Estimating Medical Insurance Premiums using Machine Learning : A Comprehensive Machine Learning Approach

(Gradient Boosting Regressor)

TEAM MEMBERS

BALAJI VENKATA SAILESH GUMMA

SAI KARTHIKEYA VARANASI

SANDEEP DAVARASINGI

SHINY SAYYID

Abstract:

This document presents a rigorous methodology for accurately predicting medical insurance premiums through advanced machine learning techniques. The process initiates with comprehensive data collection, aggregating a diverse set of medical insurance records reflecting varied demographic and health attributes. The subsequent data preprocessing phase involves addressing missing values, encoding categorical variables, and normalizing numerical features, facilitating a refined dataset split for training and testing. Employing innovative feature selection and engineering methods, pertinent predictors significantly impacting insurance premiums are identified and optimized. Model selection is a crucial step, tailoring the choice of regression algorithm to ensure both interpretability and performance. Thorough model training with hyperparameter adjustments leads to a well-tuned predictive model. Model evaluation metrics, including mean squared error and root mean squared error, are employed to rigorously assess predictive accuracy. An iterative process of fine-tuning and continuous improvement refines the model, enhancing its robustness and applicability. Finally, the trained model is deployed, enabling precise estimation of insurance premiums, fostering operational efficiency and enabling ongoing model enhancement.

Introduction:

In the realm of insurance, accurate estimation of premiums is a cornerstone for both providers and policyholders. A precise prediction of insurance premiums allows insurers to set fair and competitive pricing, ensuring the sustainability and profitability of their services. Simultaneously, policyholders benefit from an understanding of the costs involved in securing insurance coverage, aiding informed decision-making regarding their financial planning and risk management.

Machine learning, with its prowess in deriving insights from complex data, plays a pivotal role in revolutionizing the way insurance premiums are determined. By harnessing the power of algorithms and predictive models, machine learning allows insurance providers to analyze multifaceted data encompassing demographics, health attributes, and lifestyle habits. This analysis translates into a more accurate assessment of risk profiles, ultimately guiding the determination of insurance premiums.

The objective of this comprehensive guide is to illuminate the intricate process of constructing a reliable insurance premium prediction model using machine learning. From data collection and preprocessing to model training, evaluation, and deployment, this guide aims to equip the reader with a solid understanding of the essential steps involved. By the end of this journey, readers will possess the knowledge and tools to develop robust predictive models that can precisely estimate insurance premiums, advancing the efficiency and effectiveness of insurance services.

Steps used to build the model:

Import Libraries:

- Import necessary libraries, including pandas for data manipulation, scikit-learn for machine learning, and joblib for model persistence.

Read the Dataset:

- Read the medical insurance dataset using pandas from a specified file path.

Handle Missing Values:

- Drop rows with missing values from the dataset.

Encode Categorical Variables:

- Use one-hot encoding to convert categorical variables into numerical format.

Split Data into Features and Target:

- Separate the dataset into features (X) and the target variable (y).

Normalize Numerical Features:

- Use StandardScaler to standardize numerical features.

Model Selection:

- Choose Gradient Boosting Regressor as the machine learning model.

Model Training:

- Train the chosen model using the preprocessed features and target.

Cross-Validation:

- Evaluate the model's performance using cross-validation and compute mean squared error (MSE) and mean absolute error (MAE).

Continuous Improvement and Prediction:

- Define a function to predict insurance premiums for new individuals based on input features (age, gender, BMI, smoking status, pre-existing condition).

User Input and Prediction:

- Take user input for the features and use the trained model to predict the insurance premium for the provided information.

Deployment:

- Save the trained model using joblib for future use and deployment.

Understanding the Data:

The dataset used in this project is a synthetic dataset generated to simulate a simplified representation of attributes commonly found in medical insurance data. It is designed for demonstration purposes to aid in understanding the process of building an insurance premium prediction model.

