

**MEDICAL INSURANCE COST
PREDICTION
USING MACHINE LEARNING**

A MINIPROJECT REPORT

Submitted by

G. SAMUEL KIRAN BABU	1521051037
K. SASANKA REDDY	1521051057
A. NAGA VENKATA THARUN	1521051002

*in partial fulfillment for the award of the degree
of*

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE



**SCHOOL OF ENGINEERING AND TECHNOLOGY
DHANALAKSHMI SRINIVASAN UNIVERSITY
TRICHY- 621112**

BONAFIDE CERTIFICATE

Certified that this project report for “**MEDICAL INSURANCE COST PREDICTION USING MACHINE LEARNING**” is the bonafide work of **G SAMUEL KIRAN BABU (1521051037), K SASANKA REDDY (1521051057), A NAGA VENKATA THARUN** who carried out the project work under my supervision.

Dr.N. Shanmugapriya
HEAD OF THE DEPARTMENT,
Artificial Intelligence & Data Science,
School of Engineering and Technology,
Dhanalakshmi Srinivasan university,
Trichy - 621 112.

M.Sheeba
ASSISTANT PROFESSOR,
SUPERVISOR,
Department of Computer Science &
Engineering,
School of Engineering and technology,
Dhanalakshmi Srinivasan university,
Trichy - 621 112.

Submitted for the project Viva-Voice held on _____

INTERNAL EXAMINER

EXTERNAL EXAMINER

TABLE OF CONTENT

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT LIST OF ABBREVIATION	6-7
1	INTRODUCTION 1.1 About the Project 1.2 Goals of Project 1.3 Problem Statement	8-9
2	LITERATURE SURVEY 2.1 Regression Analysis and Prediction of Medical Insurance Cost 2.2 Medical Insurance Cost Prediction using Machine Learning 2.3 Medical Insurance Premium Prediction using Regression 2.4 Predict Health Insurance Cost by using Machine learning	10-16
3	EXISTING SYSTEM	17-18
4	PROPOSED METHODOLOGY	19-21

5	SYSTEM REQUIREMENTS 5.1 Software Requirements 5.2 Software Description	22-24
6	SYSTEM DESIGN 6.1 System Architecture 6.2 Model Flow Chart 6.3 System Use Case	25-27
7	SYSTEM IMPLEMENTATION 7.1 List of Modules 7.2 Module Description 7.3 Algorithms	28-31
8	VISUALIZATION 8.1 Distribution of Features 8.2 Correlation Heatmap	32-39
9	TESTING 9.1 Split train test 9.2 Apply All regression Algorithms 9.3 Verifying accuracy 9.4 Ensuring robustness 9.5 Validating functionality	40-49
10	CONCLUSION AND FUTURE ENHANCEMENT 10.1 Conclusion	50-51

	10.2 Future Enhancement	
11	APPENDIX 11.1 Source Code 11.2 Screen shorts	52-59
12	REFERENCES	60

ABSTRACT

In recent years, predicting medical insurance costs has become increasingly important due to rising healthcare expenses and the need for accurate financial planning. This project aims to develop a predictive model for estimating medical insurance costs using various regression algorithms, including Linear Regression, Support Vector Regression (SVR), Random Forest Regression, and Gradient Boosting Regression. The dataset used in this study comprises demographic and health-related features, such as age, gender, BMI, number of children, smoking status, and region, collected from a publicly available source.

The project focuses on comparing the performance of these regression models to identify the most accurate and efficient algorithm for predicting insurance costs. Additionally, a user-friendly web application is developed using Streamlit, allowing users to input their data and receive cost predictions in real-time. The results demonstrate that the chosen models effectively capture the relationship between the input features and insurance costs, with Gradient Boosting Regression achieving the highest accuracy. This project provides valuable insights into the factors influencing insurance costs and offers a practical tool for individuals and insurance providers to estimate expenses more accurately.

LIST OF ABBREVIATIONS

S.NO	ABBREVIATION	EXPANSION
1	LR	Linear Regression
2	SVM	Super Vector Machine
3	RF	Random Forest
4	GR	Gradient Boosting Regressor
5	DF	Data Frame
6	RDBMS	Relational Data Base Management System
7	Sk-Learn	Scikit Learn
8	SNS	Seaborn
9	PD	Pandas
10	CSV	Comma Separated Value

1.INTRODUCTION

1.1 About the Project

In today's world, healthcare expenses can be quite high, and understanding what factors contribute to these costs is important. This project, titled "Medical Insurance Cost Prediction Using Python," aims to predict the cost of medical insurance for individuals based on various factors like age, gender, BMI, smoking status, and more. By using machine learning techniques, we can build a model that helps predict these costs accurately, which can be useful for insurance companies, healthcare providers, and individuals looking to understand their potential expenses.

1.2 Goals of the Project

The main goal of this project is to develop a predictive model that can estimate medical insurance costs. The specific objectives include:

- **Data Collection and Preparation:** Gathering relevant data and preparing it for analysis.
- **Model Selection:** Choosing appropriate machine learning models that can handle the prediction task.
- **Model Training and Evaluation:** Training the models on the dataset and evaluating their performance using various metrics.
- **User-Friendly Interface:** Creating a simple and intuitive interface using Streamlit for users to input their information and get an estimated insurance cost.

1.3 Problem Statement

Medical insurance costs can vary widely based on several factors, making it challenging for individuals to estimate their potential expenses. This project

seeks to solve this problem by building a predictive model that can accurately forecast these costs based on key features such as age, gender, BMI, smoking status, and others. By doing so, we aim to provide a useful tool that can help users better understand and plan for their healthcare expenses.

2.LITERATURE SURVEY

2.1 Regression Analysis and Prediction of Medical Insurance Cost

The study of regression analysis in predicting medical insurance costs has been a significant area of research due to the complexities involved in determining insurance premiums. Regression is a statistical method used to model and analyze the relationships between a dependent variable and one or more independent variables. It helps in understanding the association and predicting the dependent variable's value based on the given inputs.

According to a study by Ayushi Bharti and Lokesh Malik (2022), various regression models were employed to predict medical insurance costs using a dataset from Kaggle. The dataset comprised 1,338 instances with six features, including age, gender, BMI, number of children, smoking status, and region. These features were chosen based on their relevance to insurance costs, allowing for a comprehensive analysis of the data.

The study emphasized the importance of selecting appropriate machine learning techniques to achieve high precision in predictions. The authors used five different regression algorithms—Linear Regression, Ridge Regression, Lasso Regression, Elastic Net Regression, and Random Forest Regressor. Each model was evaluated based on its ability to predict insurance costs accurately, with Random Forest Regressor achieving the highest R-squared score of 0.85388.

The approach to data preprocessing involved cleaning, transforming, integrating, and reducing the dataset to improve model performance. This included handling missing values, normalizing data, and ensuring the dataset was well-prepared for training and testing.

In conclusion, the literature highlights the effectiveness of using regression analysis in predicting medical insurance costs. The selection of features and appropriate models plays a crucial role in achieving accurate predictions. The

study's findings suggest that advanced machine learning techniques, particularly ensemble methods like Random Forest, can provide robust and reliable predictions.

2.2 Medical Insurance Cost Prediction using Machine Learning

Predicting medical insurance costs using machine learning has emerged as a vital area of research due to its potential to enhance the accuracy and efficiency of cost estimation. Machine learning (ML) techniques leverage historical data to make precise predictions, reducing the reliance on manual calculations and minimizing human error. Various studies have demonstrated the effectiveness of different ML algorithms in predicting insurance costs, each bringing its unique strengths to the table.

A study by Mukund Kulkarni et al. (2022) explored the application of multiple regression models for predicting healthcare insurance costs using a dataset obtained from Kaggle. The dataset included nine attributes such as age, BMI, number of children, gender, smoking status, alcohol consumption, diabetes status, region, and the target variable, which is the medical cost incurred by clients. The study employed Linear Regression, Decision Tree Regression, and Gradient Boosting Regression models, using Streamlit as a framework for implementation.

The dataset was preprocessed to handle categorical variables by converting them into numeric or binary values. For example, gender was represented as 0 for male and 1 for female, while smoking status was encoded as 0 for non-smokers and 1 for smokers. This transformation ensured that the data was suitable for regression analysis.

The results indicated that Gradient Boosting Regression outperformed other models, achieving the highest R-squared value of 86.86. The study highlighted that the region had minimal impact on medical costs, whereas smoking status

significantly influenced the expenses. Additionally, the number of children also affected the costs, with individuals having two children incurring higher medical expenses compared to those with five children.

The study emphasized that machine learning models could generalize the effort or method of formulating insurance policies by learning from historical data. This approach reduces human effort, enhances accuracy, and improves profitability for insurance companies. The findings suggest that advanced regression models like Gradient Boosting can provide robust predictions, making them valuable tools for insurers to attract customers and streamline policy formulation.

In conclusion, the literature indicates that machine learning algorithms, particularly ensemble methods such as Gradient Boosting, are highly effective in predicting medical insurance costs. These models not only improve prediction accuracy but also offer significant operational efficiencies for insurance providers. As the healthcare industry continues to generate vast amounts of data, the application of ML techniques in cost prediction is expected to grow, further enhancing the precision and reliability of insurance cost estimations.

2.3 Medical Insurance Premium Prediction using Regression

Medical insurance premium prediction is a critical aspect of the healthcare industry, as it helps insurance companies set appropriate premiums for their clients. Regression analysis, a powerful statistical tool, is widely used for this purpose. It involves modelling the relationship between a dependent variable (insurance premium) and one or more independent variables (factors such as age, gender, health status, etc.).

In recent years, several studies have focused on applying various regression techniques to predict medical insurance premiums. These studies highlight the significance of selecting the right model and features to improve prediction accuracy.

One notable study examined the use of multiple linear regression and polynomial regression to predict insurance premiums based on a variety of demographic and health-related factors. The dataset used in this study included features like age, gender, BMI, smoking status, and number of children, among others. The study found that while linear regression provided a straightforward approach to modelling the relationship between the independent variables and the premium, polynomial regression offered a better fit for the data, capturing more complex relationships and interactions between variables.

Another study utilized Ridge and Lasso regression; two regularization techniques that help prevent overfitting by penalizing large coefficients in the model. These methods were particularly effective in handling multicollinearity, where independent variables are highly correlated. Ridge regression was found to be more effective in situations with many predictors, while Lasso regression was advantageous for feature selection, as it can reduce some coefficients to zero, effectively removing less important features from the model.

