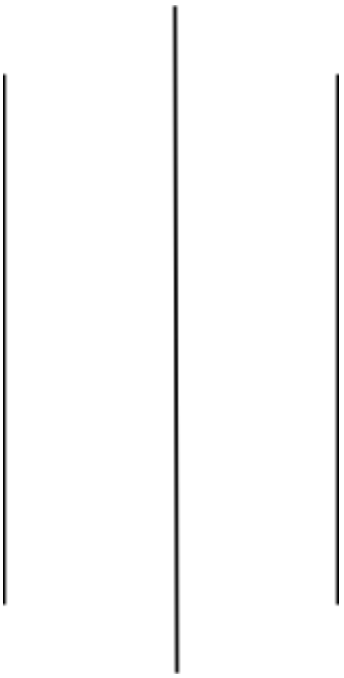
BIRMINGHAM CITY
University



Unveiling Patterns in Medical Premiums: A Random Forest-Based Exploratory Study

Utsav Rai
Student ID: 25123857
Faculty of Computing, Engineering and the Built Environment
Birmingham City University
Birmingham, United Kingdom
utsav.rai@mail.bcu.ac.uk
June 13, 2025

Table of Contents

Abstract

This study undertakes an exploratory data analysis and a predictive model to approximate yearly medical insurance premiums based on an individual's characteristics and health attributes like age, BMI, smoking, children, and area of residence. The dataset was obtained from Kaggle and the analysis was conducted using a Random Forest Regressor as it performs well with intricate data. The model determined that among all the factors, age had the greatest impact on premium prediction. The results demonstrate the capacity of machine learning in alleviating the obscurity of pricing policies and helping better decisions among the people in the healthcare insurance industry.

Keywords: *Medical Insurance Premium, Predictive Model, Random Forest Regressor, Exploratory Data Analysis, Health Attributes, Pricing Policies, Machine Learning, Kaggle, Age, Better Decisions*

INTRODUCTION

Domain Description

The health insurance industry contributes significantly to contemporary societies by reducing the financial impact associated with healthcare emergencies (Selvaraj, 2012). Various factors such as age, lifestyle, medical history, and even demographic information influence the costs of medical insurance premiums (Choi and Blackburn, 2018). According to the paper (Srinivasagopalan, 2023), analyzing datasets containing information on medical premiums, insurers are able to comprehend the possible risk outlines that pose harsh impacts, equitably price policies, and even discover some hidden patterns or disparities related to the distribution of the premiums.

Objective

This project is designed to investigate and study a given dataset containing medical insurance premiums with respect to the age, gender, medical history, BMI, smoking status, and region of the smokers and non-smokers. This will be performed using exploratory data analysis and statistical modeling. The results obtained will enhance value-based decisions in the domain of insurance policies and also unveil issues of unjust billing discrimination or overage that is sustained by citing unsubstantiated high-risk group data.

LITERATURE REVIEW

In-depth studies have reviewed elements determining the medical insurance costs. The paper (Finkelstein et al., 2009) underlined the impact of obesity on the expenditure of healthcare services, thus supporting the application of BMI as a metric within prognostic models.

Also, the study (Baicker et al., 2010) accentuated the financial implications of smoking on the health systems infrastructure, underscoring the relevance of the smoker's status as a dominant factor of health smoking costs.

In health informatics, for capturing specific non-linear interactions, (Brieman, 2009) proposed the Random Forest as an ensemble learning algorithm which partitions data into trees.

These studies demonstrate the relevance of demographic and psychographic variables to spend smart modeling simulation, justifying the application of Random Forest in the project model.

METHODOLOGY

Dataset Description

The dataset titled **Medicalpremium.csv** contains 986 records with the following columns:

- **Age:** Age of the customer
- **Diabetes:** Whether the person has got abnormal blood sugar levels
- **BloodPressure:** Whether the person has abnormal blood pressure levels
- **AnyTransplants:** Any major organ transplants
- **Height:** Height of the customer in cm
- **Weight:** Weight of the person in kg
- **KnownAllergies:** Whether the customer has any known allergies
- **HistoryOfCancerInFamily:** Whether any blood relative of the customer has had any form of cancer
- **NumberOfMajorSurgeries:** The number of major surgeries that the person has had
- **PremiumPrice:** Yearly premium price

The dataset is clean, complete, and contains no missing values.

Dataset Source

The dataset was collected from Kaggle, arguably the most popular and recognized website in the scholarly and data science community for its rich collection of datasets.

Dataset Available at:
https://www.kaggle.com/datasets/tejashvi14/medical-insurance-premium-prediction

Data Types

All features are numerical, with integer (int64) data types, making the dataset suitable for regression analysis.

The initial rows reveal a mix of binary indicators (e.g., diabetes, blood pressure problems) and continuous variables (e.g., age, height, weight). The premium price ranges from Indian Rupees (INR) 15,000 to 39,000.

Data Quality Assessment

Code snippet:

```python
# Checking for duplicates
initial_duplicates = df.duplicated().sum()
print(f"\nNumber of duplicates before processing: {initial_duplicates}")

# Checking for duplicates after preprocessing
final_duplicates = df.duplicated().sum()
print(f"\nNumber of duplicates after processing: {final_duplicates}")

# Removing duplicates if any
if final_duplicates > 0:
    df = df.drop_duplicates()
    print(f"Removed {final_duplicates} duplicates")

# Featuring engineering - BMI calculation
df['BMI'] = df['Weight'] / ((df['Height']/100) ** 2)

# Checking the updated dataframe
print("\n=== Data after Preprocessing ===")
print(df.head())
```

Fig.-1: Analyzing and handling the duplicates in data

The evaluation of the data quality showed that the entire dataset was clean, which means that it had no missing values for any of the variables. This is clearly indicative of very good data collection practices. The lack of null values meant that no imputation strategies had to be devised; hence, all records could be analyzed directly.

Duplicate analysis was conducted at multiple levels within the preprocessing pipeline. No duplicates were found so the process of removal of duplicate data wasn’t required.

The first evaluation employed descriptive statistics to illustrate the differences among the groups. The age of respondents was in line with normal insurance demographics, height and weight were average, and the count of major surgeries per person was right skewed, indicating that most people have few of such procedures.

Exploratory Data Analysis

Distribution Analysis

Code snippet:

```
# Basic statistics
print("\n=== Descriptive Statistics ===")
print(df.describe())

# Correlation analysis
print("\n=== Correlation Matrix ===")
corr_matrix = df.corr()
print(corr_matrix['PremiumPrice'].sort_values(ascending=False))

# Visualize data distribution
numeric_cols = ['Age', 'Height', 'Weight', 'NumberOfMajorSurgeries', 'BMI', 'PremiumPrice']
plt.figure(figsize=(15, 10))
for i, col in enumerate(numeric_cols, 1):
    plt.subplot(3, 2, i)
    sns.histplot(df[col], kde=True, bins=30)
    plt.title(f'Distribution of {col}')
plt.tight_layout()
plt.show()
print("-----------------------------------------------")
```

Fig.-2: Visualization of data distribution

The by-group examination of quantitative variables brought forth important characteristics of the dataset. Age exhibited almost normal distribution indicating all age groups were represented. BMI, height, and weight reflected the expected health trends of the population.

Premium prices also demonstrated customary insurance practice, with a concentration of policies sold at low premiums and fewer policies sold at high premiums. This is in support of the hypothesis that people who seek expensive coverage tend to be higher-risk individuals.

Binary Variable Analysis

Code snippet:

```
# Identifying binary columns (0/1 values)
binary_cols = [col for col in df.columns if sorted(df[col].dropna().unique()) == [0, 1]]

colors = ['#66b3ff', '#99ff99']
explode = (0.05, 0.1)  # Slightly pulling out both slices for emphasis

for col in binary_cols:
    plt.figure(figsize=(6,6))
    counts = df[col].value_counts().sort_index()
    total = counts.sum()

    # Pie chart without labels, only autopct for percentage inside slices
    wedges, texts, autotexts = plt.pie(
        counts,
        autopct='%1.1f%%',
        explode=explode,
        shadow=True,
        startangle=90,
        colors=colors,
        textprops={'fontsize': 12, 'weight': 'bold', 'color': 'black'}
    )

    # Creating legend labels with count and percentage
    legend_labels = [f'0: {counts[0]} ({counts[0]/total*100:.1f}%)',
                     f'1: {counts[1]} ({counts[1]/total*100:.1f}%)']

    plt.title(f'Distribution of {col} (Binary)', fontsize=16, weight='bold')
    plt.axis('equal')  # Equal aspect ratio ensures pie is circular
    plt.legend(wedges, legend_labels, title="Value", loc="center left", bbox_to_anchor=(1, 0, 0.5, 1))

    plt.show()
    print("-----------------------------------------------")
```

Fig.-3: Identification, Analysis and Visualization of Binary Variable

The dataset includes binary variables that capture different health conditions and lifestyle choices. Some of these factors showed different prevalence rates; some exhibited nearly equal distribution while others showed clear majority-minority patterns. These discrepancies are common in clinical datasets where some conditions dominate within the population.