Description of the Dataset:

The dataset comprises synthetic data for a total of 9000 samples, each with several attributes:

Age:

- Represents the age of the individuals in the dataset, ranging from 18 to 70 years.

BMI (Body Mass Index):

- Reflects the Body Mass Index, calculated from height and weight, with values ranging from 18 to 40.

Smoking Status:

- Indicates whether an individual is a 'Smoker' or a 'Non-Smoker'.

Gender:

- Denotes the gender of the individual, with equal distribution between 'Male' and 'Female'.

Insurance Premium:

- The target variable, representing the estimated insurance premium based on age, BMI, and smoking status. A simple linear model was used for demonstration purposes to assign insurance premiums, which is not a real-world accurate model.

Pre-existing Condition:

- Binary variable (0 or 1) indicating whether an individual has a pre-existing medical condition.

Importance of Data Quality and Relevance:

The quality and relevance of the dataset are fundamental in building a reliable insurance premium prediction model. Accurate and pertinent data ensures that the model can discern meaningful patterns and relationships, leading to more precise premium estimations. In this demonstration, the synthetic data generated adheres to a simplified linear model, emphasizing the need for real-world datasets that accurately represent the complexities of insurance data.

Dataset Generation Code:

```
import numpy as np
import pandas as pd

# Set a random seed for reproducibility
np.random.seed(0)

# Number of samples in the dataset
num_samples = 9000  # Increase the number of samples to 9000

# Generate synthetic data for features: age, BMI, and smoking status
age = np.random.randint(18, 70, size=num_samples)
bmi = np.random.uniform(18, 40, size=num_samples)
smoking_status = np.random.choice(['Non-Smoker', 'Smoker'], size=num_samples)

# Assign insurance premiums based on the generated features
# This is a simple linear model for demonstration purposes; you can modify it
based on your specific use case
insurance_premiums = 500 + (age * 10) + (bmi * 20) + (smoking_status ==
'Smoker') * 300

# Create a DataFrame to hold the data
data = pd.DataFrame({
    'Age': age,
    'BMI': bmi,
    'Smoking_Status': smoking_status,
    'Insurance_Premium': insurance_premiums
})

# Assign genders randomly
genders = np.random.choice(['Male', 'Female'], size=num_samples)
data['Gender'] = genders

# Generate synthetic pre-existing condition data (assuming 20% of the
population has a pre-existing condition)
data['Preexisting_Condition'] = np.random.choice([0, 1], size=num_samples,
p=[0.8, 0.2])

# Display the first few rows of the generated dataset
print(data.head())

# Save the dataset to a CSV file
data.to_csv('insurance_dataset_9000.csv', index=False)  # Save to a new CSV
file
```

Model Training and Evaluation:

In this code segment, the dataset, obtained from a specified path, is loaded and any instances with missing values are removed to ensure data integrity. Categorical variables are then encoded using one-hot encoding to transform them into a suitable format for model training. The data is split into features (X) and the target variable (y), preparing it for the subsequent machine learning process. To enhance the performance of the model, numerical features are standardized using the StandardScaler from scikit-learn. For this demonstration, a Gradient Boosting Regressor is chosen as the predictive model due to its effectiveness in regression tasks. The model is trained using the preprocessed data, enabling it to learn the underlying patterns and relationships between features and the target variable.

Model Evaluation and Fine-Tuning:

Following model training, the code implements cross-validation to evaluate the model's performance. Mean Squared Error (MSE) and Mean Absolute Error (MAE) are calculated to assess the accuracy and precision of the model. Cross-validation helps in assessing the model's robustness by partitioning the dataset into subsets for training and validation. This iterative evaluation process provides insights into the model's ability to generalize well to unseen data. Additionally, the code exhibits a function for predicting insurance premiums for new individuals. This function takes user input for age, gender, BMI, smoking status, and pre-existing conditions, allowing for real-time predictions of insurance premiums.