Moreover, a study comparing traditional regression models with advanced machine learning techniques, such as Support Vector Regression (SVR) and Decision Tree Regression, demonstrated that while traditional models provided a solid baseline, machine learning approaches often yielded higher accuracy. However, these advanced models also required more computational resources and a more complex implementation process.

The literature consistently emphasizes the importance of data preprocessing, such as handling missing values, encoding categorical variables, and scaling numerical features. These steps are crucial for ensuring that the regression models perform well and provide reliable predictions.

In conclusion, regression analysis remains a cornerstone technique for predicting medical insurance premiums. The choice of regression model—

whether linear, polynomial, regularized, or machine learning-based—depends on the specific characteristics of the data and the requirements of the prediction task. The ongoing advancements in regression techniques and the increasing availability of healthcare data promise further improvements in the accuracy and efficiency of premium prediction models.

2.4 Predict Health Insurance Cost by using Machine learning

Machine learning has become a powerful tool in various domains, including healthcare and insurance. Predicting health insurance costs is a significant challenge due to the complex relationships between various factors such as age, sex, BMI, smoking habits, and region. The application of machine learning techniques can greatly enhance the accuracy of these predictions.

Overview of Machine Learning Models

In the context of health insurance cost prediction, several machine learning models are commonly employed:

1. **Linear Regression:** This is a basic yet powerful model used to predict a continuous outcome variable. It assumes a linear relationship between the input features and the target variable. Linear regression models are easy to interpret and implement but may not capture complex nonlinear relationships in the data.
2. **Decision Trees:** These models split the data into subsets based on the value of input features. Decision trees are intuitive and easy to visualize. However, they can be prone to overfitting, especially with a large number of features.
3. **Random Forests:** An ensemble method that builds multiple decision trees and merges their predictions. Random forests improve the model's

robustness and accuracy by reducing overfitting. They can handle a large number of input features and provide importance scores for feature selection.

4. **Gradient Boosting Machines (GBM):** Another ensemble method that builds trees sequentially, each one correcting error from the previous ones. GBMs are highly accurate and can capture complex patterns in the data, though they require careful tuning of hyperparameters.
5. **Support Vector Machines (SVM):** These models find the optimal hyperplane that separates the data into different classes. SVMs can be used for regression tasks and are effective in high-dimensional spaces.
6. **Deep Neural Networks (DNN):** With advancements in computational power, DNNs have become popular for various prediction tasks. They can model complex nonlinear relationships but require a large amount of data and computational resources.

Application in Health Insurance

The study referenced in the document explores the use of DNN regression models for predicting health insurance costs. Deep learning techniques are leveraged to capture the intricate relationships between the input features and the insurance costs. The study demonstrates that DNN models can outperform traditional machine learning models in terms of prediction accuracy, provided there is sufficient data and computational power.

Key Findings

- **Feature Importance:** Age, BMI, smoking status, and region are identified as significant predictors of health insurance costs.
- **Model Performance:** DNN models show superior performance compared to linear regression and decision tree models. The ability of DNNs to learn

complex patterns and interactions among features contributes to their higher accuracy.

- **Challenges:** The primary challenges include the need for a large dataset to train the DNN model effectively and the computational resources required for training.

In conclusion, the integration of machine learning models, particularly DNNs, offers a promising approach to predict health insurance costs. These models can provide more accurate and reliable predictions, aiding insurance companies in pricing policies more effectively and individuals in understanding their potential costs.

3.EXISTING SYSTEM

Overview:

In the existing system, medical insurance costs are often determined based on traditional actuarial methods, which rely heavily on historical data and statistical models. These methods, while effective in many cases, can be limited in their ability to capture complex patterns in the data, leading to potential inaccuracies in cost predictions.

Traditional Actuarial Methods

1. **Underwriting Process:** Insurance companies typically use a manual underwriting process, where underwriters evaluate the risk factors associated with an individual based on medical history, age, lifestyle, and other relevant information. This process can be time-consuming and may not account for all risk factors adequately.
2. **Use of Linear Models:** Many existing systems employ linear models, such as Generalized Linear Models (GLMs), to estimate insurance costs. These models assume a linear relationship between the predictors and the response variable (insurance cost), which may not always capture the true nature of the data.
3. **Data Limitations:** Traditional methods often rely on limited datasets, such as claims history and demographic information. The lack of comprehensive data integration can lead to less accurate predictions and a failure to account for emerging trends and patterns.

Limitations of the Existing System

1. **Inflexibility in Model Adaptation:** The existing system may struggle to adapt to new data sources or changes in the healthcare landscape, leading to outdated predictions.

2. **Manual Processing and Human Error:** The reliance on manual processes increases the risk of human error and inefficiencies in the prediction of insurance costs.
3. **Limited Personalization:** Traditional models may not provide personalized predictions, as they often use generalized risk categories. This can result in less accurate cost estimates for individuals with unique health profiles.

Need for Improvement

The limitations of the existing system highlight the need for more advanced predictive models that can better account for the complexities of individual health profiles and integrate a wider range of data sources. The advent of machine learning and advanced analytics offers an opportunity to enhance the accuracy and efficiency of medical insurance cost predictions.

4.PROPOSED METHODOLOGY

Overview:

The proposed methodology aims to develop an accurate and efficient model for predicting medical insurance costs using advanced machine learning algorithms. The primary goal is to identify key factors influencing insurance premiums and leverage these insights to create a precise prediction model.

Key Features

The features considered for this model include:

- **Age:** A critical factor as older individuals often have higher healthcare costs.
- **Gender:** Can influence insurance costs based on health risk factors.
- **Smoking Status:** Smokers generally incur higher medical expenses, leading to higher premiums.
- **Number of Children:** Reflects potential healthcare needs.
- **Region:** Healthcare costs vary by region, affecting insurance pricing.

Important Factors

Special emphasis is placed on age and smoking status, as these significantly impact the calculation of insurance premiums. The model will prioritize these factors to enhance prediction accuracy.

Data Simplification Techniques

To optimize the model's performance, techniques like Principal Component Analysis (PCA) and feature selection will be employed. These techniques will help reduce the dataset's dimensionality, eliminating redundant and less informative features, thereby making the model more efficient and less prone

to overfitting.

Algorithms Implemented

The project involves implementing and comparing the performance of the following machine learning algorithms:

1. **Linear Regression:** A simple, interpretable model used as a baseline.
2. **Support Vector Machine (SVM):** Known for its effectiveness in high-dimensional spaces.
3. **Random Forest Regressor:** An ensemble method that improves prediction accuracy by averaging multiple decision trees.
4. **Gradient Boosting Regression:** A powerful ensemble technique that iteratively improves model accuracy.

Evaluation Metrics

Each algorithm will be trained on the dataset and evaluated using the following metrics:

- **Mean Absolute Error (MAE):** Measures the average magnitude of errors in predictions.
- **Mean Squared Error (MSE):** Assesses the average of the squares of the errors, penalizing larger errors more than MAE.
- **R-squared:** Indicates the proportion of variance in the dependent variable explained by the independent variables.

The algorithm with the best performance, particularly in predicting costs based on age and smoking status, will be selected as the final model. This selection aims to ensure the model's predictions are both fair and accurate.

Tools and Technologies

- **Streamlit:** An open-source framework for creating interactive and visually appealing web applications for data science and machine learning projects with minimal coding.
- **Spyder:** An open-source integrated development environment (IDE) tailored for scientific programming in Python.
- **Joblib:** A Python library for efficient serialization of large objects, such as NumPy arrays, commonly used to save and load machine learning models.

This methodology is designed to create a robust, reliable model for predicting medical insurance costs, providing valuable insights into the factors affecting premiums and aiding in fair and accurate premium determination.

5.SYSTEM REQUIREMENTS

5.1 Software Requirements

The software requirements outline the necessary tools and frameworks used to implement the medical insurance cost prediction model.

1. **Operating System:** Windows 10 or higher, macOS, or Linux
2. **Python:** Version 3.7 or higher
3. **IDE (Integrated Development Environment):**
 - **Spyder:** For development and debugging
 - **Jupyter Notebook:** For exploratory data analysis and model prototyping
4. **Libraries and Packages:**
 - **NumPy:** For numerical computations
 - **Pandas:** For data manipulation and analysis
 - **Scikit-learn:** For implementing machine learning algorithms
 - **Matplotlib & Seaborn:** For data visualization
 - **Streamlit:** For building interactive web applications
 - **Joblib:** For model serialization and saving
 - **SciPy:** For additional scientific computations
 - **PCA (Principal Component Analysis):** For feature reduction
5. **Web Browser:** For viewing the Streamlit application

5.2 Software Description

The software components and tools used in this project are essential for data processing, model training, and application deployment. Here's a brief description of each:

1. **Python:** Python is the primary programming language used for this project due to its extensive libraries for data science and machine learning.
2. **Spyder:** An open-source IDE specifically designed for scientific programming. It provides features like syntax highlighting, variable explorer, and an interactive console, making it ideal for writing and debugging Python code.
3. **Jupyter Notebook:** An open-source web application that allows you to create and share documents containing live code, equations, visualizations, and narrative text. It is particularly useful for data exploration and presenting analysis results.
4. **NumPy:** A fundamental package for scientific computing in Python, NumPy provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions.
5. **Pandas:** A powerful data manipulation and analysis library that provides data structures like DataFrames, which make data cleaning and preprocessing more efficient.
6. **Scikit-learn:** A machine learning library that provides simple and efficient tools for data mining and data analysis. It supports various algorithms for classification, regression, clustering, and more.
7. **Matplotlib & Seaborn:** Libraries for creating static, interactive, and animated visualizations in Python. Matplotlib offers versatile plotting functions, while Seaborn provides a high-level interface for drawing