The depiction of the binary variables using pie charts enabled quick assessment of condition prevalence and aided in understanding the underlying structure of the dataset, as well as class imbalance issues that could hinder effective predictive modeling.

Correlation Analysis

Code snippet:

```
# Correlation analysis
print("\n=== Correlation Matrix ===")
corr_matrix = df.corr()
print(corr_matrix['PremiumPrice'].sort_values(ascending=False))

# Visualizing correlation
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', center=0)
plt.title('Correlation Matrix')
plt.show()
```

Fig.-4: Analysis and Visualization of Correlation

The relationships of various metrics to premium pricing have been found to be significant in a correlation analysis. Correlation matrix showed that age alongside weight and the number of major surgeries were positively correlated to premium costs computationally, which echoes the relations used in the insurance industry's risk evaluation models.

The heatmap visual representation not only identified some relationships but also confirmed others. The correlations observed between the associated variables (height, weight, and BMI) being strongly interrelated verified the rationality of the datasets, while correlations of the health indicators with premium pricing validated the rationale of insurance underwriting pricing systems.

Outlier Detection & Handling

Initial Outlier Assessment

Code snippet:

```
# Outlier detection before handling
print("\n=== Outlier Detection ===")
numeric_cols = ['Age', 'Height', 'Weight', 'NumberOfMajorSurgeries', 'PremiumPrice']

plt.figure(figsize=(15, 8))
for i, col in enumerate(numeric_cols, 1):
    plt.subplot(2, 3, i)
    sns.boxplot(y=df[col])
    plt.title(f'Boxplot of {col}')
plt.tight_layout()
plt.show()

# Calculating number of outliers using Z-score
print("\nNumber of outliers before handling (Z-score > 3):")
for col in numeric_cols:
    z_scores = stats.zscore(df[col])
    outliers = np.where(np.abs(z_scores) > 3)
    print(f"{col}: {len(outliers[0])}")
```

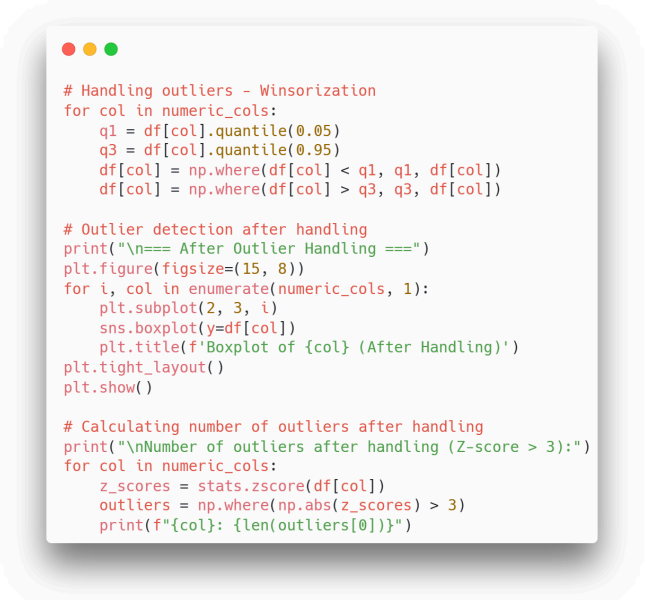
Fig.-5: Detection of Outliers using Z-score

Outlier detection with Z-score methodology ((threshold > 3) and extreme values across numerical variables) showed different patterns for age, height, weight, number of major surgeries, and premium price. The metrics provided diverse outlier patterns because of the insurance data's complex nature where extreme cases are common.

Boxplot form visualizations worked well for revealing powerful extreme values in each variable and outlier distribution comparison. These forms enhanced distinguishing features leading to differences between real outliers and ordinary discrepancies found in datasets.

Winsorization Treatment

Code snippet:



```
# Handling outliers - Winsorization
for col in numeric_cols:
    q1 = df[col].quantile(0.05)
    q3 = df[col].quantile(0.95)
    df[col] = np.where(df[col] < q1, q1, df[col])
    df[col] = np.where(df[col] > q3, q3, df[col])

# Outlier detection after handling
print("\n=== After Outlier Handling ===")
plt.figure(figsize=(15, 8))
for i, col in enumerate(numeric_cols, 1):
    plt.subplot(2, 3, i)
    sns.boxplot(y=df[col])
    plt.title(f'Boxplot of {col} (After Handling)')
plt.tight_layout()
plt.show()

# Calculating number of outliers after handling
print("\nNumber of outliers after handling (Z-score > 3):")
for col in numeric_cols:
    z_scores = stats.zscore(df[col])
    outliers = np.where(np.abs(z_scores) > 3)
    print(f"{col}: {len(outliers[0])}")
```

Fig-6: Handling Outliers using Winsorization Treatment and Rechecking

Instead of eliminating outliers as was done previously, winsorization was applied to trim extreme values at the 5th and 95th percentiles. This method maintained data points lower than the extreme values that potentially biased statistical analyses or models.

The winsorization process maintained the dataset's size and its representativeness while significantly reducing extreme outliers across all variables. The boxplots prepared after the treatment were clearer than before, exhibiting the desired shapes of distributions without losing essential information.

Feature Engineering

Code snippet:

Fig.-7: Feature Engineering - Calculation of BMI

The algorithms performed feature engineering by calculating BMI from a person's height and weight, which represents a critical step as it serves as a composite metric of their health. This derived feature performed better than others and was accepted in the medical and insurance industries.

The standard medical formulas for BMI were applied, which is consistent with healthcare practice. This step added analytical value to the dataset by calculating indicator features relevant to health insurance underwriting and, therefore, premium assessment.

Predictive Modeling

Model Selection and Training

Code snippet:

```python
# Preparing data for modeling
X = df.drop('PremiumPrice', axis=1)
y = df['PremiumPrice']

# Spliting data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Defining hyperparameter grid for Random Forest
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2]
}

# Initializing Random Forest Regressor
rf = RandomForestRegressor(random_state=42)

# Initializing GridSearchCV with 5-fold cross-validation
grid_search = GridSearchCV(estimator=rf, param_grid=param_grid,
                           cv=5, n_jobs=-1, scoring='r2', verbose=1)

# Fitting GridSearchCV to training data
grid_search.fit(X_train, y_train)

# Best estimator from grid search
best_model = grid_search.best_estimator_
print(f"\nBest hyperparameters: {grid_search.best_params_}")

# Making predictions with the best model
y_pred = best_model.predict(X_test)

# Evaluating the best model
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print("\n=== Model Evaluation After Hyperparameter Tuning ===")
print(f"Mean Squared Error: {mse:.2f}")
print(f"Root Mean Squared Error: {rmse:.2f}")
print(f"Accuracy (R^2 Score): {r2:.4f}")

# Getting feature importances from the best model
feature_importances = pd.DataFrame({
    'Feature': X.columns,
    'Importance': best_model.feature_importances_
}).sort_values(by='Importance', ascending=False)

print("\nFeature Importances:")
print(feature_importances)
```

Fig.-8: Random Forest Regression Model Training, Evaluation, and Feature Importance Analysis

Random Forest Regression was selected for its capability of capturing feature importance, performing well on varying distributions, and managing different data types. Its ensemble nature is well-suited for insurance data with interdependent premium factors. The dataset was divided into an 80% training set and a 20% test set.

Performance optimization with hyperparameter tuning using GridSearchCV was conducted for the number of estimators, maximum depth, minimum samples for splitting and leaf node, and positioned them as model performance criteria. Furthermore, 5-fold cross-validation improved parameter selection while mitigating overfitting.

RESULTS

Model Performance

Code snippet:

```
# Actual vs Predicted values plot
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'k--',
plt.xlabel('Actual Premium Price')
plt.ylabel('Predicted Premium Price')
plt.title('Actual vs Predicted Premium Prices')
plt.show()
```

Fig.-9: Visualization of Model Performance

The Random Forest model performed strongly, achieving an R^2 score of 0.8990, which indicates high explained variance in the premium prices. The model's RMSE of 1991.18 for the range of 15,000 to 39,000 demonstrates the model's accuracy and practical prediction capability in real-world scenarios.

Analysis related to feature importance demonstrated the most relevant predictors premium estimation. Undoubtedly, age, transplant history and weight were the top three most relevant predictors which validates their importance in an insurers risk assessment. Binary Health condition indicators displayed additional impacts towards premium estimation as well.

