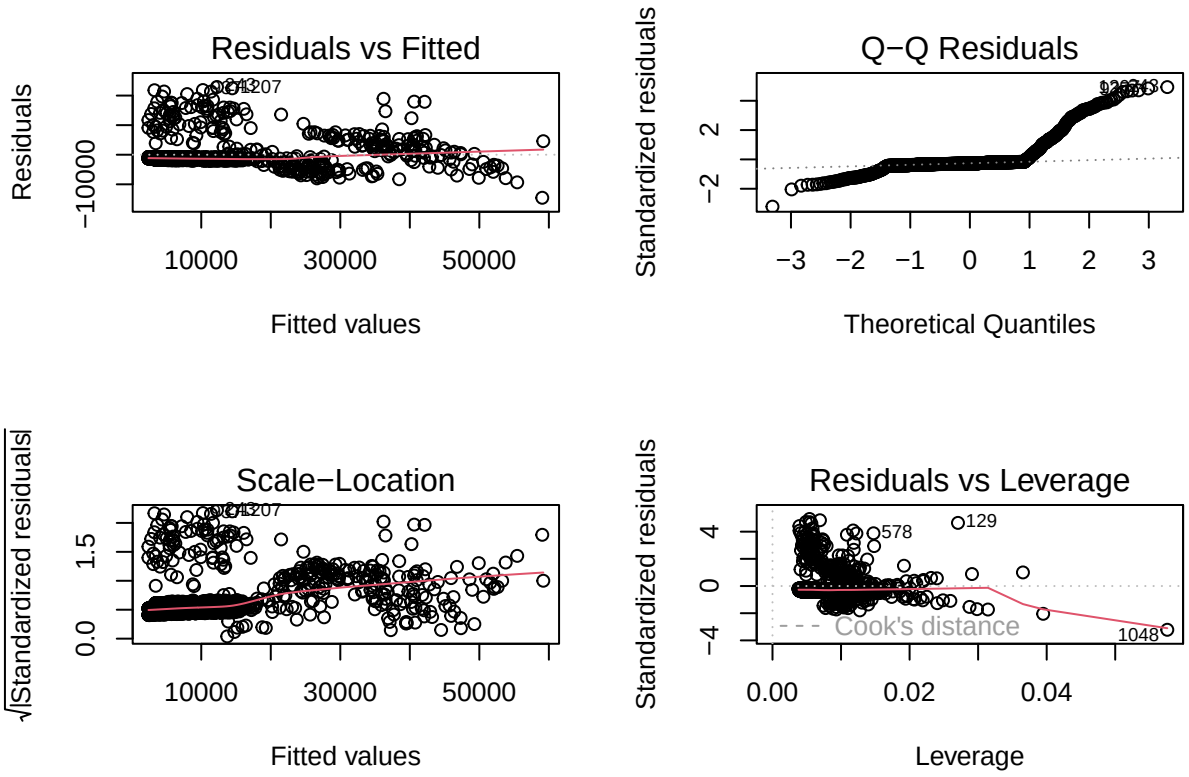
```
##
## Call:
## lm(formula = charges ~ age_squared + children + smoker + region +
##     smoker_bmi, data = train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14551.7  -1548.3  -1287.5   -888.2  22896.5
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      2.345e+03  3.963e+02   5.917 4.42e-09 ***
## age_squared      3.418e+00  1.271e-01  26.890  < 2e-16 ***
## children         6.367e+02  1.187e+02   5.362 1.01e-07 ***
## smokeryes       -2.112e+04  1.569e+03 -13.461  < 2e-16 ***
## regionnorthwest -4.730e+02  4.086e+02  -1.157  0.24736
## regionsoutheast -1.044e+03  4.024e+02  -2.593  0.00963 **
```

```
## regionsouthwest -1.056e+03  4.163e+02  -2.538  0.01130 *
## smoker_bmi       1.456e+03  4.946e+01  29.446  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4655 on 1064 degrees of freedom
## Multiple R-squared:  0.8537, Adjusted R-squared:  0.8528
## F-statistic: 887.2 on 7 and 1064 DF,  p-value: < 2.2e-16
```

```r
# Calculate RMSE for the refined model
predicted_values <- predict(refined_model, newdata = train_data)
actual_values <- train_data$charges
rmse_refined <- sqrt(mean((predicted_values - actual_values)^2))

cat("Refined Model RMSE:", rmse_refined, "\n")
```

```
## Refined Model RMSE: 4637.44
```

```r
# Assessing model assumptions
par(mfrow=c(2,2))
plot(refined_model)
```