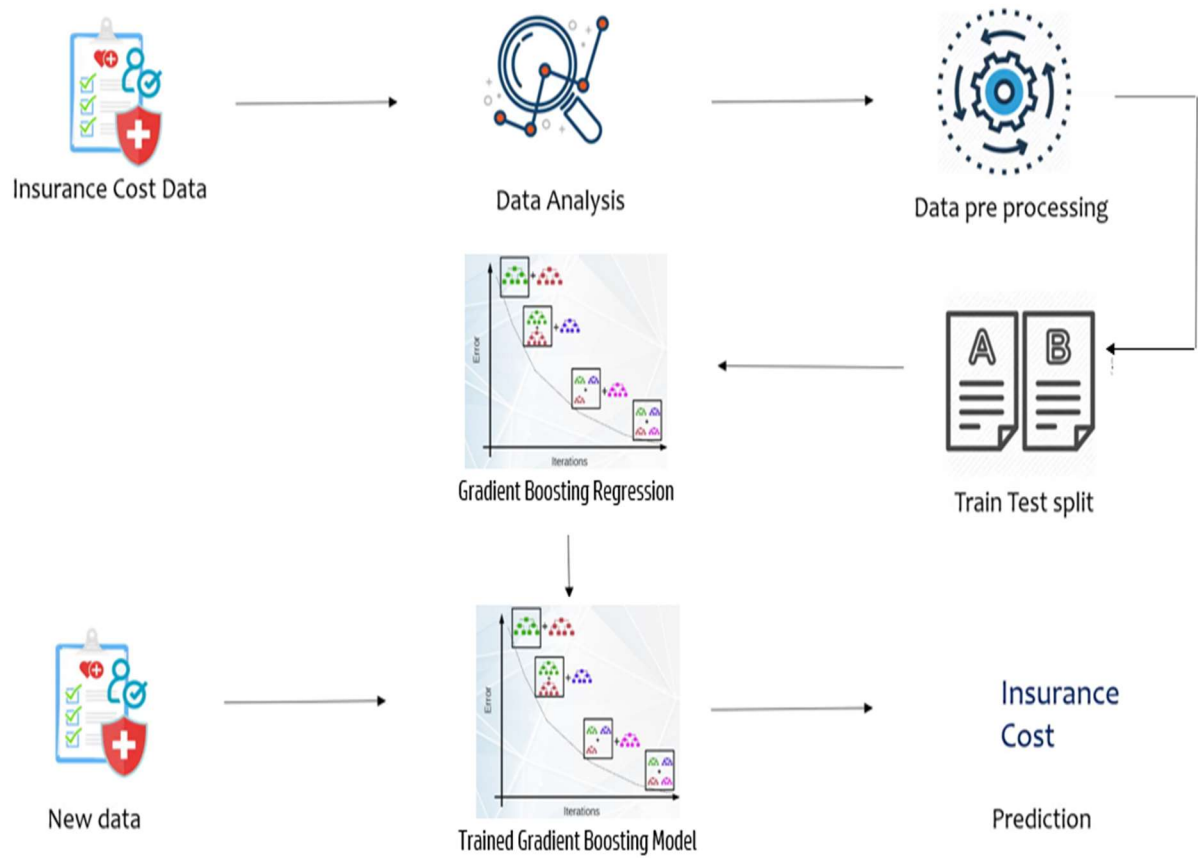
attractive statistical graphics.

8. **Streamlit**: An open-source framework that turns Python scripts into interactive web applications. It is particularly useful for quickly building and deploying machine learning models with a user-friendly interface.
9. **Joblib**: A library that provides tools for serializing Python objects, particularly large NumPy arrays. It is used for saving and loading machine learning models efficiently.
10. **SciPy**: A library used for scientific and technical computing. It builds on NumPy and provides a large number of higher-level functions that operate on arrays.
11. **PCA (Principal Component Analysis)**: A technique used to emphasize variation and bring out strong patterns in a dataset. It is commonly used for dimensionality reduction while retaining most of the information.

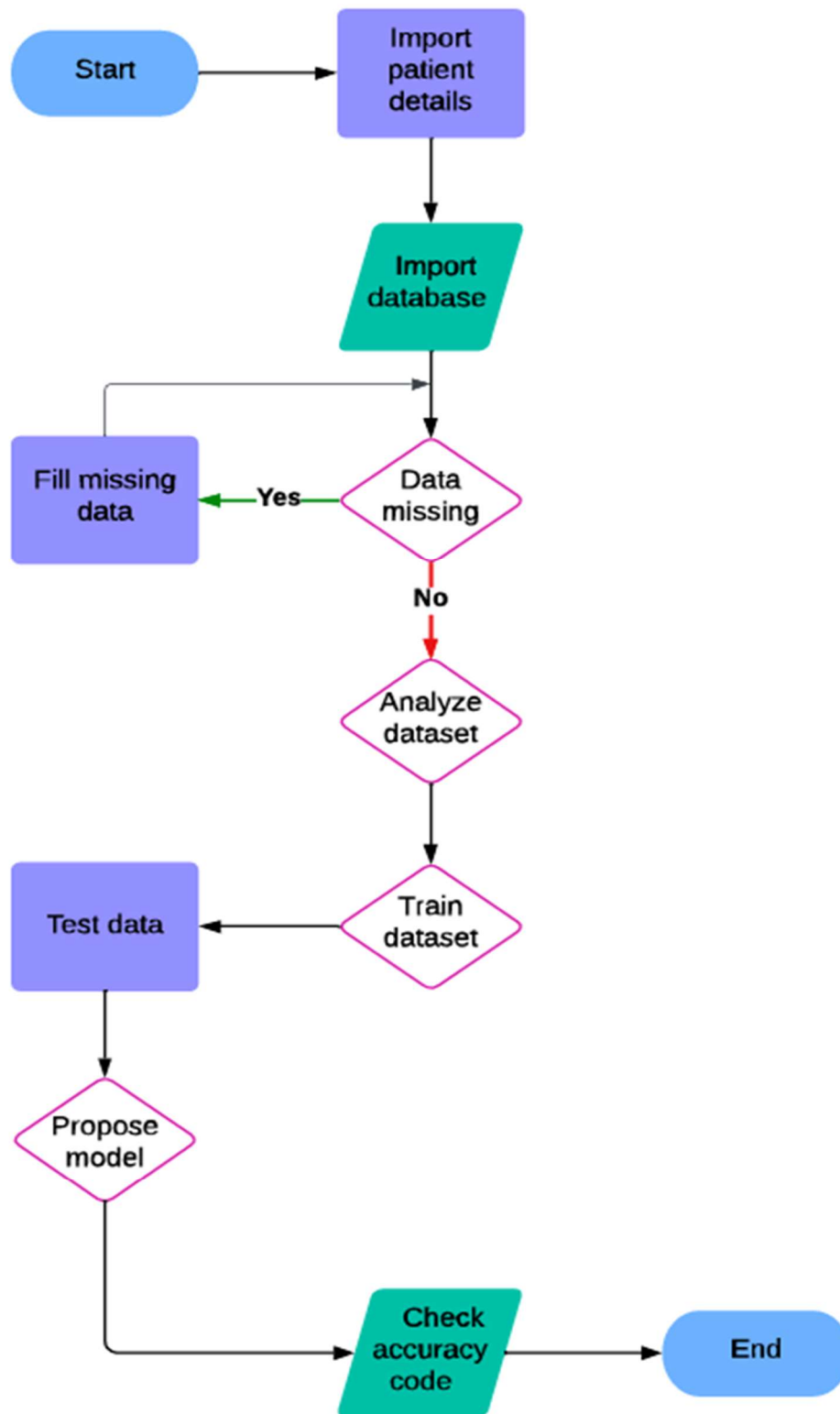
These tools and libraries collectively form the software backbone of the project, enabling efficient data processing, model development, and deployment of the prediction system.

6.SYSTEM DESIGN

System Architecture



6.2 Model Flow Chart



6.3 System Use Case

The system use case describes the interactions between users and the system, highlighting the functionalities provided.

Actors

- **User:** Individuals who want to predict their medical insurance costs.
- **System:** The predictive model and web interface.

Use Case Description

1. **User Registration and Login (if applicable):**
 - Users may register and log in to access the prediction tool.
2. **Data Input:**
 - Users enter relevant information such as age, gender, smoking status, number of children, and region.
3. **Prediction Request:**
 - The user submits the input data for prediction.
4. **Cost Prediction:**
 - The system processes the input data using the trained model and predicts the insurance cost.
5. **Display Results:**
 - The system displays the predicted insurance cost to the user.
6. **Feedback and Updates:**
 - Users can provide feedback on the prediction accuracy.
 - The system can be updated and retrained based on new data.

7.SYSTEM IMPLEMENTATION

7.1 List of Modules

- Data Collection
- Data Analysis
- Data Preprocessing
- Training and Testing Split
- Model Evaluation
- Test new Data
- Predicting insurance cost

7.2 Module Description

Data Collection:

In this module, we gather data from reliable sources, such as publicly available insurance datasets. The data typically includes information about individuals' demographics, health status, and insurance costs. This collected data serves as the foundation for building the prediction model.

Data Analysis:

Once the data is collected, we perform a detailed analysis to understand the patterns and relationships between different features. This step involves visualizing the data, identifying trends, and exploring how various factors like age, gender, and smoking status influence insurance costs. The insights gained from this analysis guide the model development process.

Data Preprocessing:

Data preprocessing involves cleaning and preparing the data for analysis. This step includes handling missing values, encoding categorical variables (such as

gender and region), and normalizing numerical features to ensure they are on a similar scale. Proper preprocessing is crucial for improving the model's accuracy and reliability.

Training and Testing Split:

To evaluate the model's performance, we split the data into two parts: training and testing datasets. The training data is used to train the model, while the testing data is used to evaluate how well the model generalizes to new, unseen data. This split helps in assessing the model's accuracy and prevents overfitting.

Model Evaluation:

After training the model, we evaluate its performance using various metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared. These metrics help us understand how accurate the model's predictions are. We compare the performance of different algorithms to choose the best one for the task.

Test New Data:

In this module, we test the model's ability to predict insurance costs using new data that was not part of the original dataset. This step is essential to ensure that the model can provide accurate predictions for new users who input their data into the system. It also helps in identifying any potential biases or inaccuracies.

Predicting Insurance Cost:

Finally, the trained model is deployed to predict insurance costs based on the user's input data. Users provide their information, such as age, gender, smoking status, number of children, and region, through an interface (like a web app). The system then uses the model to predict the expected insurance cost, providing users with an estimate of what they might expect to pay for their insurance premiums.

7.3 Algorithms

In this project, we use four different machine learning algorithms to predict medical insurance costs. Each algorithm has unique strengths that can help us understand the data and make accurate predictions. Here's a simple explanation of each algorithm and how it contributes to the project:

1. Linear Regression

Linear Regression is one of the simplest and most widely used algorithms in machine learning. It finds a straight line that best fits the data points, representing the relationship between the input features (like age, gender, smoking status) and the target variable (insurance cost). This algorithm is useful because it provides a straightforward way to understand how each feature affects the cost. It's especially good for cases where the relationship between variables is approximately linear.