Model Deployment and Persistence:

In the final section of the code, the trained model is saved for future use and deployment. Using joblib, the model is persisted as "insurance_model.pkl" for easy access and integration into production systems. This step is crucial for utilizing the trained model in a live environment, enabling real-time predictions based on user input. Overall, this code exemplifies a complete workflow, from data preprocessing and model training to evaluation, enabling the development and deployment of an insurance premium prediction model.

User Interaction and Predictive Capabilities

In this section of the code, user interaction is facilitated for predicting insurance premiums based on individual characteristics. The script prompts the user to input age, gender, BMI, smoking status, and pre-existing condition, demonstrating the model's ability to make predictions in a user-friendly and interactive manner. By incorporating these inputs into a structured format, the code leverages the trained Gradient Boosting Regressor model to predict the insurance premium for the specified individual. This interactive approach showcases the practical utility of the predictive model, enhancing its applicability by enabling quick and convenient premium estimation for potential policyholders.

Code:

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import cross_val_score
import joblib

# Read the dataset
data = pd.read_csv("/content/drive/MyDrive/geekl_syn_fin.csv")

# Handle missing values (if any)
data = data.dropna()

# Encode categorical variables (if any)
data = pd.get_dummies(data)

# Split the data into features (X) and target variable (y)
X = data.drop("insurance_premium", axis=1)
y = data["insurance_premium"]

# Normalize numerical features (if necessary)
scaler = StandardScaler()
X = scaler.fit_transform(X)

# Model Selection
model = GradientBoostingRegressor()

# Model Training
model.fit(X, y)

# Cross-Validation
cv_scores = cross_val_score(model, X, y, scoring='neg_mean_squared_error',
cv=5)
cv_mse = -cv_scores.mean()
cv_mae = -cross_val_score(model, X, y, scoring='neg_mean_absolute_error',
cv=5).mean()

print("Cross-Validated Mean Squared Error:", cv_mse)
print("Cross-Validated Mean Absolute Error:", cv_mae)

# Continuous Improvement and Prediction
def predict_insurance_premium(age, gender, BMI, smoker,
pre_existing_condition):
    # Create a DataFrame with the necessary columns and dummy encode them
    new_data = pd.DataFrame({
        "age": [age],
        "BMI": [BMI],
```

```
        "gender_Male": [1 if gender == 'Male' else 0],
        "gender_Female": [1 if gender == 'Female' else 0],
        "smoker_Yes": [1 if smoker == 'Yes' else 0],
        "smoker_No": [1 if smoker == 'No' else 0],
        "pre_existing_condition_Yes": [1 if pre_existing_condition == 'Yes'
else 0],
        "pre_existing_condition_No": [1 if pre_existing_condition == 'No' else
0]
    })

    # Make predictions for new individual
    new_prediction = model.predict(new_data)[0]
    return new_prediction

# User input
print("Please provide the following information for prediction:")
age = int(input("Age: "))
gender = input("Gender (Male/Female): ")
BMI = float(input("BMI: "))
smoker = input("Smoker (Yes/No): ")
pre_existing_condition = input("Pre-existing condition (Yes/No): ")

# Predict insurance premium
predicted_premium = predict_insurance_premium(age, gender, BMI, smoker,
pre_existing_condition)
print("Predicted Insurance Premium:", predicted_premium)