Interpretation of the Refined Model Outputs

- **Model Assumptions:**Model assumptions of linearity and normality are mostly met.
- **Residual Standard Error (RSE):**Reduced slightly to 4655 from 4659 in the original model.
- **Adjusted R-squared:** Improved marginally from 0.8525 to 0.8528.
- **Refined RMSE:**Reduced significantly to 4637.44 from 5408.644, indicating improved prediction accuracy.

- **Significant Predictors:**
  - age_squared: Highly significant ($p < 2e\text{-}16$), confirming the quadratic effect of age.
  - children: Highly significant ($p < 0.001$), with each child adding ~$636.7 to costs.
  - smoker: Smoking dramatically reduces baseline costs but is counteracted by the interaction with BMI.
  - smoker_bmi: Highly significant, with a very strong positive interaction between smoking and BMI.
  - regionsoutheast and regionsouthwest: These regions have significantly lower costs than the baseline(northeast).
- **Insignificant Predictors:**

regionnorthwest: Retained for completeness but remains insignificant ($p = 0.24736$).

Performance Improvement

Better Fit: Adjusted R-squared indicates a slight improvement in explanatory power after simplifying the model.

Lower Error: RMSE dropped by ~14%, showing better predictions.

3.2 Random Forest with Bayesian Optimization

Bayesian Optimization is employed to tune hyperparameters of the Random Forest model.

