# Medical Insurance Predictive Modelling: An Analysis of Machine Learning Methods

Project Report

Submitted to the Faculty of

**JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY KAKINADA**

In partial fulfilment of the requirements for the award of the Degree of

**BACHELOR OF TECHNOLOGY**

IN

**Artificial Intelligence and Data Science**

By


**R. Sai Nithin**
**20481A5448**

**M. Venkatesh**
**20481A5431**

**V. Kondala Rao**
**20481A5461**

**K. Sai Ram**
**21485A5404**

Under the Supervision of

**Mr. K. Ashok Reddy** **M.Tech, (PhD)**
Assistant Professor, Department of AI&DS

**DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**

**SESHADRI RAO GUDLAVALLERU ENGINEERING COLLEGE**
**(An Autonomous Institute with Permanent Affiliation to JNTUK, Kakinada)**

**SESHADRIRAO KNOWLEDGE VILLAGE**

**GUDLAVALLERU – 521 356**

**ANDHRA PRADESH**


**2022-2023**

# DEPARTMENT OF

# ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

## SESHADRI RAO GUDLAVALLERU ENGINEERING COLLEGE

**(An Autonomous Institute with Permanent Affiliation to JNTUK, Kakinada)**

SHESHADRI RAO KNOWLEDGE VILLAGE

GUDLAVALLERU-521356

## CERTIFICATE

This is to certify that the community service project report entitled **"Medical Insurance Predictive Modelling: An Analysis of Machine Learning Methods"**is a bonafide record of work carried out by **R. Sai Nithin (20481A5448), M. Venkatesh (20481A5431), V. Kondala Rao (20481A5461), K. Sai Ram (21485A5404)** under the guidance and supervision of **Mr. K. Ashok Reddy M.Tech, (PhD)** partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Artificial Intelligence and Data Science** of **Seshadri Rao Gudlavalleru Engineering College, Gudlavalleru.**

**Project Guide**

**Mr. K. Ashok Reddy M.Tech,(PhD)**
**Assistant Professor**

**Head of the Department**

**Dr. K. Srinivas M.Tech, PhD**
**Professor & H.O.D**

# ACKNOWLEDGMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of people who made it possible and whose constant guidance and encouragements crown all the efforts with success.

We would like to express our deep sense of gratitude and thanks to **Mr. K. Ashok Reddy M.Tech,(PhD)**, Assistant professor, Department of Artificial Intelligence and Data Science for his constant guidance, supervision and motivation in completing the project work.

We feel elated to express our floral gratitude and sincere thanks to **Dr. K. Srinivas M.Tech, PhD,** Head of the Department, Artificial Intelligence and Data Science for his encouragements all the way during analysis of the project. His annotations, insinuations and criticisms are the key behind the successful completion of the project work.

We would like take opportunity to thank our beloved principal **Dr. G.V.S.N.R.V. Prasad M.Tech, M.S, PhD** for providing great support for us in completing our project and giving us the opportunity to do the project.

Our Special thanks to the faculty of our department and programmers of our computer lab. Finally, we thank our family members, non-teaching staff, attendants and our friends, who have directly or indirectly helped and supported us in completing our project in time.

**Project Associates**

| | |
|---|---|
| **R. Sai Nithin** | **20481A5448** |
| **M. Venkatesh** | **20481A5431** |
| **V. Kondala Rao** | **20481A5461** |
| **K. Sai Ram** | **21485A5404** |