Model Validation

Code Snippet:

```
# Residual plot
residuals = y_test - y_pred
plt.figure(figsize=(10, 6))
plt.scatter(y_pred, residuals, alpha=0.5)
plt.axhline(y=0, color='r', linestyle='-')
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')
plt.title('Residual Plot')
plt.show()
print("-------------------------------------------------------------------------------------------------------
-----------------------------------")
# Distribution of residuals
plt.figure(figsize=(10, 6))
sns.histplot(residuals, kde=True)
plt.xlabel('Residuals')
plt.title('Distribution of Residuals')
plt.show()
print("-------------------------------------------------------------------------------------------------------
-----------------------------------")
# Feature importance plot
plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=feature_importances)
plt.title('Feature Importances from Random Forest')
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.show()
```

Fig.-10: Residual Analysis and Visualization of Feature Importance

Residual analysis confirmed the model appropriateness by analyzing error predictions through model evaluation. The scatter plot actual vs predicted diagram confirmed strong

linear correlation between the true and predicted values, also residual plots showcased random distribution around zero which confirms unbiased predictions.
The distribution of residuals tends to normality implying that the model does not incorporate systematic bias in estimating underlying data relationships. Therefore, this validation demonstrates the model's confirmation of fact premium pricing factors.

DISCUSSION

The examination brought to light unique insights concerning the pricing of medical premiums. Careful scrutiny uncovered age as the strongest predictor for determining insurance pricing and thus reinforcing the view that people become riskier as they get older. The transplant history also proved to be significant as it was clear that the person's medical history does impact the premiums they are charged. Weight of a person was also a determining factor of insurance pricing.

Basic binary health conditions were also useful in determining the cost of the premiums although they did not influence it equally. The model is capable of understanding the relationships of various parameters that are responsible for the mechanisms of determining prices of medical premiums.

CONCLUSIONS & IMPLICATIONS

This analysis highlights the interplay between demographic, health, and lifestyle factors for determining medical premium pricing, revealing the complex nature of medical premium pricing. The success of predictive modeling validates the reasoning behind current pricing practices and gives measurable perspectives on the importance of different factors.

Strong findings stem from the data's high quality, sound outlier handling, and model performance. Organizations can improve risk assessment, pricing, and customer targeting tactics based on the insights derived from the analysis.

These methodologies can be extrapolated to other datasets, while specific results provide strategic guidance for premium pricing and assist in refining risk management initiatives.

REFERENCES

Baicker, K., Cutler, D. and Song, Z. (2010) Workplace Wellness Programs Can Generate Savings. *Health Affairs*, 29 (2): 304–311. doi:10.1377/hlthaff.2009.0626.

Choi, S. and Blackburn, J. (2018) Patterns and Factors Associated With Medical Expenses and Health Insurance Premium Payments. *Journal of Financial Counseling and Planning*, 29 (1): 6–18. doi:10.1891/1052-3073.29.1.6.

Finkelstein, E.A., Trogdon, J.G., Cohen, J.W., et al. (2009) Annual Medical Spending

Attributable To Obesity: Payer-And Service-Specific Estimates: Amid calls for health reform, real cost savings are more likely to be achieved through reducing obesity and related risk factors. *Health Affairs*, 28 (Supplement 1): w822–w831. doi:10.1377/hlthaff.28.5.w822.

Selvaraj, S. (2012) The impact of health insurance in low- and middle-income countries. *Global Public Health*, 7 (4): 434–436. doi:10.1080/17441692.2012.660975.

Srinivasagopalan, L.N. (2023) Predicting health insurance premiums using machine learning: A novel regression-based model for enhanced accuracy and personalization. *World Journal of Advanced Research and Reviews*, 19 (1): 1580–1592. doi:10.30574/wjarr.2023.19.1.1355.

Breiman, L., 2001. *Random forests*. Machine Learning, 45(1), pp.5–32. Available at: https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf [Accessed 5 June 2025].

APPENDICES

Full code with outputs available at
https://gist.github.com/Utu8848/ff39f4c4975fe7b0af32868b2ca76b26

Appendix-A: Initial Data Loading and Basic Exploration of Dataset

Code snippet:

```python
# Importing necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import GridSearchCV
from scipy import stats

# Loading the dataset
df = pd.read_csv('Medicalpremium.csv')

# Displaying basic information
print("=== Dataset Information ===")
print(f"Shape: {df.shape}")
print(f"\nColumns: {df.columns.tolist()}")
print(f"\nData types:\n{df.dtypes}")
print("\nFirst 5 rows:")
print(df.head())
```

Output:

```
=== Dataset Information ===
Shape: (986, 11)

Columns: ['Age', 'Diabetes', 'BloodPressureProblems', 'AnyTransplants', 'AnyChronicDiseases', 'Height',
'Weight', 'KnownAllergies', 'HistoryOfCancerInFamily', 'NumberOfMajorSurgeries', 'PremiumPrice']

Data types:
Age                       int64
Diabetes                  int64
BloodPressureProblems     int64
AnyTransplants            int64
AnyChronicDiseases        int64
Height                    int64
Weight                    int64
KnownAllergies            int64
HistoryOfCancerInFamily   int64
NumberOfMajorSurgeries    int64
PremiumPrice              int64
dtype: object

First 5 rows:
   Age  Diabetes  BloodPressureProblems  AnyTransplants  AnyChronicDiseases  \
0   45         0                      0               0                   0
1   60         1                      0               0                   0
2   36         1                      1               0                   0
3   52         1                      1               0                   1
4   38         0                      0               0                   1

   Height  Weight  KnownAllergies  HistoryOfCancerInFamily  \
0     155      57               0                        0
1     180      73               0                        0
2     158      59               0                        0
3     183      93               0                        0
4     166      88               0                        0

   NumberOfMajorSurgeries  PremiumPrice
0                       0         25000
1                       0         29000
2                       1         23000
3                       2         28000
4                       1         23000
```

Appendix-B: Duplicate Removal and Feature Engineering: Calculating BMI

Code snippet:

```python
# Checking for duplicates
initial_duplicates = df.duplicated().sum()
print(f"\nNumber of duplicates before processing: {initial_duplicates}")

# Checking for duplicates after preprocessing
final_duplicates = df.duplicated().sum()
print(f"\nNumber of duplicates after processing: {final_duplicates}")

# Removing duplicates if any
if final_duplicates > 0:
    df = df.drop_duplicates()
    print(f"Removed {final_duplicates} duplicates")

# Featuring engineering - BMI calculation
df['BMI'] = df['Weight'] / ((df['Height']/100) ** 2)

# Checking the updated dataframe
print("\n=== Data after Preprocessing ===")
print(df.head())
```

Output:

```
Number of duplicates before processing: 0

Number of duplicates after processing: 0

=== Data after Preprocessing ===
   Age  Diabetes  BloodPressureProblems  AnyTransplants  AnyChronicDiseases  \
0   45         0                      0               0                   0
1   60         1                      0               0                   0
2   36         1                      1               0                   0
3   52         1                      1               0                   1
4   38         0                      0               0                   1

   Height  Weight  KnownAllergies  HistoryOfCancerInFamily  \
0     155      57               0                        0
1     180      73               0                        0
2     158      59               0                        0
3     183      93               0                        0
4     166      88               0                        0

   NumberOfMajorSurgeries  PremiumPrice        BMI
0                       0         25000  23.725286
1                       0         29000  22.530864
2                       1         23000  23.634033
3                       2         28000  27.770313
4                       1         23000  31.934969
```