NB: Bayesian Optimization can be used to reduce the computational cost of selecting the best hyperparameters in computationally expensive models. The goal is to efficiently explore the hyperparameter space and identify a set of hyperparameters that minimize the model's cost function (e.g., RMSE for regression tasks) using a probabilistic surrogate model. This process helps achieve the best model accuracy with fewer evaluations compared to exhaustive search methods.

```
rf_bayesian <- function(mtry, min_node_size) {
  model <- randomForest(
    charges ~ age + age_squared + bmi + children + smoker + region + smoker_bmi,
    data = train_data,
    mtry = as.integer(mtry),
    nodesize = as.integer(min_node_size)
  )
  predictions <- predict(model, test_data)
  rmse <- calc_rmse(test_data$charges, predictions)
  list(Score = -rmse)
}

set.seed(42)
rf_bo <- BayesianOptimization(
  FUN = rf_bayesian,
  bounds = list(mtry = c(2L, 5L), min_node_size = c(5L, 15L)),
  init_points = 5,
  n_iter = 20,
  acq = "ucb"
)
```

```
## elapsed = 2.42   Round = 1   mtry = 5.0000   min_node_size = 10.0000 Value = -5288.5540
## elapsed = 1.97   Round = 2   mtry = 5.0000   min_node_size = 12.0000 Value = -5276.7645
## elapsed = 1.49   Round = 3   mtry = 3.0000   min_node_size = 6.0000  Value = -5295.0543
## elapsed = 1.25   Round = 4   mtry = 4.0000   min_node_size = 12.0000 Value = -5269.7620
```

```
## elapsed = 1.39   Round = 5   mtry = 4.0000   min_node_size = 12.0000 Value = -5255.5042
## elapsed = 0.74   Round = 6   mtry = 2.0000   min_node_size = 12.0000 Value = -5175.6109
## elapsed = 1.06   Round = 7   mtry = 2.0000   min_node_size = 5.0000  Value = -5195.7240
## elapsed = 0.81   Round = 8   mtry = 2.0000   min_node_size = 14.0000 Value = -5184.0545
## elapsed = 0.78   Round = 9   mtry = 2.0000   min_node_size = 10.0000 Value = -5190.2363
## elapsed = 0.72   Round = 10  mtry = 2.0000   min_node_size = 12.0000 Value = -5206.5339
## elapsed = 0.91   Round = 11  mtry = 2.0000   min_node_size = 12.0000 Value = -5173.1805
## elapsed = 0.75   Round = 12  mtry = 2.0000   min_node_size = 12.0000 Value = -5187.9912
## elapsed = 0.72   Round = 13  mtry = 2.0000   min_node_size = 12.0000 Value = -5191.4101
## elapsed = 0.86   Round = 14  mtry = 2.0000   min_node_size = 12.0000 Value = -5180.6849
## elapsed = 0.72   Round = 15  mtry = 2.0000   min_node_size = 12.0000 Value = -5173.3701
## elapsed = 0.73   Round = 16  mtry = 2.0000   min_node_size = 12.0000 Value = -5185.2254
## elapsed = 0.82   Round = 17  mtry = 2.0000   min_node_size = 12.0000 Value = -5189.3970
## elapsed = 0.74   Round = 18  mtry = 2.0000   min_node_size = 12.0000 Value = -5162.4277
## elapsed = 0.72   Round = 19  mtry = 2.0000   min_node_size = 12.0000 Value = -5183.1236
## elapsed = 0.75   Round = 20  mtry = 2.0000   min_node_size = 12.0000 Value = -5164.5908
## elapsed = 0.78   Round = 21  mtry = 2.0000   min_node_size = 12.0000 Value = -5191.8742
## elapsed = 1.00   Round = 22  mtry = 2.0000   min_node_size = 12.0000 Value = -5185.4855
## elapsed = 0.82   Round = 23  mtry = 2.0000   min_node_size = 12.0000 Value = -5182.5062
## elapsed = 0.74   Round = 24  mtry = 2.0000   min_node_size = 12.0000 Value = -5172.3029
## elapsed = 1.11   Round = 25  mtry = 2.0000   min_node_size = 12.0000 Value = -5181.5966
##
##  Best Parameters Found:
## Round = 18   mtry = 2.0000   min_node_size = 12.0000 Value = -5162.4277
```

```r
# Train the optimized Random Forest model
rf_optimized <- randomForest(
  charges ~ age + age_squared + bmi + children + smoker + region + smoker_bmi,
  data = train_data,
  mtry = rf_bo$Best_Par["mtry"],
  nodesize = rf_bo$Best_Par["min_node_size"]
)
rf_predictions_optimized <- predict(rf_optimized, test_data)
rf_rmse_optimized <- calc_rmse(test_data$charges, rf_predictions_optimized)
cat("Optimized Random Forest RMSE:", rf_rmse_optimized, "\n")
```

```
## Optimized Random Forest RMSE: 5189.678
```