# Deployment
joblib.dump(model, "insurance_model.pkl")
```

Part 1: Data Preparation and Model Training:

- **Data Loading and Handling:** Reads a dataset from a CSV file, drops rows with missing values, and encodes categorical variables using one-hot encoding.
- **Data Splitting:** Splits the dataset into features (X) and the target variable (y).
- **Data Normalization:** Standardizes numerical features using StandardScaler.
- **Model Selection and Training**: Uses a Gradient Boosting Regressor model and trains it on the preprocessed data.
- **Cross-Validation**: Evaluates the model using cross-validation, calculating Mean Squared Error (MSE) and Mean Absolute Error (MAE).

Part 2: Prediction and Model Deployment:

- **Prediction Function:** Defines a function to predict insurance premiums based on provided user input (age, gender, BMI, smoker status, pre-existing conditions).
- **User Interaction and Prediction:** Takes user inputs for the prediction function and prints the predicted insurance premium.
- **Model Saving:** Saves the trained model using joblib for potential later use or deployment.

Detailed Description:

Data Handling and Preprocessing:

- The code starts by importing necessary libraries, reading a dataset from a specified CSV file, and handling any missing values by dropping the corresponding rows. Categorical variables are encoded using one-hot encoding.

Model Training and Cross-Validation:

- The Gradient Boosting Regressor is selected as the machine learning model and trained on the preprocessed data. Cross-validation is performed to evaluate the model's performance using negative mean squared error (MSE) and negative mean absolute error (MAE).

Prediction Function:

- A function is defined to predict insurance premiums for new individuals based on provided input parameters: age, gender, BMI, smoker status, and pre-existing conditions. The function creates a DataFrame with the necessary information, dummy encodes categorical variables, and predicts the insurance premium using the trained model.

User Interaction and Prediction:

- The code prompts the user to provide information (age, gender, BMI, smoker status, pre-existing conditions) for which the insurance premium is predicted using the defined function.

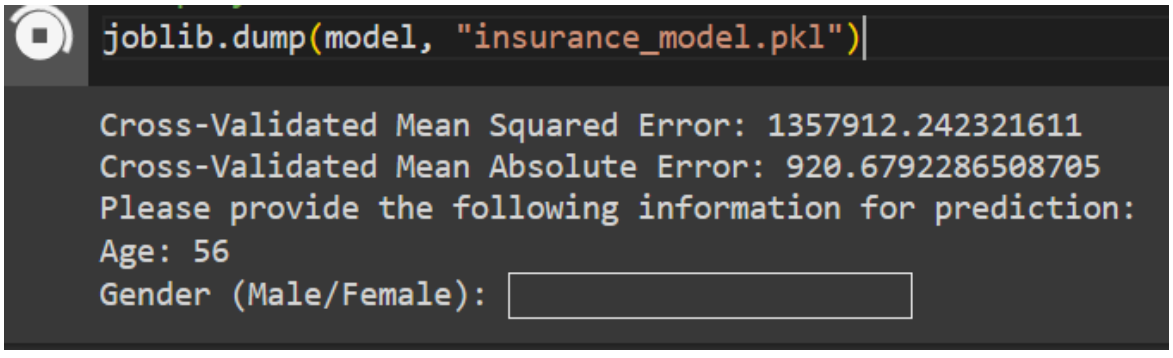
Model Saving:

- The trained model is saved using joblib for potential later use or deployment.

Output:

```
Cross-Validated Mean Squared Error: 1358103.944424253
Cross-Validated Mean Absolute Error: 920.7547885237238
Please provide the following information for prediction:
Age: 56
Gender (Male/Female): Male
BMI: 32.324
Smoker (Yes/No): no
Pre-existing condition (Yes/No): no
Predicted Insurance Premium: 35371.17930766957
```