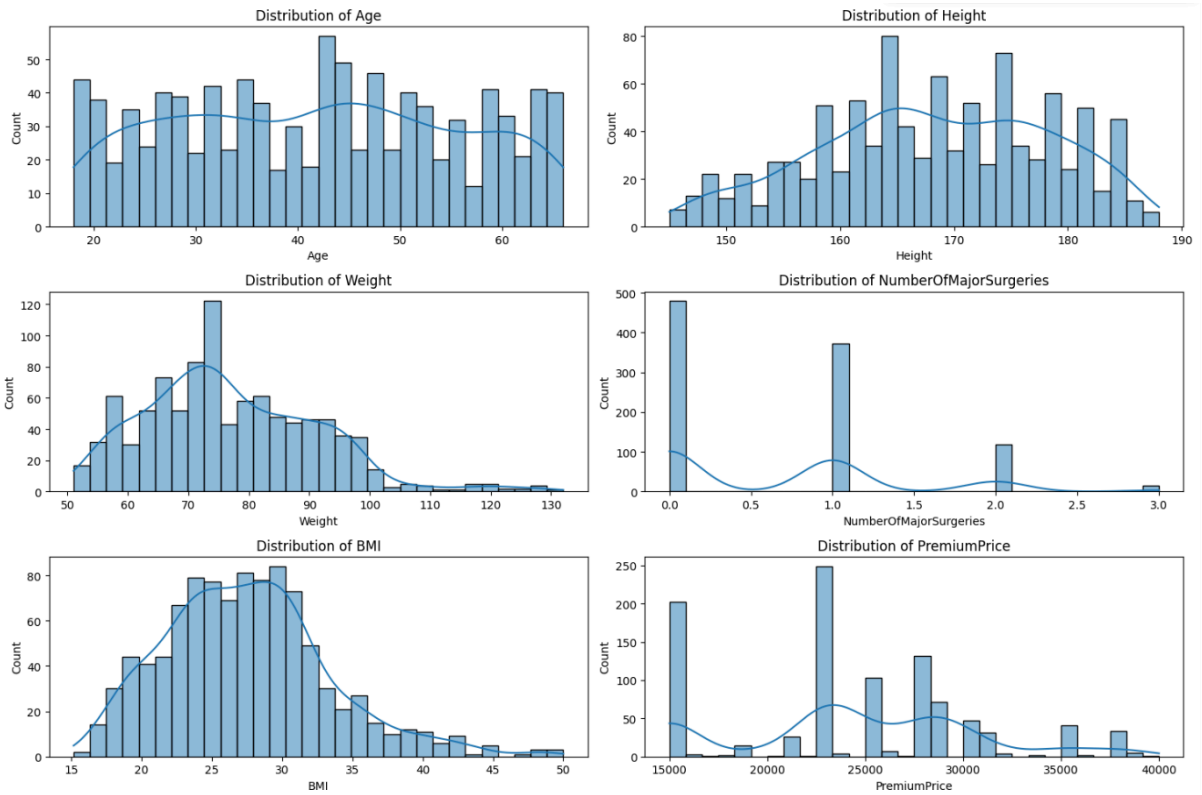
Appendix-C: Data Quality Assessment and Analysis

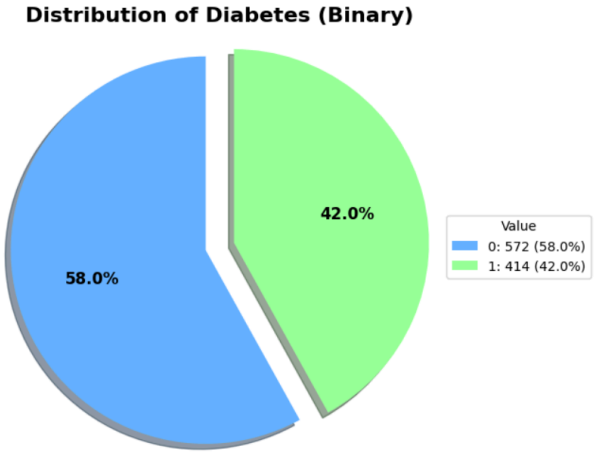
Code snippet:

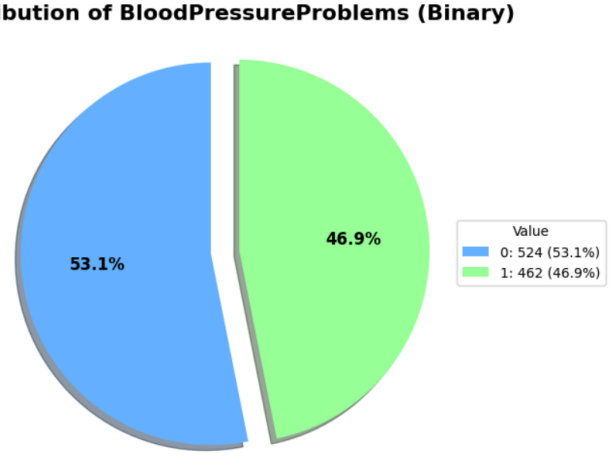
```python
# Checking for missing values
print("\n=== Missing Values ===")
print(df.isnull().sum())

# Basic statistics
print("\n=== Descriptive Statistics ===")
print(df.describe())

# Correlation analysis
print("\n=== Correlation Matrix ===")
corr_matrix = df.corr()
print(corr_matrix['PremiumPrice'].sort_values(ascending=False))

# Visualize data distribution
numeric_cols = ['Age', 'Height', 'Weight', 'NumberOfMajorSurgeries', 'BMI', 'PremiumPrice']
plt.figure(figsize=(15, 10))
for i, col in enumerate(numeric_cols, 1):
    plt.subplot(3, 2, i)
    sns.histplot(df[col], kde=True, bins=30)
    plt.title(f'Distribution of {col}')
plt.tight_layout()
plt.show()
print("-------------------------------------------------------------------------------------------------------
------------------")

# Identifying binary columns (0/1 values)
binary_cols = [col for col in df.columns if sorted(df[col].dropna().unique()) == [0, 1]]

colors = ['#66b3ff', '#99ff99']
explode = (0.05, 0.1)  # Slightly pulling out both slices for emphasis

for col in binary_cols:
    plt.figure(figsize=(6,6))
    counts = df[col].value_counts().sort_index()
    total = counts.sum()

    # Pie chart without labels, only autopct for percentage inside slices
    wedges, texts, autotexts = plt.pie(
        counts,
        autopct='%1.1f%%',
        explode=explode,
        shadow=True,
        startangle=90,
        colors=colors,
        textprops={'fontsize': 12, 'weight': 'bold', 'color': 'black'}
    )

    # Creating legend labels with count and percentage
    legend_labels = [f'0: {counts[0]} ({counts[0]/total*100:.1f}%)',
                     f'1: {counts[1]} ({counts[1]/total*100:.1f}%)']

    plt.title(f'Distribution of {col} (Binary)', fontsize=16, weight='bold')
    plt.axis('equal')  # Equal aspect ratio ensures pie is circular
    plt.legend(wedges, legend_labels, title="Value", loc="center left", bbox_to_anchor=(1, 0, 0.5, 1))

    plt.show()
    print("-------------------------------------------------------------------------------------------------------
----------------------")

# Visualizing correlation
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', center=0)
plt.title('Correlation Matrix')
plt.show()
```

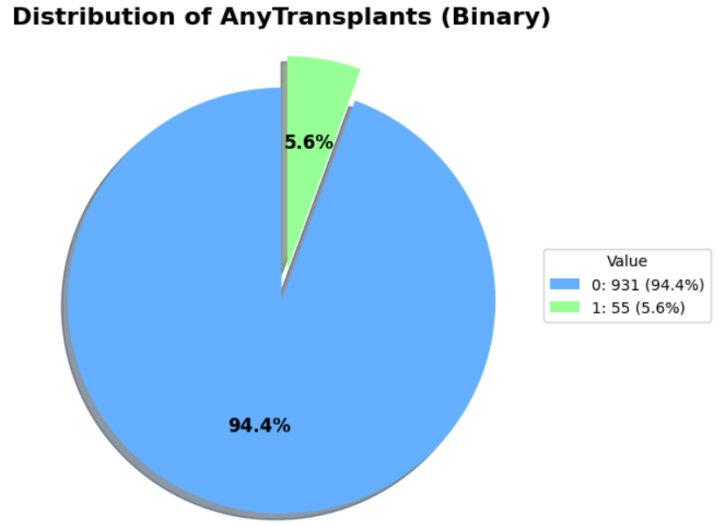
Outputs:

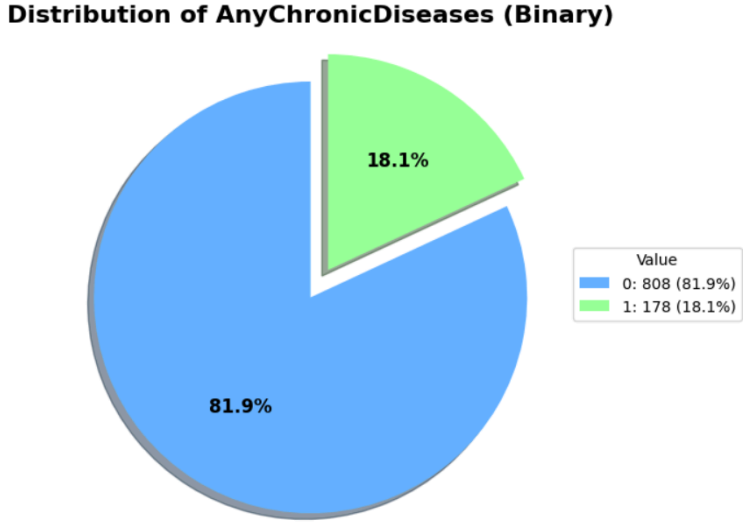
```
=== Missing Values ===
Age                          0
Diabetes                     0
BloodPressureProblems        0
AnyTransplants               0
AnyChronicDiseases           0
Height                       0
Weight                       0
KnownAllergies               0
HistoryOfCancerInFamily      0
NumberOfMajorSurgeries       0
PremiumPrice                 0
BMI                          0
dtype: int64

=== Descriptive Statistics ===
                Age    Diabetes  BloodPressureProblems  AnyTransplants  \
count  986.000000  986.000000             986.000000      986.000000
mean    41.745436    0.419878               0.468560        0.055781
std     13.963371    0.493789               0.499264        0.229615
min     18.000000    0.000000               0.000000        0.000000
25%     30.000000    0.000000               0.000000        0.000000
50%     42.000000    0.000000               0.000000        0.000000
75%     53.000000    1.000000               1.000000        0.000000
max     66.000000    1.000000               1.000000        1.000000

       AnyChronicDiseases      Height      Weight  KnownAllergies  \
count          986.000000  986.000000  986.000000      986.000000
mean             0.180527  168.182556   76.950304        0.215010
std              0.384821   10.098155   14.265096        0.411038
min              0.000000  145.000000   51.000000        0.000000
25%              0.000000  161.000000   67.000000        0.000000
50%              0.000000  168.000000   75.000000        0.000000
75%              0.000000  176.000000   87.000000        0.000000
max              1.000000  188.000000  132.000000        1.000000

       HistoryOfCancerInFamily  NumberOfMajorSurgeries  PremiumPrice  \
count               986.000000              986.000000    986.000000
mean                  0.117647                0.667343  24336.713996
std                   0.322353                0.749205   6248.184382
min                   0.000000                0.000000  15000.000000
25%                   0.000000                0.000000  21000.000000
50%                   0.000000                1.000000  23000.000000
75%                   0.000000                1.000000  28000.000000
max                   1.000000                3.000000  40000.000000

              BMI
count  986.000000
mean    27.460709
std      5.878671
min     15.156281
25%     23.393392
50%     27.156602
75%     30.759870
max     50.000000

=== Correlation Matrix ===
PremiumPrice               1.000000
Age                        0.697540
AnyTransplants             0.289056
NumberOfMajorSurgeries     0.264250
AnyChronicDiseases         0.208610
BloodPressureProblems      0.167097
Weight                     0.141507
BMI                        0.103812
HistoryOfCancerInFamily    0.083139
Diabetes                   0.076209
Height                     0.026910
KnownAllergies             0.012103
Name: PremiumPrice, dtype: float64
```

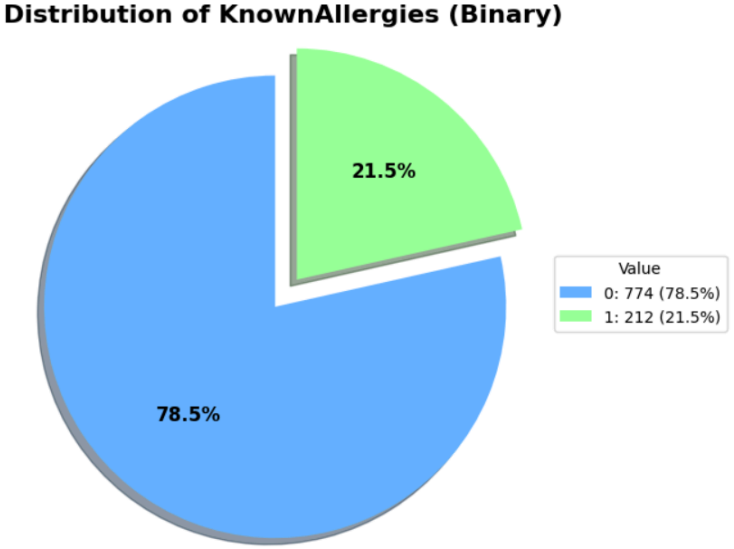

Distribution of Age
Count
Age
Distribution of Height
Count
Height
Distribution of Weight
Count
Weight
Distribution of NumberOfMajorSurgeries
Count
NumberOfMajorSurgeries
Distribution of BMI
Count
BMI
Distribution of PremiumPrice
Count
PremiumPrice

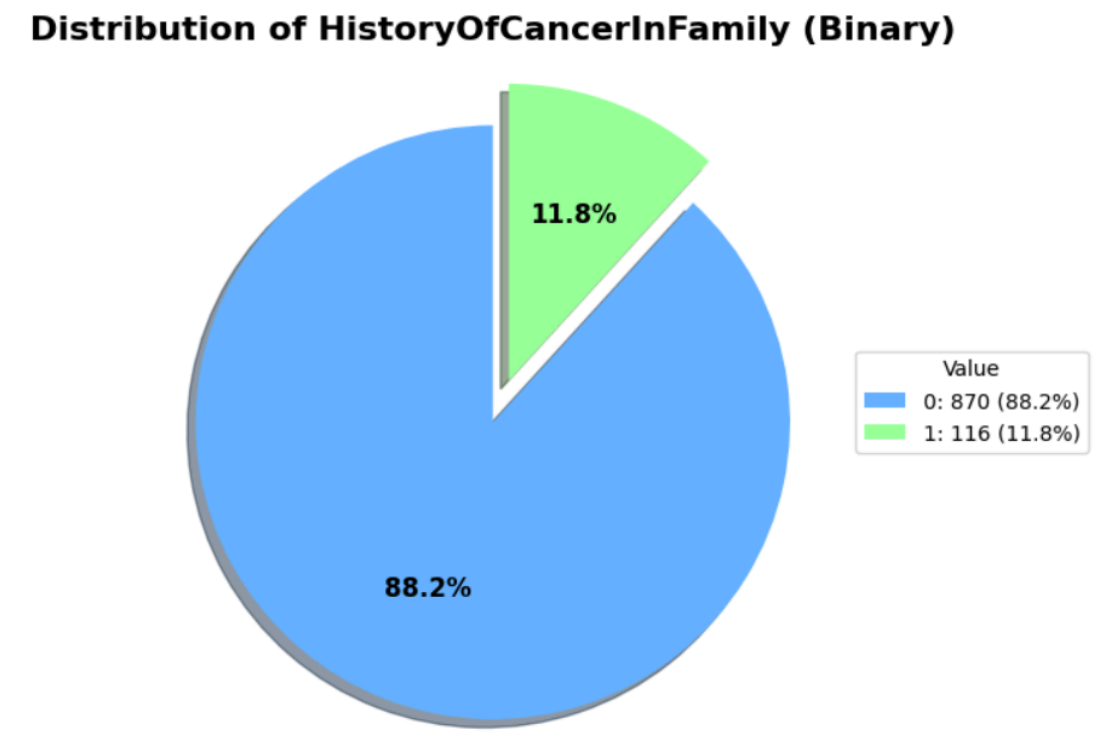

Distribution of Diabetes (Binary)
42.0%
58.0%
Value
0: 572 (58.0%)
1: 414 (42.0%)