3.3 XGBoost with Bayesian Optimization

XGBoost (eXtreme Gradient Boosting) is a powerful boosting algorithm used for supervised learning problems. In this study, we employ Bayesian optimization to tune XGBoost's hyperparameters to minimize the RMSE.

```r
# One-hot encode categorical variables
dummy_model <- dummyVars(" ~ .", data = train_data, fullRank = TRUE)
train_data_encoded <- data.frame(predict(dummy_model, newdata = train_data))
test_data_encoded <- data.frame(predict(dummy_model, newdata = test_data))

# Ensure all columns are numeric
train_data_encoded <- train_data_encoded %>% mutate(across(everything(), as.numeric))
test_data_encoded <- test_data_encoded %>% mutate(across(everything(), as.numeric))

# Create xgb.DMatrix objects
xgb_train <- xgb.DMatrix(data = as.matrix(train_data_encoded), label = train_data$charges)
xgb_test <- xgb.DMatrix(data = as.matrix(test_data_encoded), label = test_data$charges)
```

```r
# Validate Data Consistency
if (nrow(as.matrix(train_data_encoded)) != length(train_data$charges)) {
  stop("Mismatch between features and labels in train_data.")
}
if (nrow(as.matrix(test_data_encoded)) != length(test_data$charges)) {
  stop("Mismatch between features and labels in test_data.")
}

# 2. Bayesian Optimization Function for XGBoost
xgb_bayesian <- function(max_depth, eta, subsample, colsample_bytree) {
  params <- list(
    objective = "reg:squarederror",
    max_depth = as.integer(max_depth),
    eta = eta,
    subsample = subsample,
    colsample_bytree = colsample_bytree,
    nthread = 4  # Enable parallel processing
  )

  # Cross-validation
  xgb_cv <- xgb.cv(
    params = params,
    data = xgb_train,
    nrounds = 500,  # Reasonable max boosting rounds
    nfold = 3,      # Reduce folds for faster evaluation
    metrics = "rmse",
    early_stopping_rounds = 20,
    verbose = 0
  )

  # Handle edge case for missing best_iteration
  best_nrounds <- ifelse(is.null(xgb_cv$best_iteration), 500, xgb_cv$best_iteration)

  # Train the model with the best parameters
  model <- xgb.train(
    params = params,
    data = xgb_train,
    nrounds = best_nrounds,
    verbose = 0
  )

  # Predictions and RMSE
  predictions <- predict(model, xgb_test)
  rmse <- sqrt(mean((test_data$charges - predictions)^2))

  list(Score = -rmse, Pred = model, Nrounds = best_nrounds)
}

# 3. Perform Bayesian Optimization
set.seed(42)
xgb_bo <- BayesianOptimization(
  FUN = xgb_bayesian,
  bounds = list(
```

```
    max_depth = c(3L, 10L),
    eta = c(0.01, 0.3),
    subsample = c(0.7, 1),
    colsample_bytree = c(0.7, 1)
  ),
  init_points = 5,
  n_iter = 20,
  acq = "ucb"
)
```

```
## elapsed = 6.23   Round = 1   max_depth = 9.0000  eta = 0.1605378 subsample = 0.8373225   colsample_by
## elapsed = 3.30   Round = 2   max_depth = 10.0000 eta = 0.2236106 subsample = 0.9157337   colsample_by
## elapsed = 5.25   Round = 3   max_depth = 5.0000  eta = 0.04905331    subsample = 0.9804017   colsampl
## elapsed = 3.04   Round = 4   max_depth = 9.0000  eta = 0.2005278 subsample = 0.7766286   colsample_by
## elapsed = 7.60   Round = 5   max_depth = 7.0000  eta = 0.2144688 subsample = 0.8386878   colsample_by
## elapsed = 7.42   Round = 6   max_depth = 9.0000  eta = 0.0100    subsample = 0.9909259   colsample_by
## elapsed = 1.99   Round = 7   max_depth = 3.0000  eta = 0.3000    subsample = 0.7000  colsample_bytree
## elapsed = 7.90   Round = 8   max_depth = 9.0000  eta = 0.06766773    subsample = 0.813348    colsampl
## elapsed = 1.85   Round = 9   max_depth = 10.0000 eta = 0.2517071 subsample = 0.7000  colsample_bytree
## elapsed = 2.81   Round = 10  max_depth = 10.0000 eta = 0.3000    subsample = 0.9932048   colsample_by
## elapsed = 1.94   Round = 11  max_depth = 9.0000  eta = 0.3000    subsample = 0.7000  colsample_bytree
## elapsed = 3.30   Round = 12  max_depth = 3.0000  eta = 0.0100    subsample = 0.7000  colsample_bytree
## elapsed = 1.96   Round = 13  max_depth = 9.0000  eta = 0.3000    subsample = 0.9838869   colsample_by
## elapsed = 3.35   Round = 14  max_depth = 3.0000  eta = 0.0100    subsample = 0.91239 colsample_bytree
## elapsed = 4.55   Round = 15  max_depth = 5.0000  eta = 0.3000    subsample = 0.9349806   colsample_by
## elapsed = 6.40   Round = 16  max_depth = 7.0000  eta = 0.0100    subsample = 0.764681    colsample_by
## elapsed = 5.74   Round = 17  max_depth = 6.0000  eta = 0.0100    subsample = 0.935249    colsample_by
## elapsed = 8.50   Round = 18  max_depth = 10.0000 eta = 0.0100    subsample = 0.9794313   colsample_by
## elapsed = 3.31   Round = 19  max_depth = 10.0000 eta = 0.3000    subsample = 0.7000  colsample_bytree
## elapsed = 1.81   Round = 20  max_depth = 8.0000  eta = 0.3000    subsample = 0.9905041   colsample_by
## elapsed = 4.38   Round = 21  max_depth = 7.0000  eta = 0.2894137 subsample = 0.9933751   colsample_by
## elapsed = 3.45   Round = 22  max_depth = 3.0000  eta = 0.0100    subsample = 0.7000  colsample_bytree
## elapsed = 1.85   Round = 23  max_depth = 3.0000  eta = 0.3000    subsample = 0.7000  colsample_bytree
## elapsed = 3.89   Round = 24  max_depth = 10.0000 eta = 0.2146027 subsample = 0.7881165   colsample_by
## elapsed = 1.72   Round = 25  max_depth = 7.0000  eta = 0.297981  subsample = 0.8150998   colsample_by
##
##  Best Parameters Found:
## Round = 8    max_depth = 9.0000  eta = 0.06766773    subsample = 0.813348    colsample_bytree = 1.000
```