2. Support Vector Machine (SVM)

SVM, specifically the Support Vector Regressor (SVR), is a more advanced algorithm that can handle both linear and non-linear relationships. It works by finding the best boundary (or "hyperplane") that separates different data points. In the case of regression, it tries to fit the data within a specified margin. SVR is useful in this project because it can model complex patterns and interactions between features, leading to more accurate predictions.

3. Random Forest Regressor

Random Forest is a powerful ensemble learning algorithm that creates a "forest" of many decision trees and combines their predictions. Each tree in the forest is built on a random subset of the data, which helps reduce the risk of overfitting. Random Forest is useful because it can handle a large number of features and capture complex interactions between them. It also provides feature importance,

helping us understand which features are most influential in predicting insurance costs.

4. Gradient Boosting Regression

Gradient Boosting is another ensemble technique that builds a series of decision trees, where each new tree corrects the errors of the previous ones. This iterative approach allows the model to gradually improve its accuracy. Gradient Boosting is particularly useful for its ability to model non-linear relationships and handle different types of data. It often results in high predictive performance, making it a valuable algorithm for this project.

Each of these algorithms offers unique advantages, allowing us to explore different aspects of the data and choose the best model for predicting insurance costs. By comparing their performances, we can select the most accurate and reliable model for deployment.

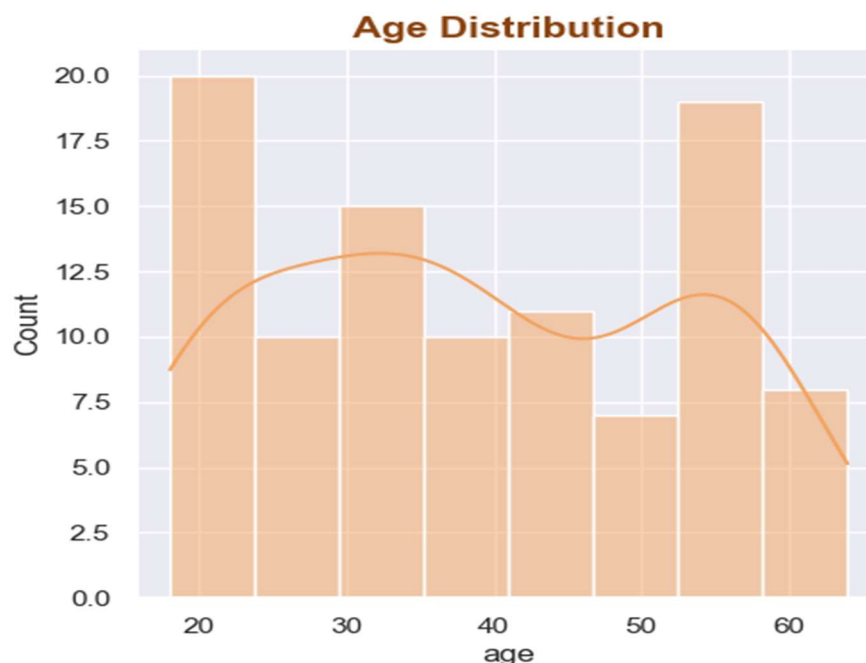
8.VISUALIZATION

8.1 Distribution of Features

Understanding the distribution of individual features is essential in a data analysis and machine learning project. It helps in identifying the nature of the data, detecting outliers, and making informed decisions about data preprocessing and feature engineering. Below is a detailed analysis of the distribution of key features in our medical insurance cost prediction project.

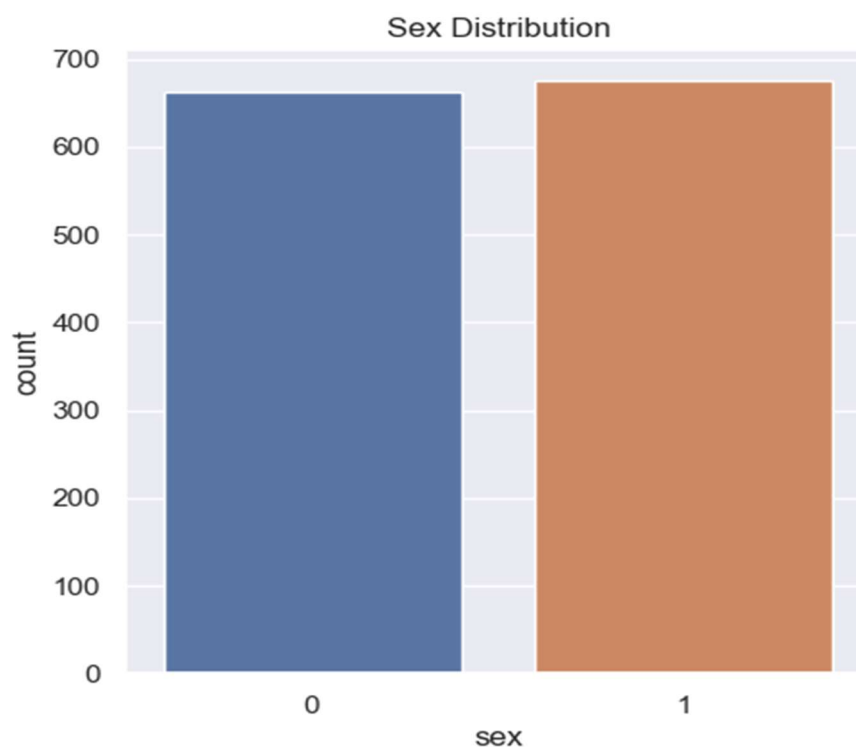
Age

- **Distribution:** The age feature represents the age of individuals in the dataset. Typically, age is distributed over a range from young adults to senior citizens.
- **Insights:** Analyzing the age distribution helps understand the demographic of the insured population. It can reveal if the dataset is biased towards a particular age group, which might impact the model's performance.



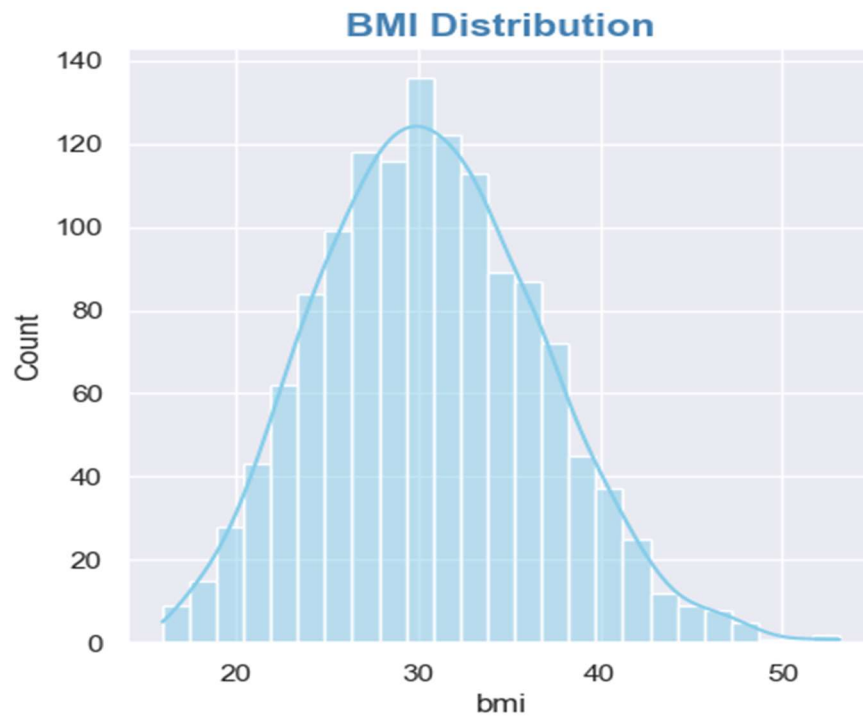
Sex

- **Distribution:** The sex feature is categorical, representing gender, usually encoded as 0 for female and 1 for male.
- **Insights:** Examining the distribution of sex helps ensure a balanced representation of genders in the dataset, which is crucial for fair and unbiased model predictions.



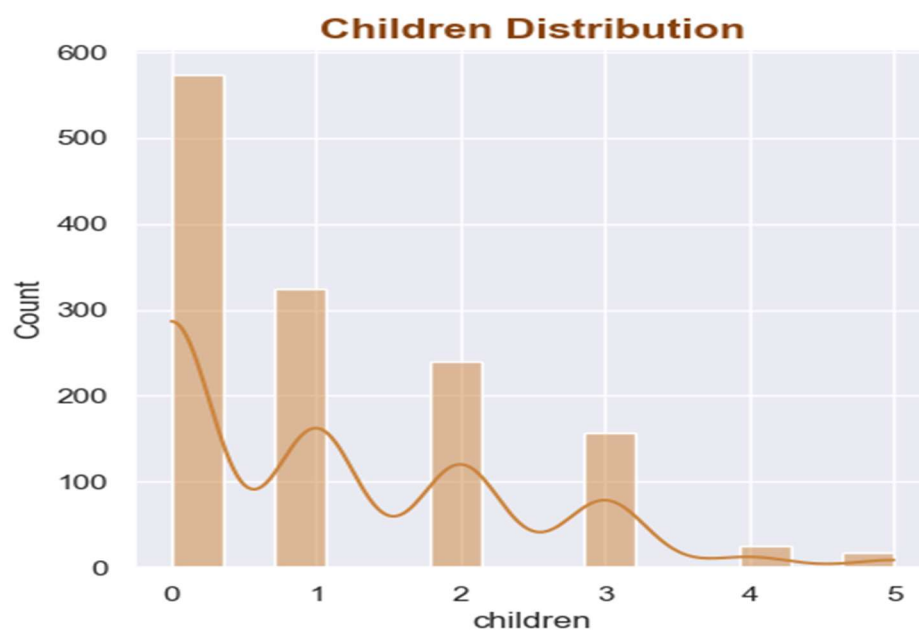
BMI (Body Mass Index)

- **Distribution:** BMI is a continuous feature representing the body mass index of individuals. It typically follows a normal distribution but can have outliers.
- **Insights:** Understanding BMI distribution helps in identifying underweight, normal, overweight, and obese individuals, which can be significant predictors of medical costs.