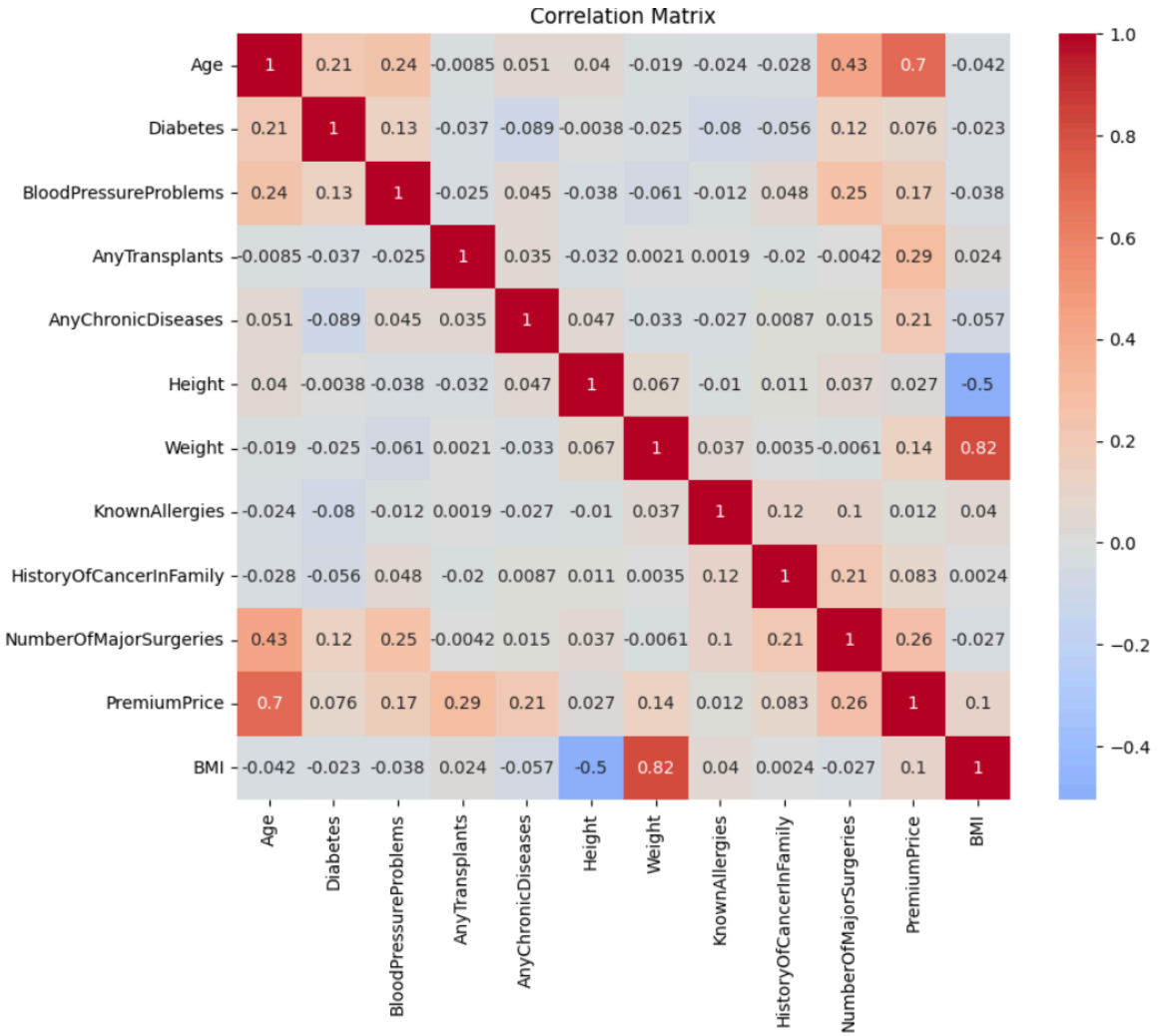

Distribution of BloodPressureProblems (Binary)
46.9%
53.1%
Value
0: 524 (53.1%)
1: 462 (46.9%)


Distribution of AnyTransplants (Binary)
5.6%
94.4%
Value
0: 931 (94.4%)
1: 55 (5.6%)


Distribution of AnyChronicDiseases (Binary)
18.1%
81.9%
Value
0: 808 (81.9%)
1: 178 (18.1%)


Distribution of KnownAllergies (Binary)
21.5%
78.5%
Value
0: 774 (78.5%)
1: 212 (21.5%)


Distribution of HistoryOfCancerInFamily (Binary)
11.8%
88.2%
Value
0: 870 (88.2%)
1: 116 (11.8%)


Correlation Matrix
Age 1 0.21 0.24 -0.0085 0.051 0.04 -0.019 -0.024 -0.028 0.43 0.7 -0.042
Diabetes 0.21 1 0.13 -0.037 -0.089 -0.0038 -0.025 -0.08 -0.056 0.12 0.076 -0.023
BloodPressureProblems 0.24 0.13 1 -0.025 0.045 -0.038 -0.061 -0.012 0.048 0.25 0.17 -0.038
AnyTransplants -0.0085 -0.037 -0.025 1 0.035 -0.032 0.0021 0.0019 -0.02 -0.0042 0.29 0.024
AnyChronicDiseases 0.051 -0.089 0.045 0.035 1 0.047 -0.033 -0.027 0.0087 0.015 0.21 -0.057
Height 0.04 -0.0038 -0.038 -0.032 0.047 1 0.067 -0.01 0.011 0.037 0.027 -0.5
Weight -0.019 -0.025 -0.061 0.0021 -0.033 0.067 1 0.037 0.0035 -0.0061 0.14 0.82
KnownAllergies -0.024 -0.08 -0.012 0.0019 -0.027 -0.01 0.037 1 0.12 0.1 0.012 0.04
HistoryOfCancerInFamily -0.028 -0.056 0.048 -0.02 0.0087 0.011 0.0035 0.12 1 0.21 0.083 0.0024
NumberOfMajorSurgeries 0.43 0.12 0.25 -0.0042 0.015 0.037 -0.0061 0.1 0.21 1 0.26 -0.027
PremiumPrice 0.7 0.076 0.17 0.29 0.21 0.027 0.14 0.012 0.083 0.26 1 0.1
BMI -0.042 -0.023 -0.038 0.024 -0.057 -0.5 0.82 0.04 0.0024 -0.027 0.1 1
Age
Diabetes
BloodPressureProblems
AnyTransplants
AnyChronicDiseases
Height
Weight
KnownAllergies
HistoryOfCancerInFamily
NumberOfMajorSurgeries
PremiumPrice
BMI
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

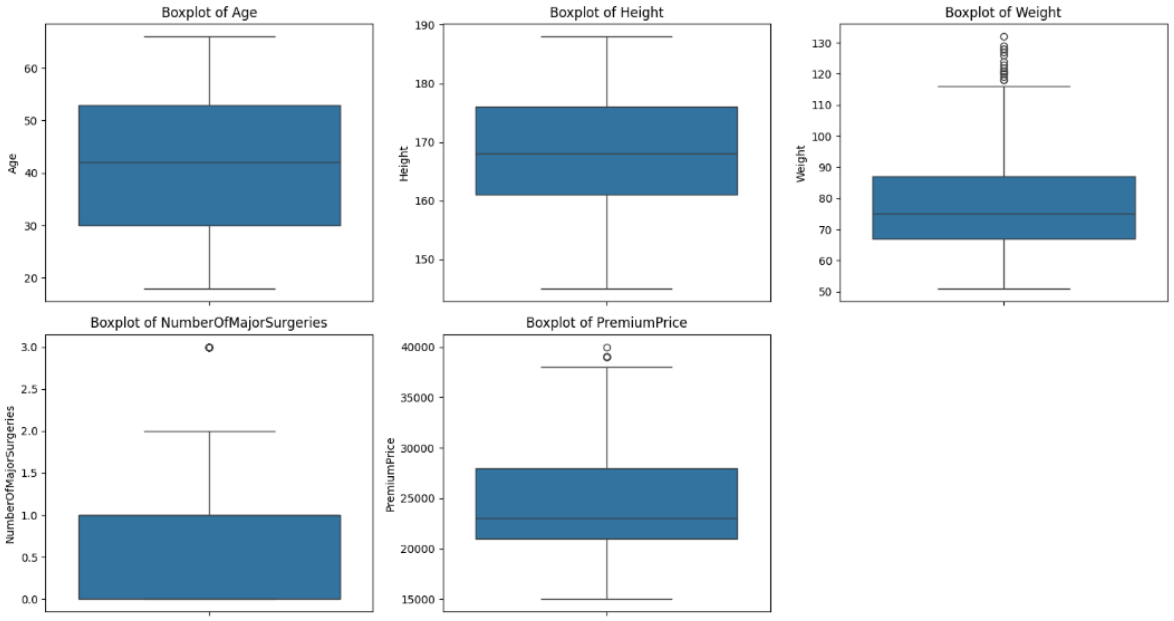
Appendix-D: Outlier Detection and Treatment using Winsorization

Code snippet:

```python
# Outlier detection before handling
print("\n=== Outlier Detection ===")
numeric_cols = ['Age', 'Height', 'Weight', 'NumberOfMajorSurgeries', 'PremiumPrice']

plt.figure(figsize=(15, 8))
for i, col in enumerate(numeric_cols, 1):
    plt.subplot(2, 3, i)
    sns.boxplot(y=df[col])
    plt.title(f'Boxplot of {col}')
plt.tight_layout()
plt.show()

# Calculating number of outliers using Z-score
print("\nNumber of outliers before handling (Z-score > 3):")
for col in numeric_cols:
    z_scores = stats.zscore(df[col])
    outliers = np.where(np.abs(z_scores) > 3)
    print(f"{col}: {len(outliers[0])}")

# Handling outliers - Winsorization
for col in numeric_cols:
    q1 = df[col].quantile(0.05)
    q3 = df[col].quantile(0.95)
    df[col] = np.where(df[col] < q1, q1, df[col])
    df[col] = np.where(df[col] > q3, q3, df[col])

# Outlier detection after handling
print("\n=== After Outlier Handling ===")
plt.figure(figsize=(15, 8))
for i, col in enumerate(numeric_cols, 1):
    plt.subplot(2, 3, i)
    sns.boxplot(y=df[col])
    plt.title(f'Boxplot of {col} (After Handling)')
plt.tight_layout()
plt.show()

# Calculating number of outliers after handling
print("\nNumber of outliers after handling (Z-score > 3):")
for col in numeric_cols:
    z_scores = stats.zscore(df[col])
    outliers = np.where(np.abs(z_scores) > 3)
    print(f"{col}: {len(outliers[0])}")
```