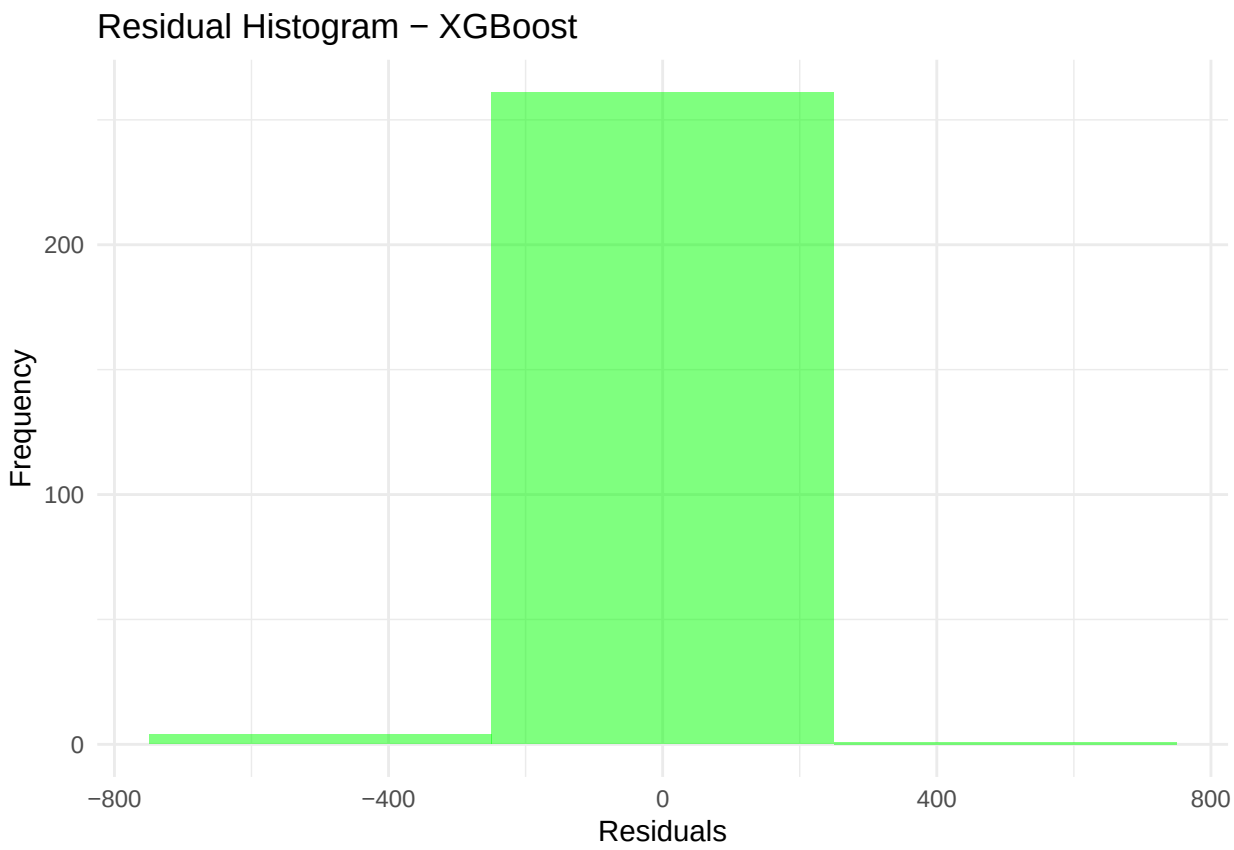
```
# Retrieve Optimal Parameters
best_params <- xgb_bo$Best_Par
best_nrounds <- ifelse(
  is.null(xgb_bo$History$Nrounds[[which.min(xgb_bo$History$Value)]]),
  500,
  xgb_bo$History$Nrounds[[which.min(xgb_bo$History$Value)]]
)

# 4. Train Final Model with Optimal Parameters
xgb_best <- xgb.train(
  params = list(
    objective = "reg:squarederror",
    max_depth = best_params["max_depth"],
    eta = best_params["eta"],
    subsample = best_params["subsample"],
```

```
    colsample_bytree = best_params["colsample_bytree"]
  ),
  data = xgb_train,
  nrounds = best_nrounds,
  verbose = 0
)

# 5. Evaluate the Final Model
xgb_predictions <- predict(xgb_best, xgb_test)
xgb_rmse <- sqrt(mean((test_data$charges - xgb_predictions)^2))
cat("Optimized XGBoost RMSE:", xgb_rmse, "\n")
```

```
## Optimized XGBoost RMSE: 76.42326
```