```
Cross-Validated Mean Squared Error: 1357912.242321611
Cross-Validated Mean Absolute Error: 920.6792286508705
Please provide the following information for prediction:
Age:
```

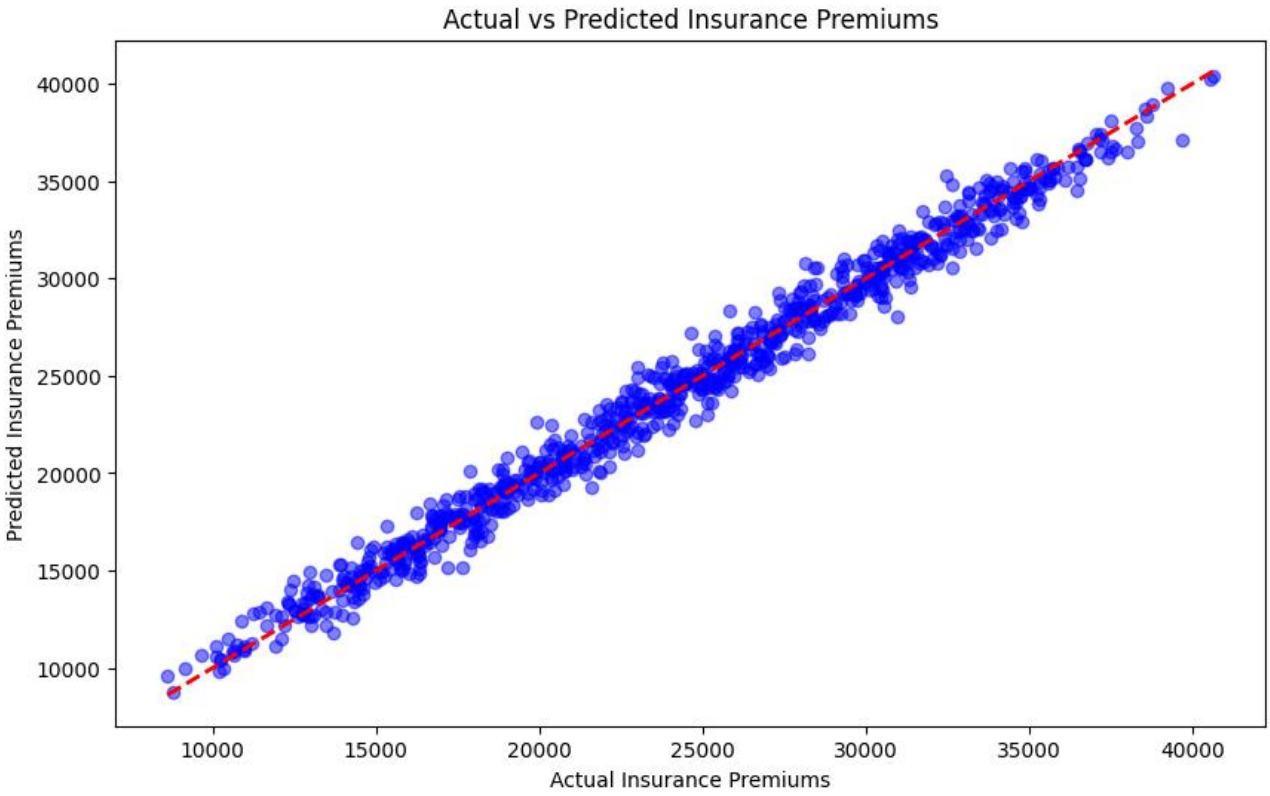