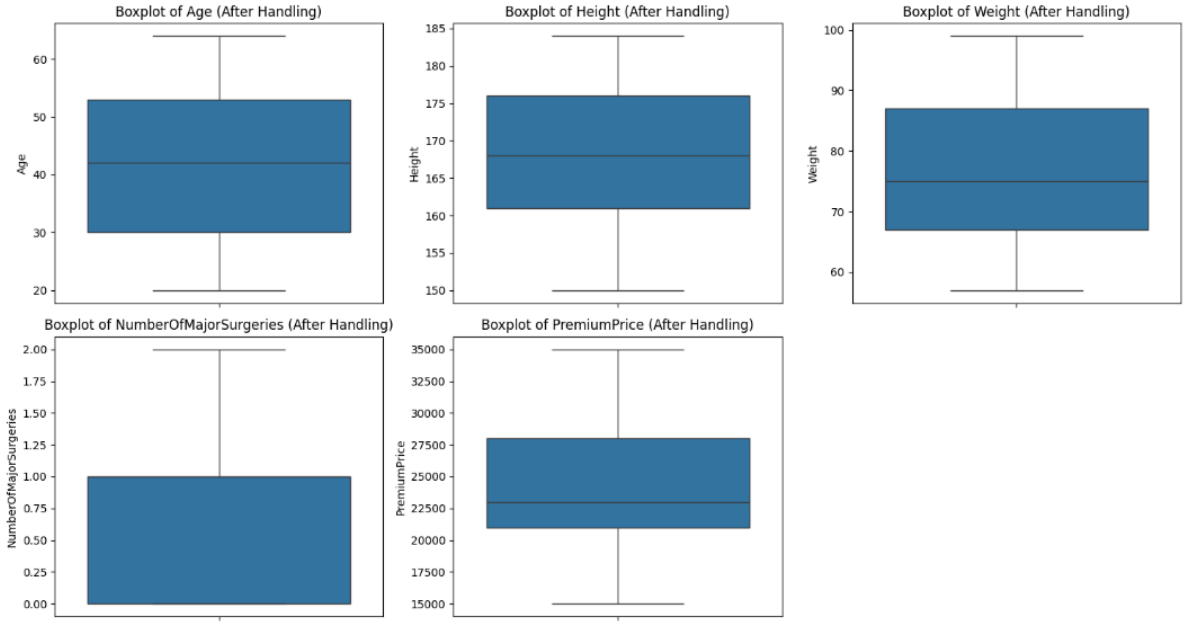
Outputs:

```
=== Outlier Detection ===
```




```
Number of outliers before handling (Z-score > 3):
Age: 0
Height: 0
Weight: 13
NumberOfMajorSurgeries: 16
PremiumPrice: 0
```

```
=== After Outlier Handling ===
```




```
Number of outliers after handling (Z-score > 3):
Age: 0
Height: 0
Weight: 0
NumberOfMajorSurgeries: 0
PremiumPrice: 0
```

Appendix-E: Random Forest Regression Model Training, Evaluation, and Feature Importance Analysis

Code snippet:

```python
# Preparing data for modeling
X = df.drop('PremiumPrice', axis=1)
y = df['PremiumPrice']

# Spliting data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Defining hyperparameter grid for Random Forest
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2]
}

# Initializing Random Forest Regressor
rf = RandomForestRegressor(random_state=42)

# Initializing GridSearchCV with 5-fold cross-validation
grid_search = GridSearchCV(estimator=rf, param_grid=param_grid,
                           cv=5, n_jobs=-1, scoring='r2', verbose=1)

# Fitting GridSearchCV to training data
grid_search.fit(X_train, y_train)

# Best estimator from grid search
best_model = grid_search.best_estimator_
print(f"\nBest hyperparameters: {grid_search.best_params_}")

# Making predictions with the best model
y_pred = best_model.predict(X_test)

# Evaluating the best model
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print("\n=== Model Evaluation After Hyperparameter Tuning ===")
print(f"Mean Squared Error: {mse:.2f}")
print(f"Root Mean Squared Error: {rmse:.2f}")
print(f"Accuracy (R^2 Score): {r2:.4f}")

# Getting feature importances from the best model
feature_importances = pd.DataFrame({
    'Feature': X.columns,
    'Importance': best_model.feature_importances_
}).sort_values(by='Importance', ascending=False)

print("\nFeature Importances:")
print(feature_importances)
```

Output:

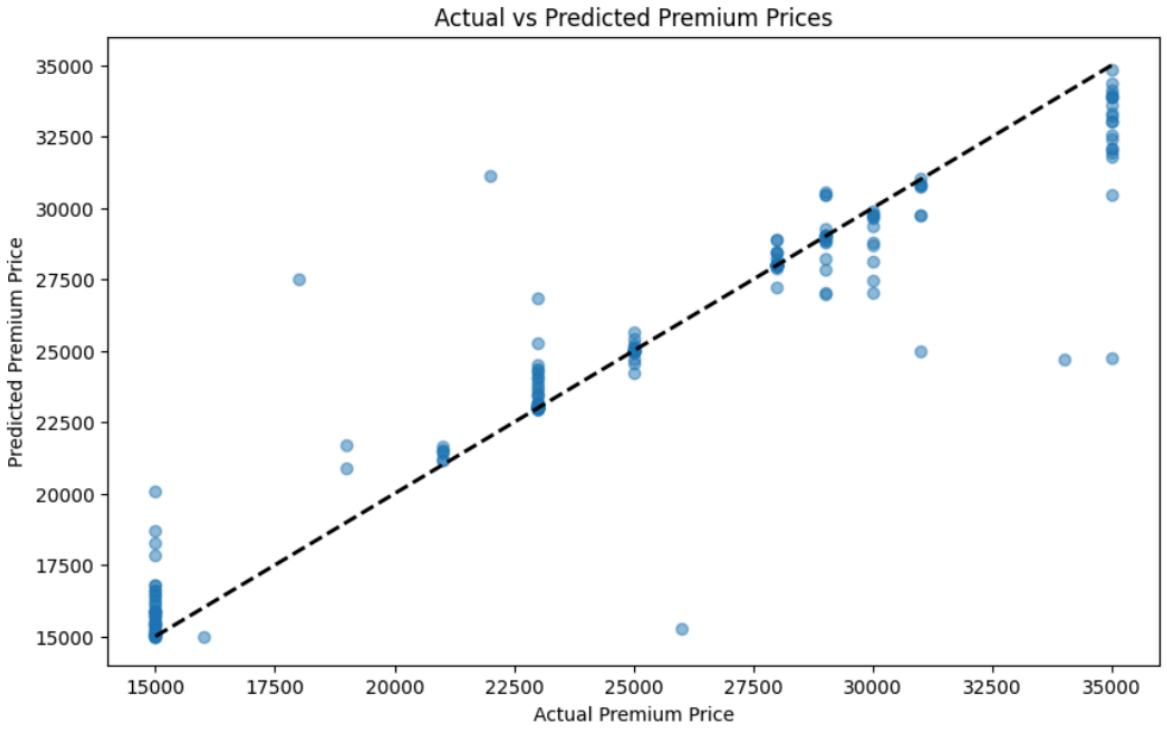
```
Fitting 5 folds for each of 24 candidates, totalling 120 fits

Best hyperparameters: {'max_depth': 10, 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators':
200}

=== Model Evaluation After Hyperparameter Tuning ===
Mean Squared Error: 3964784.56
Root Mean Squared Error: 1991.18
Accuracy (R^2 Score): 0.8990

Feature Importances:
                    Feature  Importance
0                       Age    0.700777
3           AnyTransplants    0.071362
6                    Weight    0.070653
4        AnyChronicDiseases    0.039736
10                      BMI    0.035320
9    NumberOfMajorSurgeries    0.029170
8   HistoryOfCancerInFamily    0.023267
5                    Height    0.019317
2     BloodPressureProblems    0.006089
1                  Diabetes    0.003290
7            KnownAllergies    0.001019
```