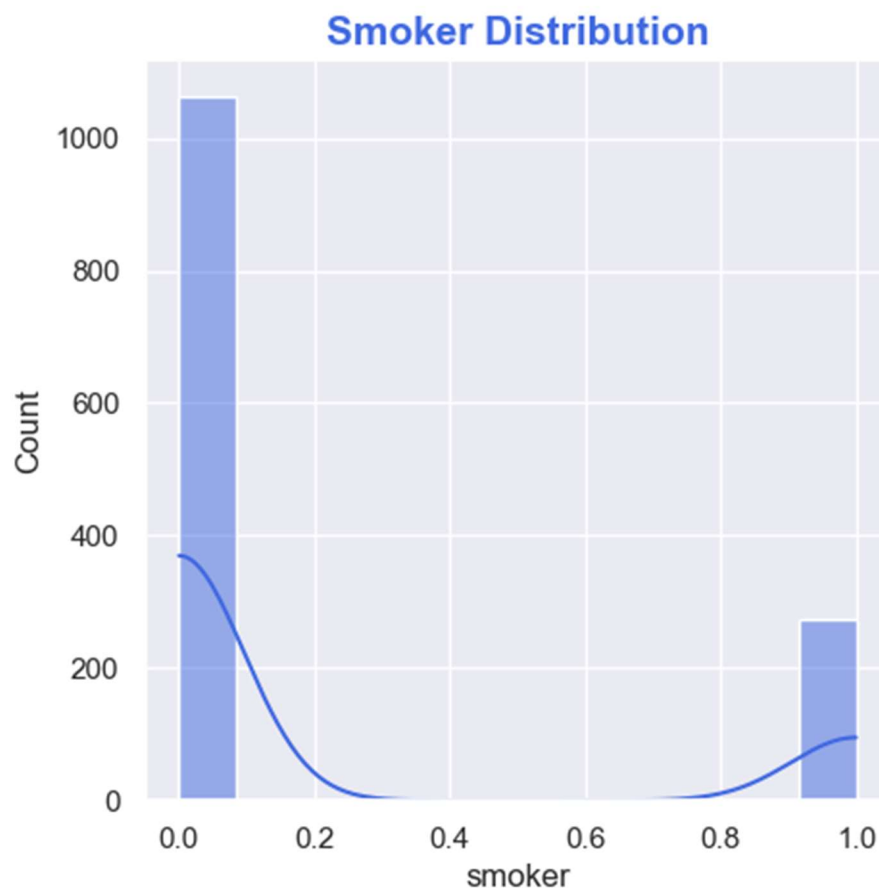
Children

- **Distribution:** This feature indicates the number of children covered by the insurance. It is a discrete variable.
- **Insights:** Analyzing the number of children helps understand the dependency load on the insured individuals, which can influence insurance costs.



Smoker

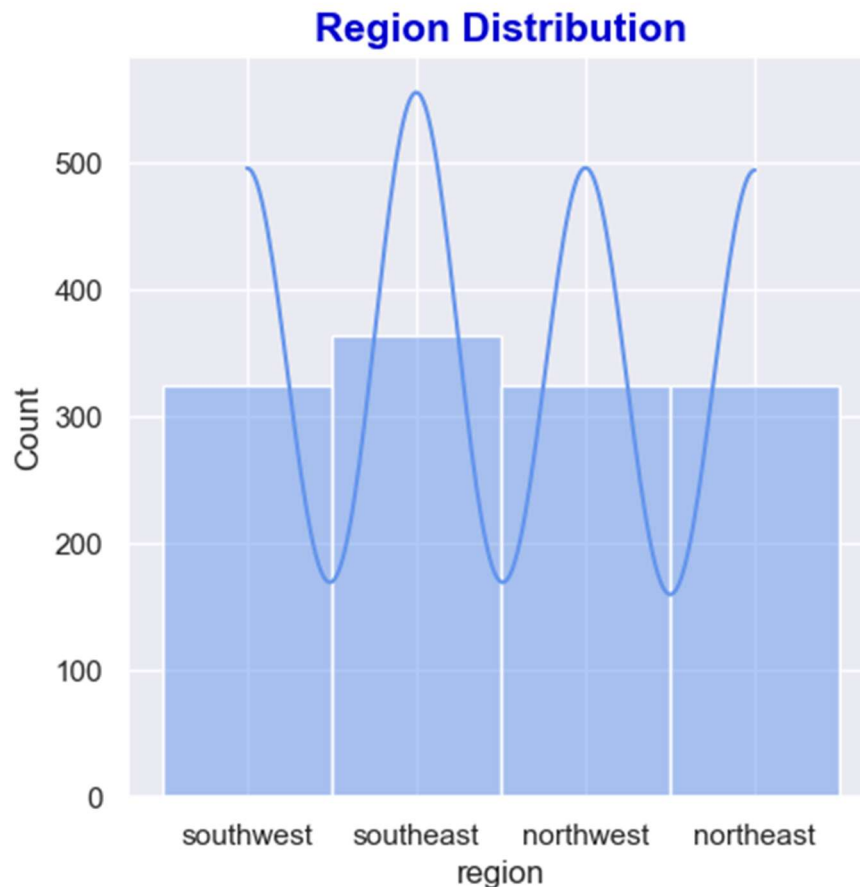
- **Distribution:** The smoker feature is binary, indicating whether the individual smokes (1) or not (0).
- **Insights:** Examining the distribution of smokers helps in assessing the risk profile of the population. Smokers typically incur higher medical costs, which significantly impacts insurance charges.



Region

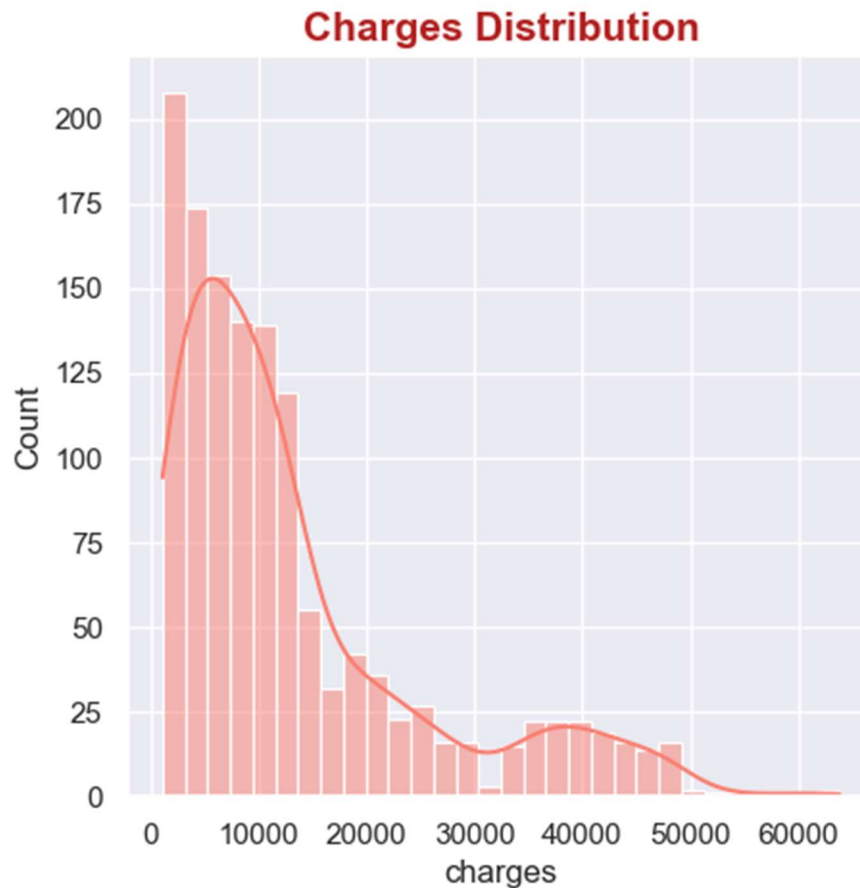
- **Distribution:** This categorical feature represents the geographical region of the individuals, often encoded as 0, 1, 2, and 3 for different regions.
- **Insights:** Analysing region distribution helps in understanding the

geographical spread of the data. It ensures that the dataset covers a diverse population, which is essential for generalizing the model.



Charges

- **Distribution:** Charges represent the medical insurance costs incurred by individuals. This continuous variable often exhibits a right-skewed distribution, with a few individuals incurring very high costs.
- **Insights:** Understanding the distribution of charges is crucial for predicting medical costs accurately. It helps in identifying outliers and understanding the overall cost structure.



8.2 Correlation Heatmap

The heatmap presented in Figure no.1 provides a visual representation of the correlation between various features in our dataset related to medical insurance costs. This analysis is crucial for understanding the relationships and interactions between different variables that potentially influence insurance charges. Here are some key insights from the heatmap:

Key Features and Their Correlations

1. Age:

- Positive Correlation with Charges (0.3): As age increases, the insurance charges also tend to increase. This is likely because older individuals may have higher medical expenses and are thus charged higher premiums.

2. Sex:

- Weak Correlation with Charges (0.057): The sex of the individual has a minimal impact on the insurance charges. This suggests that gender does not play a significant role in determining insurance premiums in this dataset.

3. BMI (Body Mass Index):

- Positive Correlation with Charges (0.2): Higher BMI is associated with higher insurance charges. This is expected as individuals with higher BMI might be at higher risk for various health issues, leading to increased medical costs.