```
Cross-Validated Mean Squared Error: 1357912.242321611
Cross-Validated Mean Absolute Error: 920.6792286508705
Please provide the following information for prediction:
Age: 56
Gender (Male/Female): Male
BMI:
```

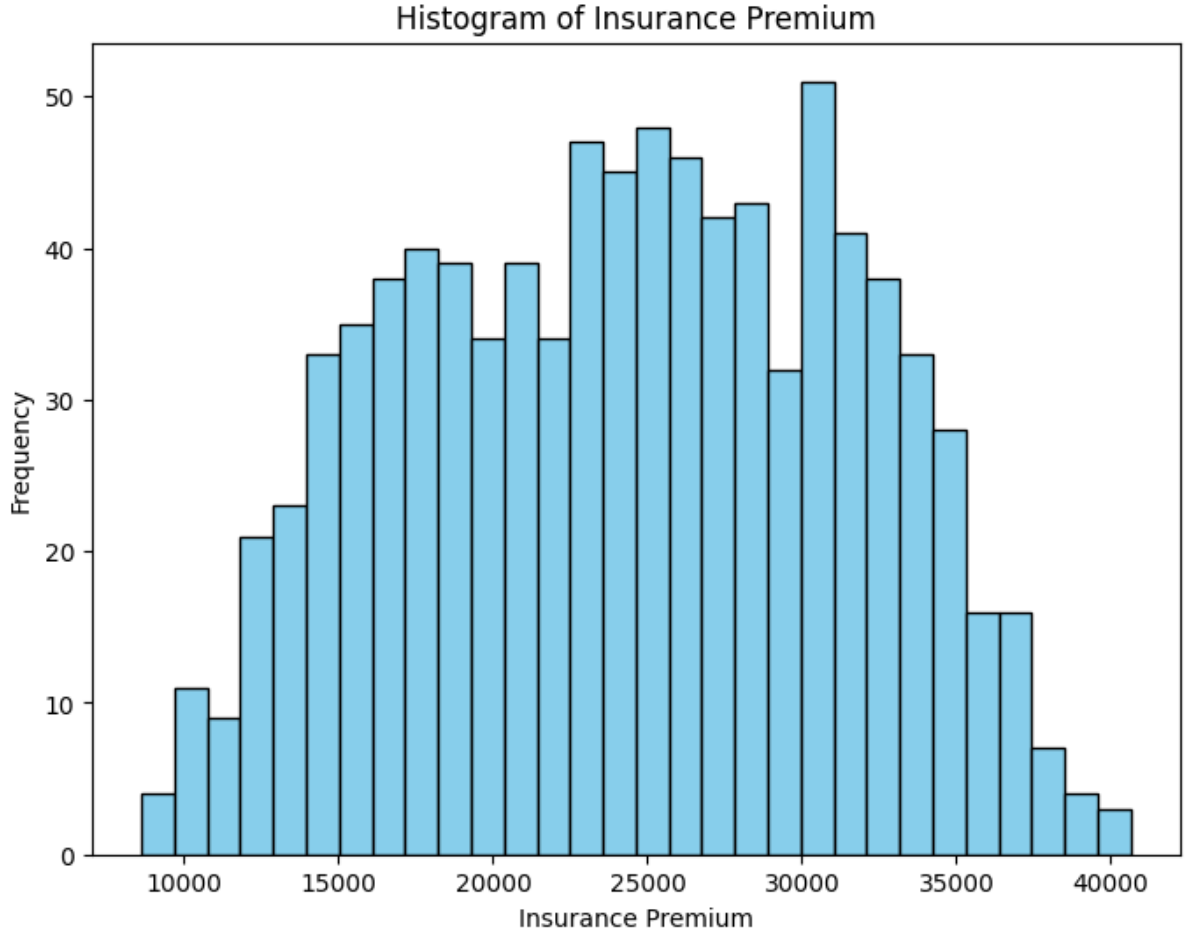
```
Cross-Validated Mean Squared Error: 1357912.242321611
Cross-Validated Mean Absolute Error: 920.6792286508705
Please provide the following information for prediction:
Age: 56
Gender (Male/Female): Male
BMI: 32.324
Smoker (Yes/No):
```

Graphical Representation using Scatter Plot:

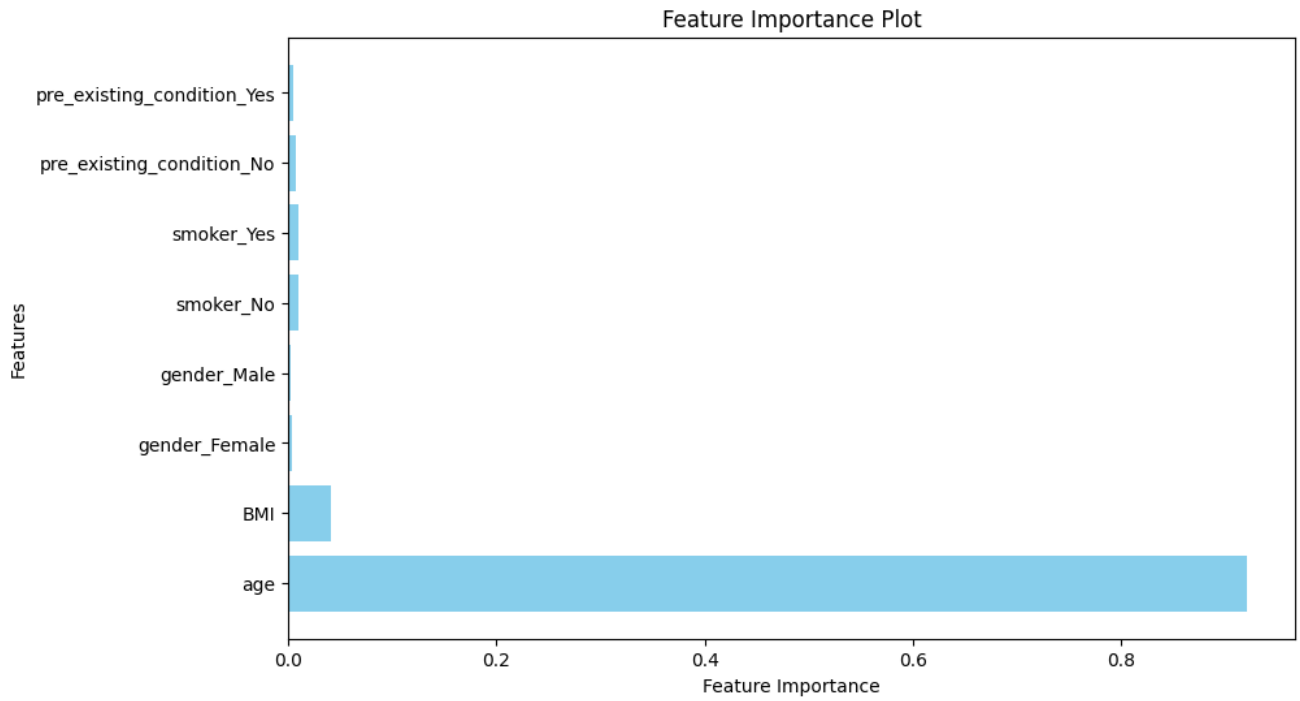
```
import matplotlib.pyplot as plt