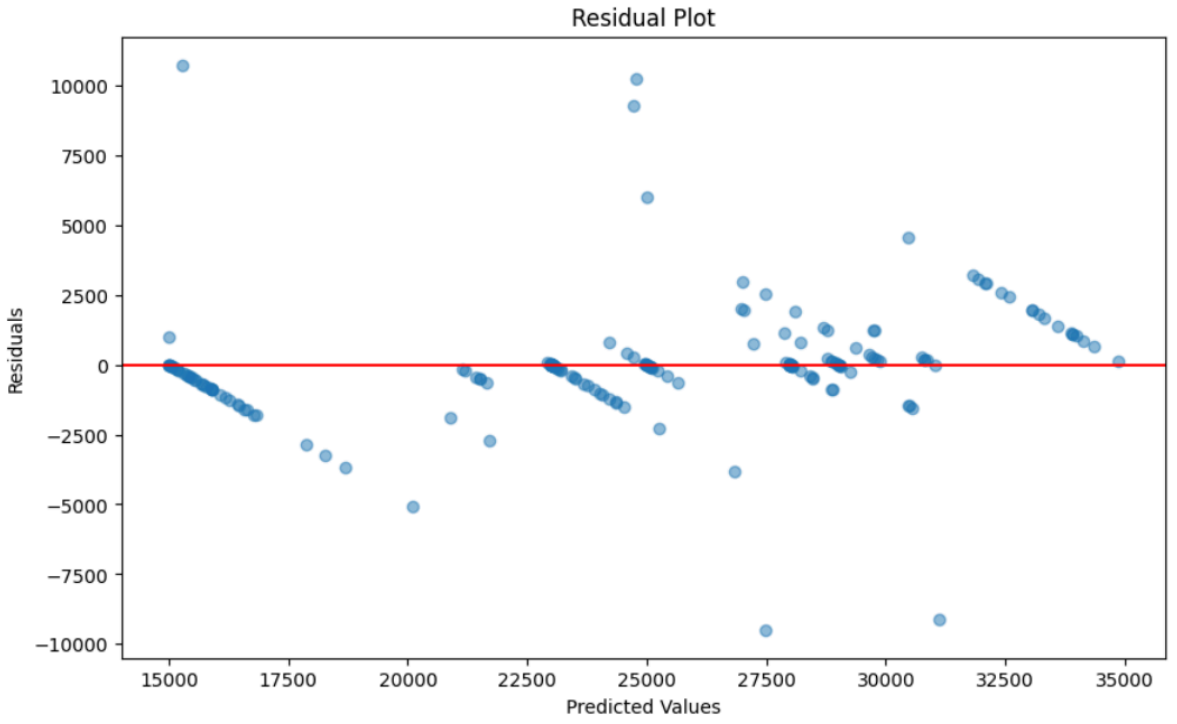
Appendix-F: Model Performance Visualization: Actual vs Predicted, Residual Analysis, and Feature Importance

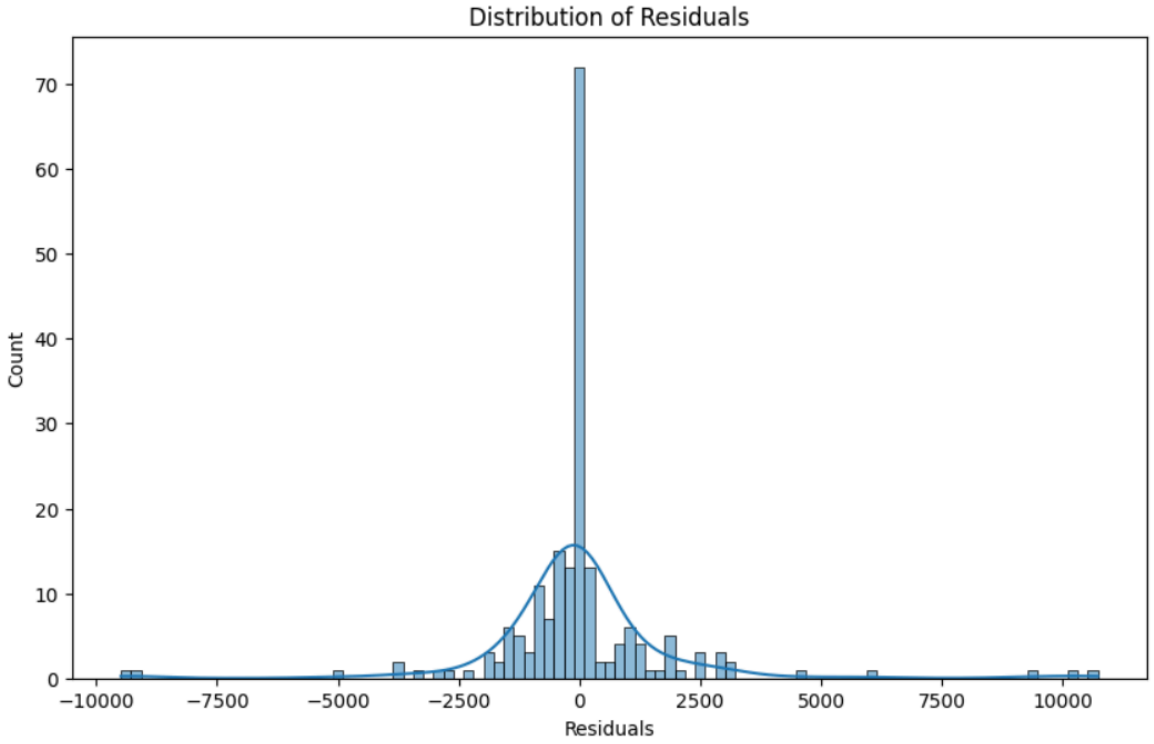
Code snippet:

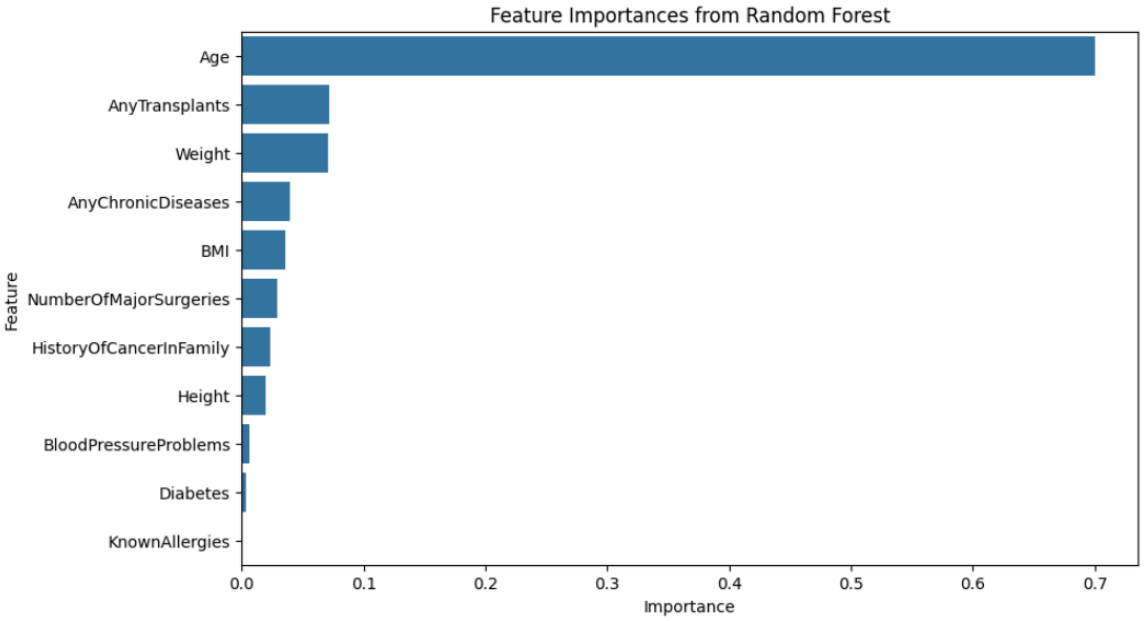
```python
# Actual vs Predicted values plot
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'k--', lw=2)
plt.xlabel('Actual Premium Price')
plt.ylabel('Predicted Premium Price')
plt.title('Actual vs Predicted Premium Prices')
plt.show()
print("-----------------------------------------------------------------------------------------------------------")
# Residual plot
residuals = y_test - y_pred
plt.figure(figsize=(10, 6))
plt.scatter(y_pred, residuals, alpha=0.5)
plt.axhline(y=0, color='r', linestyle='-')
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')
plt.title('Residual Plot')
plt.show()
print("-----------------------------------------------------------------------------------------------------------")
# Distribution of residuals
plt.figure(figsize=(10, 6))
sns.histplot(residuals, kde=True)
plt.xlabel('Residuals')
plt.title('Distribution of Residuals')
plt.show()
print("-----------------------------------------------------------------------------------------------------------")
# Feature importance plot
plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=feature_importances)
plt.title('Feature Importances from Random Forest')
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.show()
```

Outputs:




Residual Plot
Residuals
Predicted Values
10000
7500
5000
2500
0
−2500
−5000
−7500
−10000
15000
17500
20000
22500
25000
27500
30000
32500
35000


Distribution of Residuals
Count
Residuals
70
60
50
40
30
20
10
0
−10000
−7500
−5000
−2500
0
2500
5000
7500
10000


Feature Importances from Random Forest
Age
AnyTransplants
Weight
AnyChronicDiseases
BMI
NumberOfMajorSurgeries
HistoryOfCancerInFamily
Height
BloodPressureProblems
Diabetes
KnownAllergies
Feature
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Importance