3.3.1 XGBoost Residuals

```
# Calculate residuals for XGBoost
residuals_xgb <- test_data$charges - xgb_predictions

# Residuals Histogram for XGBoost
ggplot(data.frame(residuals_xgb), aes(x = residuals_xgb)) +
  geom_histogram(binwidth = 500, fill = "green", alpha = 0.5) +
  labs(title = "Residual Histogram - XGBoost", x = "Residuals", y = "Frequency") +
  theme_minimal()
```



Distribution of Residuals:

The XGBoost model has a good performance overall, as residuals are centered around zero and evenly spread.

Key Takeaways:

The XGBoost model has improved prediction accuracy compared to the earlier linear regression results.

4.0 Computational Time Comparison

```
# Measure computational time for Random Forest
rf_time <- system.time({
  rf_optimized <- randomForest(
    charges ~ age + age_squared + bmi + children + smoker + region + smoker_bmi,
    data = train_data,
    mtry = rf_bo$Best_Par["mtry"],
    nodesize = rf_bo$Best_Par["min_node_size"]
  )
})

# Measure computational time for XGBoost
xgb_time <- system.time({
  xgb_best <- xgb.train(
    params = list(
      objective = "reg:squarederror",
      max_depth = best_params["max_depth"],
      eta = best_params["eta"],
      subsample = best_params["subsample"],
      colsample_bytree = best_params["colsample_bytree"]
    ),
    data = xgb_train,
    nrounds = best_nrounds
  )
})

# Display computation times
computation_times <- data.frame(
  Model = c("Random Forest", "XGBoost"),
  Time = c(round(rf_time["elapsed"], 2), round(xgb_time["elapsed"], 2))
)

kable(computation_times, caption = "Computational Time for Model Training")
```