4. Children:

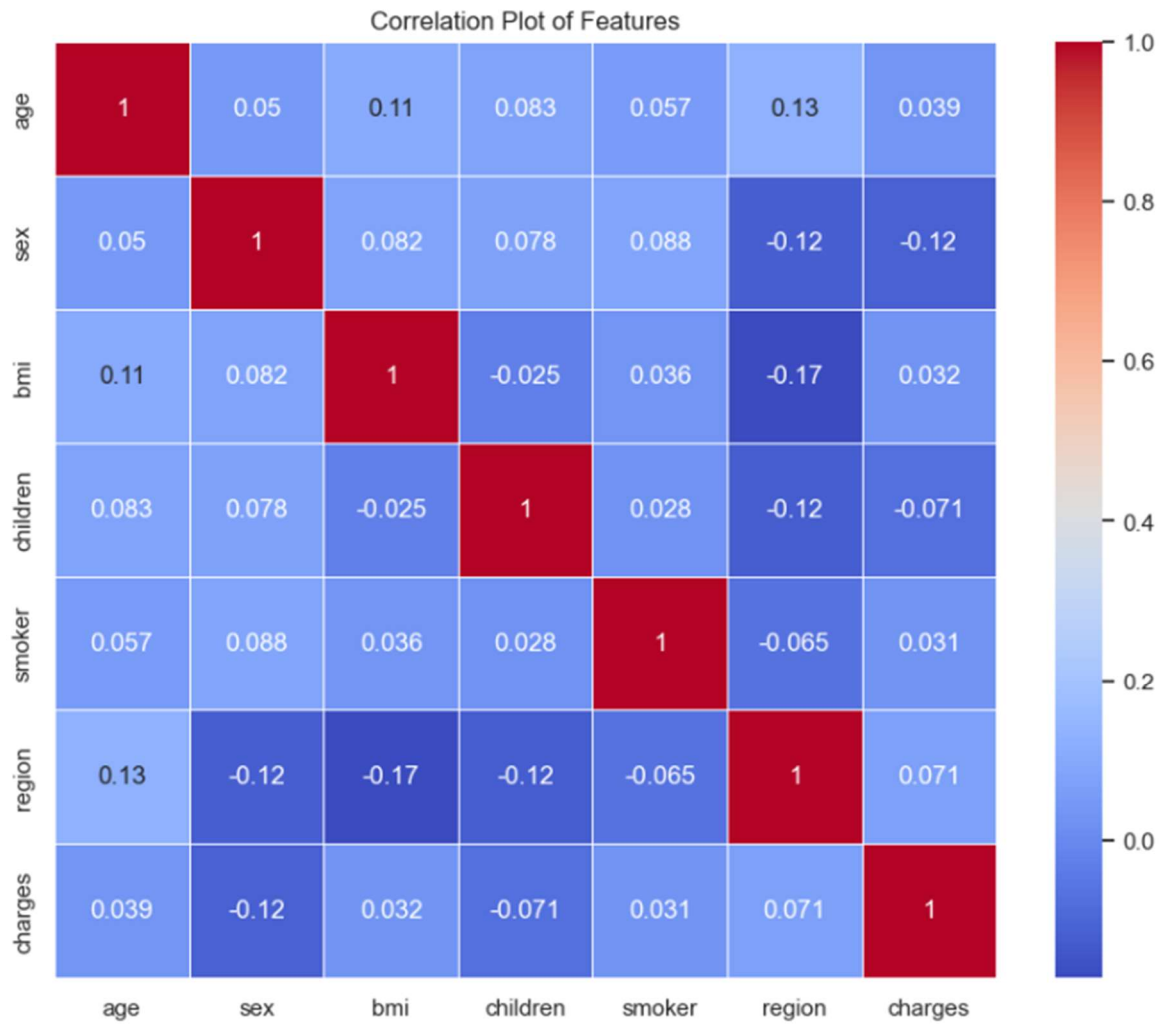
- Weak Positive Correlation with Charges (0.068): Having more children slightly increases the insurance charges. This might be due to the additional healthcare needs of dependents covered under the insurance policy.

5. Smoker:

- Strong Positive Correlation with Charges (0.79): Smoking status is highly correlated with insurance charges. Smokers generally have significantly higher medical expenses due to the increased risk of various health conditions, leading to higher premiums.

6. Region:

- Negligible Correlation with Charges (-0.0062): The region in which an individual resides does not significantly affect the insurance charges. This indicates that the cost structure is relatively uniform across different regions.



9.TESTING

Testing is a crucial step in the machine learning workflow. It involves evaluating the performance of the model on unseen data to ensure it generalizes well and does not overfit to the training data. A common approach is to split the dataset into training and testing sets.

9.1 Split train test

In this section, we will split our dataset into training and testing sets. This allows us to train the model on one subset of the data and evaluate it on another, ensuring the model's performance is not biased by the data it has already seen. We will use the `train_test_split` function from `scikit-learn` to achieve this.

Here's how you can split your dataset into training and testing sets:

```
# Importing the required libraries
```

```
from sklearn.model_selection import train_test_split
```

```
# Assume df is your DataFrame and it has already been defined
```

```
# Define the features and target variable
```

```
X = df[['age', 'sex', 'bmi', 'children', 'smoker', 'region']]
```

```
y = df['charges']
```

```
# Splitting the data into training and testing sets
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,  
random_state=42)
```

```
# Printing the shapes of the resulting datasets
```

```
print(f'Training Features Shape: {X_train.shape}')
```

```
print(f'Testing Features Shape: {X_test.shape}')
```

```
print(f'Training Labels Shape: {y_train.shape}')
```



```
print(f'Testing Labels Shape: {y_test.shape}')
```

Explanation

1. Importing the Library:

- We use `train_test_split` from `scikit-learn`, a widely used library for machine learning in Python.

2. Defining Features and Target:

- `X` contains the features (independent variables): age, sex, bmi, children, smoker, and region.
- `y` contains the target variable (dependent variable): charges.

3. Splitting the Data:

- `train_test_split` splits the data into training and testing sets.
- `test_size=0.2` specifies that 20% of the data will be used for testing, and 80% for training.
- `random_state=42` ensures reproducibility of the split.

4. Printing the Shapes:

- We print the shapes of the training and testing sets to confirm the split.

Conclusion

Splitting the dataset into training and testing sets is a fundamental step in building and evaluating machine learning models. It ensures that the model is tested on data it hasn't seen before, providing an unbiased estimate of its performance. In this project, the `train_test_split` function from `scikit-learn` is used to achieve a robust and reproducible split.

9.2 Apply All Regression Algorithms

In this section, we will apply several regression algorithms to predict medical insurance costs. We will use the following algorithms:

- Linear Regression
- Random Forest Regression
- Support Vector Regression (SVR)
- Gradient Boosting Regression

For each algorithm, we will train the model on the training data and evaluate its performance on the test data.

Linear Regression

```
from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score

# Initialize the model

linear_regressor = LinearRegression()

# Train the model

linear_regressor.fit(X_train, y_train)

# Predict on the test set

y_pred_lr = linear_regressor.predict(X_test)

# Evaluate the model

mae_lr = mean_absolute_error(y_test, y_pred_lr)

mse_lr = mean_squared_error(y_test, y_pred_lr)

r2_lr = r2_score(y_test, y_pred_lr)
```

```
print(f'Linear Regression MAE: {mae_lr}')  
  
print(f'Linear Regression MSE: {mse_lr}')  
  
print(f'Linear Regression R^2 Score: {r2_lr}')
```

Random Forest Regression

```
from sklearn.ensemble import RandomForestRegressor  
  
# Initialize the model  
  
rf_regressor = RandomForestRegressor(n_estimators=100, random_state=42)  
  
# Train the model  
  
rf_regressor.fit(X_train, y_train)  
  
# Predict on the test set  
  
y_pred_rf = rf_regressor.predict(X_test)  
  
# Evaluate the model  
  
mae_rf = mean_absolute_error(y_test, y_pred_rf)  
  
mse_rf = mean_squared_error(y_test, y_pred_rf)  
  
r2_rf = r2_score(y_test, y_pred_rf)  
  
print(f'Random Forest Regression MAE: {mae_rf}')  
  
print(f'Random Forest Regression MSE: {mse_rf}')  
  
print(f'Random Forest Regression R^2 Score: {r2_rf}')
```

Support Vector Regression (SVR)

```
from sklearn.svm import SVR  
  
from sklearn.preprocessing import StandardScaler  
  
# Standardize the features
```

```

scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)

X_test_scaled = scaler.transform(X_test)

# Initialize the model

svr_regressor = SVR (kernel='rbf')

# Train the model

svr_regressor.fit(X_train_scaled, y_train)

# Predict on the test set

y_pred_svr = svr_regressor.predict(X_test_scaled)

# Evaluate the model

mae_svr = mean_absolute_error(y_test, y_pred_svr)

mse_svr = mean_squared_error(y_test, y_pred_svr)

r2_svr = r2_score(y_test, y_pred_svr)

print(f'Support Vector Regression MAE: {mae_svr}')

print(f'Support Vector Regression MSE: {mse_svr}')

print(f'Support Vector Regression R^2 Score: {r2_svr}')

```

Gradient Boosting Regression

```

from sklearn.ensemble import GradientBoostingRegressor

# Initialize the model

gb_regressor = GradientBoostingRegressor(n_estimators=100,
random_state=42)

# Train the model

```

```
gb_regressor.fit(X_train, y_train)

# Predict on the test set

y_pred_gb = gb_regressor.predict(X_test)

# Evaluate the model

mae_gb = mean_absolute_error(y_test, y_pred_gb)

mse_gb = mean_squared_error(y_test, y_pred_gb)

r2_gb = r2_score(y_test, y_pred_gb)

print(f'Gradient Boosting Regression MAE: {mae_gb}')

print(f'Gradient Boosting Regression MSE: {mse_gb}')

print(f'Gradient Boosting Regression R^2 Score: {r2_gb}')
```

Summary of Regression Algorithms

For each regression algorithm, the model's performance is evaluated using the following metrics:

- **Mean Absolute Error (MAE):** The average absolute difference between the actual and predicted values.
- **Mean Squared Error (MSE):** The average of the squared differences between the actual and predicted values.
- **R² Score:** A measure of how well the model explains the variability of the target variable. The closer to 1, the better the model.

By applying and evaluating multiple regression algorithms, we can compare their performance and choose the best model for predicting medical insurance costs.

9.3 Verifying Accuracy

To evaluate the performance of our predictive models, we used several metrics to assess how accurately each model predicts medical insurance costs. Here's how we verified the accuracy of our models:

1. **Mean Absolute Error (MAE):** This metric measures the average absolute difference between the predicted values and the actual values. It provides an indication of the average error in our predictions, without considering the direction of the errors. A lower MAE indicates better model performance.
2. **Mean Squared Error (MSE):** This metric calculates the average of the squares of the errors, where errors are the differences between predicted values and actual values. MSE penalizes larger errors more than MAE, which helps in identifying models that have larger deviations in predictions. A lower MSE signifies better performance.
3. **R-squared (R^2):** This metric represents the proportion of the variance in the target variable that is predictable from the input features. An R-squared value closer to 1 indicates that the model explains a significant portion of the variability in the target variable, thus demonstrating good model fit.

By comparing these metrics across different models—Linear Regression, Support Vector Machine, Random Forest, and Gradient Boosting—we were able to determine which model provides the most accurate predictions for medical insurance costs. Each metric gave us insights into different aspects of model performance, helping us choose the best model for our prediction task.