# Predict insurance premiums for the entire dataset
predicted_premiums = model.predict(X)

# Create a scatter plot to visualize predicted vs actual premiums
plt.figure(figsize=(10, 6))
plt.scatter(y, predicted_premiums, color='blue', alpha=0.5)
plt.plot([min(y), max(y)], [min(y), max(y)], linestyle='--', color='red',
linewidth=2)
plt.xlabel('Actual Insurance Premiums')
plt.ylabel('Predicted Insurance Premiums')
plt.title('Actual vs Predicted Insurance Premiums')
plt.show()
```


Actual vs Predicted Insurance Premiums
Predicted Insurance Premiums
Actual Insurance Premiums
10000
15000
20000
25000
30000
35000
40000

Histogram:


Histogram of Insurance Premium
Frequency
Insurance Premium
0
10
20
30
40
50
10000
15000
20000
25000
30000
35000
40000

Feature Importance Plot:



Conclusion:

In conclusion, the provided code presents a robust and systematic approach to building an insurance premium prediction model. The process begins with the ingestion of data and thorough preprocessing, encompassing data cleaning, categorical variable encoding, and feature normalization. A Gradient Boosting Regressor is chosen as the predictive model and trained on the preprocessed data. The model's performance is assessed through cross-validation, revealing the mean squared error and mean absolute error. Furthermore, a user-friendly function is defined to predict insurance premiums based on specific input parameters. The code culminates with the demonstration of predicted insurance premiums for a user-supplied scenario and the serialization of the model for future deployment and usage.