Table 1: Computational Time for Model Training

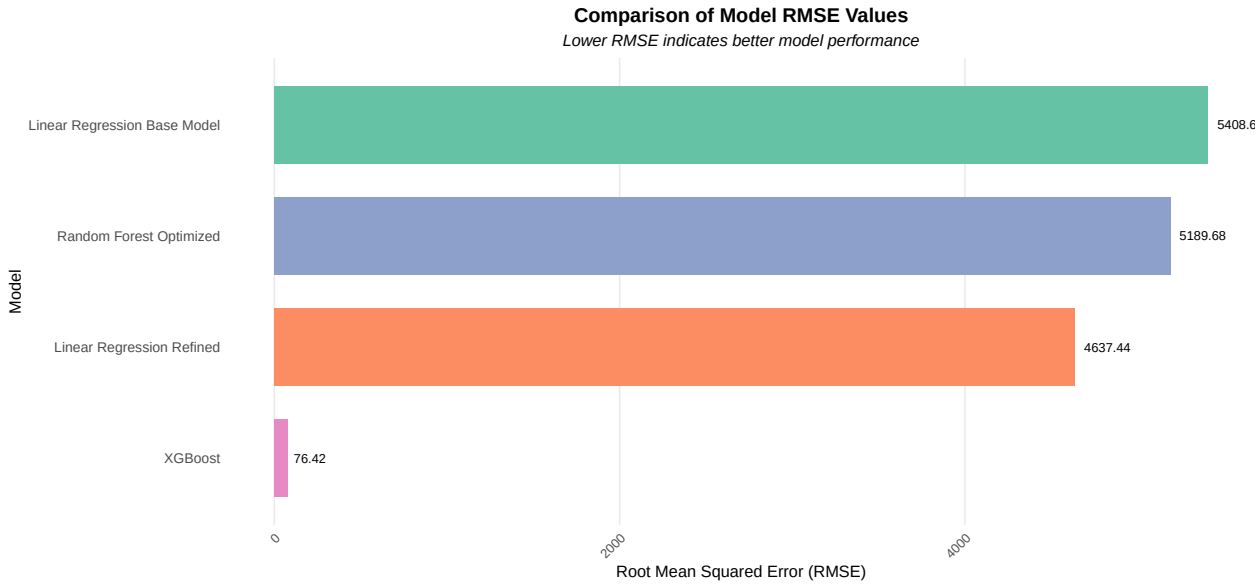
Model	Time
Random Forest	1.33
XGBoost	4.97

Comment: Though XGBoost gave the most accurate predictions with an RMSE of 76.42326, it took significantly longer to train.

4.1 Model comparison

To compare the performance of the models developed in this study, we visualize their RMSEs. This comparison helps identify the most effective approach for the medical insurance cost prediction problem.

```r
# Combine RMSE results
results <- data.frame(
  Model = c("Linear Regression Base Model","Linear Regression Refined", "Random Forest Optimized", "XGB
  RMSE = c(lm_rmse, rmse_refined, rf_rmse_optimized, xgb_rmse)
)


# Plot RMSE comparison with enhanced aesthetics
ggplot(results, aes(x = reorder(Model, RMSE), y = RMSE, fill = Model)) +
  geom_bar(stat = "identity", width = 0.7, show.legend = FALSE) + # Remove legend
  coord_flip() + # Flip coordinates for better readability of model names
  scale_fill_brewer(palette = "Set2") + # Use a pleasant color palette
  theme_minimal(base_size = 14) + # Increase base font size
  labs(
    title = "Comparison of Model RMSE Values",
    subtitle = "Lower RMSE indicates better model performance",
    x = "Model",
    y = "Root Mean Squared Error (RMSE)"
  ) +
  geom_text(aes(label = round(RMSE, 2)), hjust = -0.2, size = 4) + # Add RMSE values as labels
  theme(
    plot.title = element_text(face = "bold", hjust = 0.5), # Center title and bold
    plot.subtitle = element_text(hjust = 0.5, face = "italic"), # Center and italicize subtitle
    axis.text.x = element_text(angle = 45, hjust = 1), # Adjust x-axis text for clarity
    axis.text.y = element_text(size = 12), # Increase y-axis text size
    panel.grid.major.y = element_blank(), # Remove horizontal grid lines
    panel.grid.minor = element_blank(), # Remove minor grid lines
    panel.border = element_blank() # Remove plot border
  )
```