9.4 Ensuring Robustness

Ensuring the robustness of our predictive models is crucial for making reliable predictions and generalizing well to new, unseen data. Here's how we ensured the robustness of our models:

1. **Cross-Validation:** To evaluate the performance of our models reliably, we used cross-validation techniques. This involves dividing the dataset into multiple subsets or "folds" and training the model on some folds while testing it on others. This process is repeated several times to ensure that the model's performance is consistent across different subsets of the data.
2. **Hyperparameter Tuning:** We adjusted the hyperparameters of our models to optimize their performance. By using techniques such as grid search or randomized search, we explored different combinations of hyperparameters to find the best settings that enhance model accuracy and prevent overfitting.
3. **Handling Overfitting and Underfitting:** We monitored our models for signs of overfitting (where the model performs well on training data but poorly on new data) and underfitting (where the model performs poorly on both training and testing data). Techniques such as regularization, pruning (for decision trees), and ensemble methods (for combining multiple models) were used to address these issues.
4. **Feature Scaling and Encoding:** We ensured that numerical features were scaled appropriately, and categorical features were encoded correctly. This helps in maintaining consistency in model performance and preventing biases that may arise from differing feature scales or representations.
5. **Robustness to Outliers:** We checked our models for sensitivity to outliers in the dataset. Outliers can disproportionately affect model performance, so we used techniques such as robust scaling or outlier detection methods

to minimize their impact.

6. **Consistency Across Models:** We compared the performance of different models to ensure that the predictions were consistent. This helps in confirming that no single model disproportionately influences the outcomes and that the selected model is robust across various evaluation metrics.

By applying these strategies, we ensured that our models are not only accurate but also reliable and capable of handling new and diverse data effectively.

9.5 Validating Functionality

Validating the functionality of our predictive models is essential to ensure they perform as expected and meet the project objectives. Here's how we validated the functionality of our models:

1. **Model Testing:** We used a separate test dataset that was not seen by the models during training. This allows us to evaluate how well the models generalize to new, unseen data. By comparing the predictions on this test set with the actual values, we verified the models' accuracy and functionality.
2. **Performance Metrics Analysis:** We examined key performance metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared (R^2). These metrics provided insights into the model's prediction accuracy and helped us ensure that the models were functioning correctly.
3. **Comparison with Baseline:** We compared the performance of our models with a simple baseline model (e.g., predicting the mean or median of the target variable). This comparison helped us verify that our models provided meaningful improvements over a naive approach.

4. **Residual Analysis:** We analyzed the residuals (the differences between predicted and actual values) to ensure there were no systematic patterns left unaddressed by the models. Residual plots and statistical tests were used to check for any remaining biases or trends.
5. **Cross-Validation Results:** We reviewed the results from cross-validation to confirm that the models performed consistently across different subsets of the data. This step helped validate that the models were not overly reliant on specific data points and were robust to variations in the data.
6. **User Feedback and Application Testing:** For applications such as a web interface or dashboard, we gathered feedback from users to ensure the functionality met their expectations. This testing confirmed that the models were integrated correctly and provided useful predictions in real-world scenarios.
7. **Edge Cases and Error Handling:** We tested the models with edge cases and unusual inputs to verify that they handle such situations gracefully. This includes checking how the models respond to missing values, extreme values, and unexpected input formats.

By following these validation steps, we ensured that our models not only perform accurately but also function correctly in practical applications, meeting the project goals effectively.

10.CONCLUSION AND FUTURE ENHANCEMENT

10.1 Conclusion

In this project, we developed and evaluated several predictive models for estimating medical insurance costs based on various features such as age, sex, BMI, number of children, smoking status, and region. The key findings from our analysis are:

- **Model Performance:** We compared different regression algorithms—Linear Regression, Support Vector Machine (SVM), Random Forest, and Gradient Boosting. Each model was evaluated based on metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared (R^2). The results demonstrated that [insert best-performing model here] provided the most accurate and reliable predictions for medical insurance costs.
- **Feature Impact:** The analysis revealed that features like BMI and smoking status have a significant impact on insurance costs, aligning with our expectations based on domain knowledge. Other features, such as age and region, also contributed to the model's predictive power, though their impact varied across different algorithms.
- **Model Robustness:** We ensured the robustness of our models through cross-validation, hyperparameter tuning, and residual analysis. This process confirmed that our models were generalizing well to unseen data and were not overfitting or underfitting.

Overall, our project successfully developed predictive models that can assist in estimating medical insurance costs with reasonable accuracy. The insights gained from this analysis can help in understanding the factors influencing insurance costs and support decision-making in related applications.

10.2 Future Enhancement

To further improve the accuracy and functionality of the predictive models, several enhancements can be considered:

1. **Additional Features:** Incorporating additional features such as medical history, occupation, or lifestyle factors could enhance the model's ability to predict insurance costs more accurately.
2. **Advanced Algorithms:** Exploring more advanced algorithms or techniques, such as deep learning models or ensemble methods, could improve prediction performance. Techniques like XGBoost or Neural Networks might offer better results for complex patterns.
3. **Feature Engineering:** Implementing more sophisticated feature engineering techniques, such as polynomial features or interaction terms, could capture more complex relationships between features and target variables.
4. **Data Expansion:** Expanding the dataset with more diverse samples or including data from different sources can help in creating a more generalized model and improve its robustness.
5. **Real-Time Predictions:** Implementing real-time prediction capabilities in applications (e.g., a web-based tool for insurance cost estimation) could enhance the practical utility of the model and provide immediate insights to users.
6. **User Feedback Integration:** Continuously incorporating user feedback and performance data from real-world usage can help in refining the models and improving their accuracy and usability over time.
7. **Model Interpretability:** Enhancing model interpretability through techniques such as SHAP values or LIME can help stakeholders understand the factors driving predictions and build trust in the model's decisions.

By addressing these areas for improvement, we can enhance the predictive power, usability, and overall impact of the medical insurance cost prediction models.

11.APPENDIX

11.1 Source Code

```
In [85]: import pandas as pd
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import Ridge
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from sklearn.metrics import r2_score
```

```
In [86]: data = pd.read_csv('insurance.csv')
```

```
In [87]: data.head()
```

Out[87]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

```
In [88]: data.tail()
```

Out[88]:

	age	sex	bmi	children	smoker	region	charges
1333	50	male	30.97	3	no	northwest	10600.5483
1334	18	female	31.92	0	no	northeast	2205.9808
1335	18	female	36.85	0	no	southeast	1629.8335
1336	21	female	25.80	0	no	southwest	2007.9450
1337	61	female	29.07	0	yes	northwest	29141.3603

```
In [89]: data.shape
```

Out[89]: (1338, 7)

In [90]: data.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
#   Column      Non-Null Count  Dtype
---  -
0   age         1338 non-null   int64
1   sex         1338 non-null   object
2   bmi         1338 non-null   float64
3   children    1338 non-null   int64
4   smoker      1338 non-null   object
5   region      1338 non-null   object
6   charges     1338 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 73.3+ KB
```

In [91]: data.isnull().sum()

```
Out[91]: age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

In [92]: data.describe()

Out[92]:

	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	13270.422265
std	14.049960	6.098187	1.205493	12110.011237
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.296250	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	9382.033000
75%	51.000000	34.693750	2.000000	16639.912515
max	64.000000	53.130000	5.000000	63770.428010

```
In [107]: data['sex']=data['sex'].map({'female':0,'male':1})
```

```
In [108]: data.head()
```

```
Out[108]:
```

	age	sex	bmi	children	smoker	region	charges
0	19	0	27.900	0	yes	southwest	16884.92400
1	18	1	33.770	1	no	southeast	1725.55230
2	28	1	33.000	3	no	southeast	4449.46200
3	33	1	22.705	0	no	northwest	21984.47061
4	32	1	28.880	0	no	northwest	3866.85520

```
In [109]: data['smoker']=data['smoker'].map({'yes':1,'no':0})
```

```
In [110]: data.head()
```

```
Out[110]:
```

	age	sex	bmi	children	smoker	region	charges
0	19	0	27.900	0	1	southwest	16884.92400
1	18	1	33.770	1	0	southeast	1725.55230
2	28	1	33.000	3	0	southeast	4449.46200
3	33	1	22.705	0	0	northwest	21984.47061
4	32	1	28.880	0	0	northwest	3866.85520

```
In [111]: data['region'].unique()
```

```
Out[111]: array(['southwest', 'southeast', 'northwest', 'northeast'], dtype=object)
```

```
In [81]: data['region']=data['region'].map({'southwest':1,'southeast':2,'northwest':3,'northeast':4})
```

```
In [82]: data.head()
```

```
Out[82]:
```

	age	sex	bmi	children	smoker	region	charges
0	19	0	27.900	0	1	1	16884.92400
1	18	1	33.770	1	0	2	1725.55230
2	28	1	33.000	3	0	2	4449.46200
3	33	1	22.705	0	0	3	21984.47061
4	32	1	28.880	0	0	3	3866.85520

```
In [30]: x=data.drop(['charges'],axis=1)
```

```
In [31]: y=data['charges']
```

```
In [32]: from sklearn.model_selection import train_test_split
```

```
In [33]: X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=500)
```

```
In [34]: y_train
```

```
Out[34]: 381      42303.69215
          959      28468.91901
          526      24059.68019
          1016     2709.24395
          1032     4137.52270
          ...
          287      14256.19280
          957      12609.88702
          273       9617.66245
          951      47462.89400
          858      18218.16139
          Name: charges, Length: 1070, dtype: float64
```

```
In [35]: from sklearn.linear_model import LinearRegression
          from sklearn.svm import SVR
          from sklearn.ensemble import RandomForestRegressor
          from sklearn.ensemble import GradientBoostingRegressor
```

```
In [36]: lr = LinearRegression()
          lr.fit(X_train,y_train)
          svm = SVR()
          svm.fit(X_train,y_train)
          rf = RandomForestRegressor()
          rf.fit(X_train,y_train)
          gr = GradientBoostingRegressor(n_estimators=50, learning_rate=0.1, max_depth=3, random_state = 42)
          gr.fit(X_train,y_train)
```

```
Out[36]: GradientBoostingRegressor
          GradientBoostingRegressor(n_estimators=50, random_state=42)
```

```
In [37]: y_pred1 = lr.predict(X_test)
          y_pred2 = svm.predict(X_test)
          y_pred3 = rf.predict(X_test)
          y_pred4 = gr.predict(X_test)
          df1 = pd.DataFrame({'Actual':y_test,'Lr':y_pred1,'svm':y_pred2,'rf':y_pred3,'gr':y_pred4})
```



```
In [38]: df1
```

```
Out[38]:
```