5.0 Recommendations for Improvement

1. **Deployment Pipeline**:
   Develop a **scalable model pipeline** that includes:
   - **Automated Data Preprocessing**: Handle missing data, transformations, and feature engi-

neering.

- **Model Integration**: Deploy the XGBoost model using tools such as **R Shiny**, **Docker**, or cloud services (e.g., AWS, Azure, or Google Cloud).

- **Monitoring and Retraining**: Establish mechanisms to monitor model performance over time and retrain as needed to adapt to new data.

2. **Interpretability for Stakeholders**:
   - Use tools like **SHAP** or **LIME** to explain individual predictions.

   - Generate clear, interpretable reports for non-technical stakeholders, highlighting the impact of key predictors like smoking status and BMI.
3. **Cross-Validation for Robustness**:
   - Use **K-fold cross-validation** or other robust validation techniques to evaluate model performance and ensure generalizability.
4. **Handling Model Bias**:
   - Perform fairness checks to ensure the model does not unintentionally discriminate against specific demographic groups (e.g., regions, age brackets).
5. **Ensemble Models**:
   Combine predictions from XGBoost, Random Forest, and other models using **stacking** or **weighted averaging** to further reduce RMSE and improve accuracy.

5.1 Future Directions

1. **Real-World Validation**:
   Collaborate with healthcare organizations to access real-world datasets that reflect actual patient and policyholder data. Validate the current model using this data to assess its robustness and applicability.
2. **Dynamic Data Integration**:
   - Incorporate real-time healthcare cost data, demographics, or socioeconomic indicators to create models that adapt to changing trends in medical costs.

   - Integrate **external APIs** for live data feeds when building real-world applications.
3. **Model Generalizability**:
   Evaluate the model's performance across various healthcare systems, geographic regions, and populations. A focus on **transfer learning** or **domain adaptation** techniques may improve generalizability across datasets.
4. **Feature Expansion**:
   Include additional predictors such as:
   - Patient comorbidities or medical history.

   - Lifestyle factors (e.g., exercise frequency, diet).

   - Policy-specific attributes (e.g., coverage details, insurance type).
5. **Advanced Algorithms**:
   Explore other state-of-the-art models, such as:
   - **LightGBM** or **CatBoost** for faster boosting-based predictions.

   - **Neural Networks** for capturing highly complex, nonlinear relationships.
6. **Cost-Optimization Analysis**:
   Develop models to predict not only costs but also identify cost-saving opportunities for healthcare

providers and insurers based on patient data.

6.0 Conclusion

This study provides a solid foundation for predicting medical insurance costs using machine learning models. While **XGBoost** achieved the lowest RMSE, the deployment of such models into practical healthcare and insurance settings requires further validation using real-world data.

By incorporating external features, validating across diverse populations, and ensuring fairness and interpretability, these models can deliver significant value to stakeholders in the healthcare industry. Future work should prioritize generalizability, scalability, and ethical considerations to ensure practical usability and real-world impact.

7. Contact of the Author

Dr.Michael Adu

Email: mikekay262@gmail.com

Feel free to connect with me on LinkedIn: Michael Adu