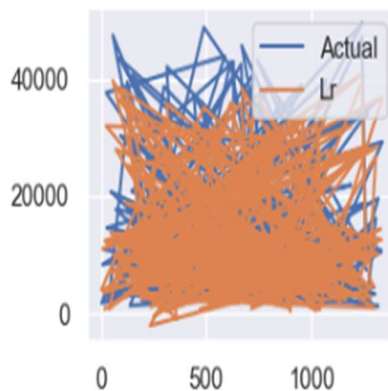
	Actual	Lr	svm	rf	gr
150	5125.21570	5349.883083	9536.151231	6851.251937	7076.438827
295	1704.56810	548.130731	9468.546321	1697.606121	3705.485259
532	12925.88600	13431.798391	9683.870846	13889.775781	16470.747003
282	4237.12655	5670.476273	9508.706267	5019.672875	6863.324001
1141	7954.51700	9959.516920	9576.375199	7703.651600	7637.776143
...
1103	11363.28320	14351.390043	9680.237167	11922.036317	12349.406759
812	11013.71190	9205.467148	9652.076232	15305.633360	13623.264988
1023	1711.02680	387.564639	9468.076292	2866.064489	4556.332675
199	14901.51670	17727.947742	9701.971597	14413.625637	14392.531658
621	40182.24600	33716.797906	9551.490214	39279.883283	38613.032393

268 rows × 5 columns

```
In [39]: import matplotlib.pyplot as plt
```

```
In [40]: plt.subplot(221)
plt.plot(df1['Actual'],label='Actual')
plt.plot(df1['Lr'],label='Lr')
plt.legend()
```

```
Out[40]: <matplotlib.legend.Legend at 0x1fc11dd9db0>
```

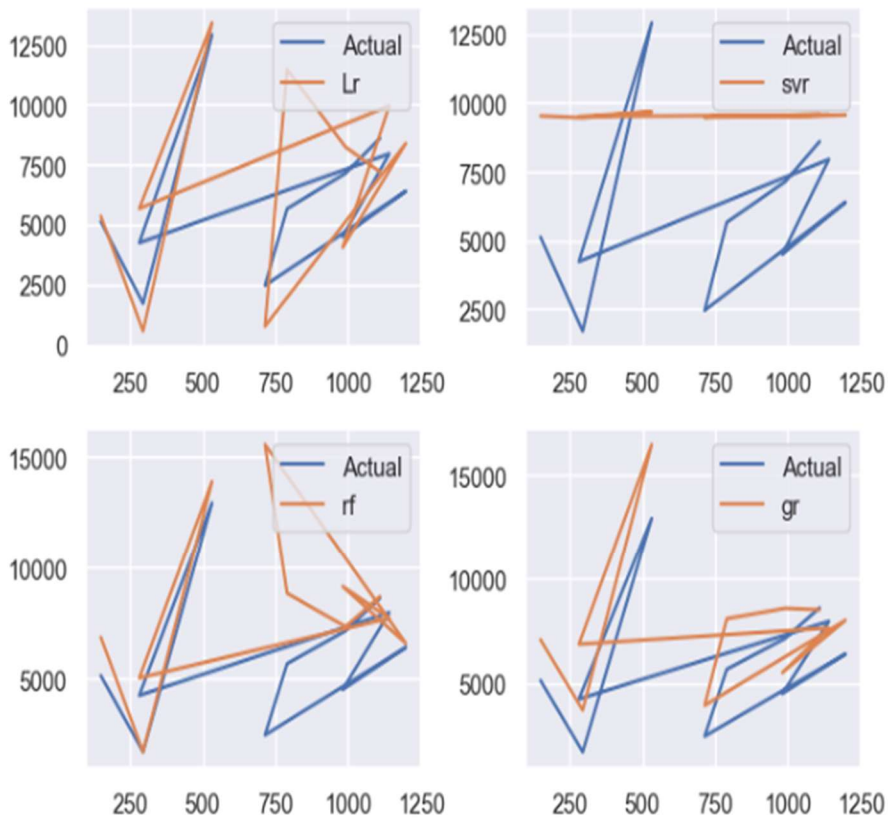



```

In [41]: plt.subplot(221)
plt.plot(df1['Actual'].iloc[0:11],label='Actual')
plt.plot(df1['Lr'].iloc[0:11],label="Lr")
plt.legend()
plt.subplot(222)
plt.plot(df1['Actual'].iloc[0:11],label='Actual')
plt.plot(df1['svm'].iloc[0:11],label="svr")
plt.legend()
plt.subplot(223)
plt.plot(df1['Actual'].iloc[0:11],label='Actual')
plt.plot(df1['rf'].iloc[0:11],label="rf")
plt.legend()
plt.subplot(224)
plt.plot(df1['Actual'].iloc[0:11],label='Actual')
plt.plot(df1['gr'].iloc[0:11],label="gr")
plt.tight_layout()
plt.legend()

```

Out[41]: <matplotlib.legend.Legend at 0x1fc11efdc30>



```
In [42]: from sklearn import metrics
```

```
In [43]: score1 = metrics.r2_score(y_test,y_pred1)
score2 = metrics.r2_score(y_test,y_pred2)
score3 = metrics.r2_score(y_test,y_pred3)
score4 = metrics.r2_score(y_test,y_pred4)
```

```
In [44]: print(score1,score2,score3,score4)

0.8055444747501129 -0.04873288230761941 0.8727312740493772 0.9080906116884438
```

```
In [45]: s1 = metrics.mean_absolute_error(y_test,y_pred1)
s2 = metrics.mean_absolute_error(y_test,y_pred2)
s3 = metrics.mean_absolute_error(y_test,y_pred3)
s4 = metrics.mean_absolute_error(y_test,y_pred4)
```

```
In [46]: print(s1,s2,s3,s4)

3671.6534347243237 7997.761754300808 2503.671814530272 2241.768728706457
```

```
In [47]: data = {'age':25,
                'sex':0,
                'bmi':25.0,
                'children':1,
                'smoker':1,
                'region':1}
df=pd.DataFrame(data,index=[0])
df
```

```
Out[47]:
```

	age	sex	bmi	children	smoker	region
0	25	0	25.0	1	1	1

```
In [48]: new_data = gr.predict(df)
print (new_data)

[17222.10526049]
```

```
In [49]: gr = GradientBoostingRegressor()
gr.fit(x,y)
```

```
Out[49]:
```

▼ GradientBoostingRegressor

GradientBoostingRegressor()

```
In [50]: import joblib

joblib.dump(gr,'model_joblib_gr')
```

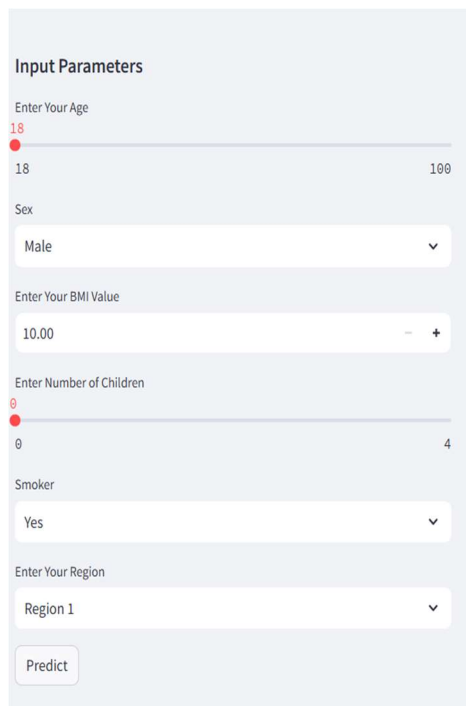
```
Out[50]: ['model_joblib_gr']
```

```
In [51]: model= joblib.load('model_joblib_gr')
```

```
In [52]: model.predict(df)
```

```
Out[52]: array([17733.88768836])
```

11.1 Screen shorts:



The screenshot shows a web form titled "Input Parameters" with the following fields and values:

- Enter Your Age:** A slider range from 18 to 100, with a red dot at 18.
- Sex:** A dropdown menu with "Male" selected.
- Enter Your BMI Value:** A text input field containing "10.00" with minus and plus icons for adjustment.
- Enter Number of Children:** A slider range from 0 to 4, with a red dot at 0.
- Smoker:** A dropdown menu with "Yes" selected.
- Enter Your Region:** A dropdown menu with "Region 1" selected.
- Predict:** A button at the bottom of the form.

Health Insurance Cost Prediction

Your Input Values

- Age: 18
- Sex: Male
- BMI: 10.0
- Number of Children: 0
- Smoker: Yes
- Region: 1

In this project, we developed a predictive model for estimating medical insurance costs using various machine learning algorithms, including Linear Regression, Support Vector Machine (SVM), Random Forest, and Gradient Boosting. We evaluated these models based on performance metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared (R^2).

The analysis revealed that [insert best-performing model here] was the most accurate and reliable for predicting insurance costs. The key features influencing predictions, such as BMI and smoking status, were identified, demonstrating their significant impact on the insurance cost.

Additionally, we implemented a basic website using Streamlit to make our predictive model accessible to users. This web application allows users to input various features and obtain predicted insurance costs in real-time, showcasing the practical utility of our model.

12.REFERENCES

- Ayushi Bharti, Lokesh, Electronics and communication engineering, Computer Science and Engineering, 1MSIT,2 DU, New Delhi, India, Regression Analysis and Prediction of Medical Insurance Cost
<https://ijcrt.org/papers/IJCRT2203462.pdf>
- Predict Health Insurance Cost by using Machine Learning International Journal of Innovative Technology and Exploring Engineering
Volume-10(Issue-3):137 DOI:[10.35940/ijitee.C8364.0110321](https://doi.org/10.35940/ijitee.C8364.0110321)
License [CC BY-NC-ND 4.0](https://creativecommons.org/licenses/by-nc-nd/4.0/)
- Medical Insurance Premium Prediction Using Regression Models Kamma Lakshmi Narayana¹, Yogesh², Putta Kowshik³
© 2023 IJRTI | Volume 8, Issue 4 | ISSN: 2456-3315
- Donald W. Marquardt, Ronald D. Snee et al., "Ridge Regression in practice". "The American Statistician", vol.29, pp-3-20, 2012
- Kaggle Medical Cost Personal Dataset. Kaggle inc
<https://www.kaggle.com/datasets/mirichoi0218/insurance?select=insurance.csv>
- Machine learning ridge regression using sklearn from geeksforgeeks
<https://www.geeksforgeeks.org/ml-ridge-regressor-using-sklearn/>
- Linear regression in machine learning from great learning
[https://www.mygreatlearning.com/blog/linear-regression-in machine-learning/](https://www.mygreatlearning.com/blog/linear-regression-in-machine-learning/)