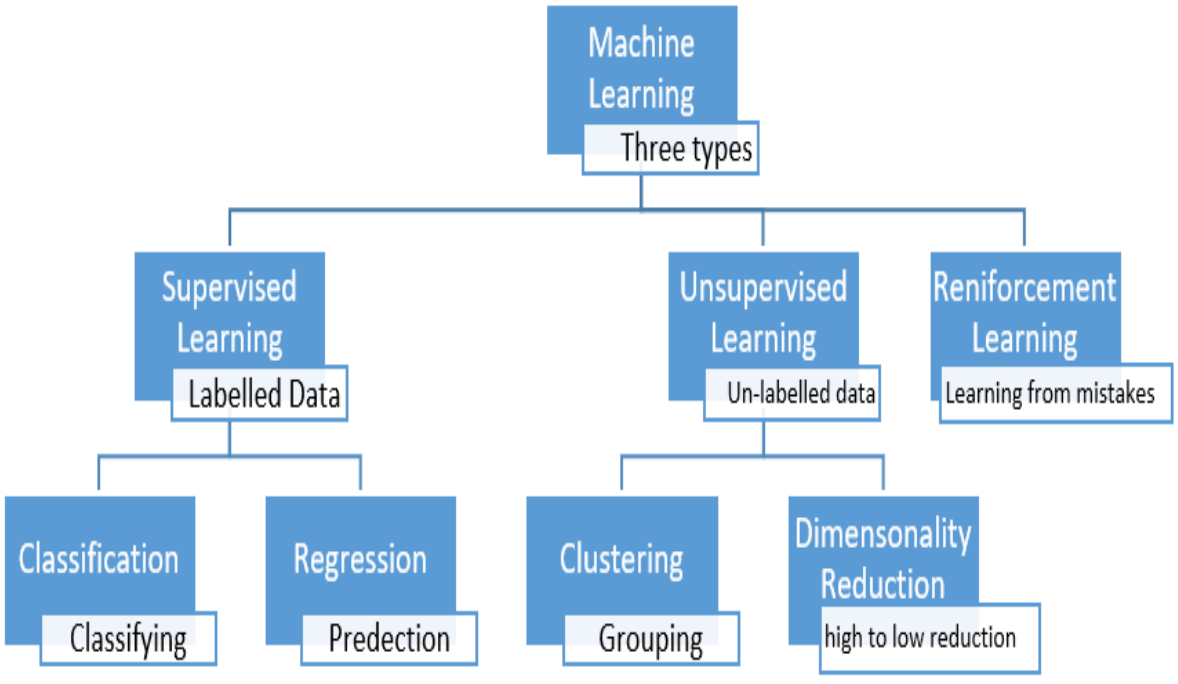
# ABSTRACT

Accurately predicting healthcare expenditures is critical for a wide variety of stakeholders and healthcare organisations. Massive amounts of data on patients, symptoms, and diagnoses are generated by the healthcare business, but this data has yet to be adequately examined to offer insight into patient healthcare spending. In the insurance sector, machine learning (ML) may assist improve policy wording efficiency, and ML algorithms are especially useful in healthcare for forecasting high-cost, high-need patient bills. Machine learning is classified into three types: supervised machine learning (for classification/regression), unsupervised machine learning (for clustering), and reinforcement learning (for decision-making). This study provides a computational intelligence strategy to calculating healthcare insurance expenses using a collection of machine learning algorithms. The Kaggle repository is used to obtain the dataset on medical insurance costs, and various regression models such as Stochastic Gradient Boosting (SGB), XGBoost (XG), Bayesian Ridge Regressor (BR), Decision Tree (DT), Random Forest Regressor (RFR), Multiple Linear Regression (MLR), and k-Nearest Neighbours (KNN), Voting Regressor (VR) are used to forecast insurance costs and compare the models' accuracy.

**Key words:** Machine Learning, Regression, Insurance, Voting Regressor, Ridge Regressor, Stochastic Gradient Boosting, XGBoost, Decision Tree, Random Forest Regressor, Multiple Linear Regression, and k-Nearest Neighbours.

# CHAPTER-I
# INTRODUCTION

This article discusses the importance of insurance in protecting individuals and businesses from a variety of risks such as illness and property damage. The importance of precision in estimating premiums and determining the total covered on insurance policies is emphasised. By training predictive models using historical insurance data, machine learning (ML) has emerged as a viable tool for enhancing the accuracy of insurance cost estimation. This article focuses on estimating health insurance prices, particularly as health care expenditures rise and health care becomes more necessary throughout the world as health uncertainty increases. The ML model used in this work to test the accuracy of several regression models is characterised as a useful tool for estimating necessary health and health care expenses. Choosing the correct ML method that can select the efficient prediction system of the health insurance policy by comparing the data corresponding to the developer after the extension of the ML algorithm, which is known to predict the cost of medical assistance in approximating the calculations based on ML.



Machine learning algorithms are especially good at forecasting high-cost, high-need client expenditures in healthcare. There are three forms of machine learning. These are supervised machine learning (a task-driven approach) for classification/regression with all data labelled; unsupervised machine learning (a data-driven approach) for clustering with all data unlabelled; and reinforcement learning (learning from mistakes) for decision making.

### 1.1. Regression

The regression analysis is a predictive method that explores the relationship between a dependent (target) and the independent variable(s) (predictor). This technology is used to forecasting, estimate model time series, and find the causal effect relationship among the variables. In this analysis, for example, I want to analyse the relationship between insurance cost (target variable) and six independent variables based on (Age, BMI, child number, individual living area, or sex and whether the customer is a smoking person). on the basis of a regression.

The regression analysis estimates the relationship between two or more variables, as stated previously. I used different regression models to estimate health insurance costs on the basis of six independent variables, and by using this regression, we can forecast future health insurance fees based on current and past data. There are several advantages of using regression analysis as follows:

- It demonstrates the essential relationships between the dependent and independent variables.
- It shows the effect intensity on the dependent variable of several independent variables.

Analysis of regression also helps one to compare the results of measured variables at various scales, such as independent variable and dependent variable effects. These advantages allow market researchers, data analysts, and data scientists to remove and determine the best range of variables for predictive model.

# CHAPTER-II
# LITERATURE REVIEW

Because of their ability to properly estimate medical expenditures, Machine learning algorithms have grown in popularity in the healthcare business. Choosing a proper algorithm and executing acceptable methods to construct and deploy the model, on the other hand, are important to its performance.It is critical to carefully analyse and pick the best machine learning algorithm for a specific healthcare context, as well as to ensure that right protocols are followed during model construction and deployment.

Machine learning methods have all been proposed for prediction of medical insurance. We present an overview of some of the related studies in this topic in this section

Sudhir Panda's [1] shows that the Polynomial Regression has an accuracy of 80.97%. by forecasting each patient's medical insurance. The accuracy of the model depends on various factors such as the quality and size of the dataset, the selection of features, and the complexity of the model.

[2] system projected by Mukund Kulkarni, shows that Gradient Boosting offers the best efficiency, with an accuracy of 86.86, it can be used in the estimation of insurance costs with better performance than other regression models.

Ensemble techniques are a popular machine learning approach for improving the accuracy and robustness of predictive models. The idea behind ensemble techniques is to combine the predictions of multiple models to create a more accurate and reliable prediction.

In Paper [3] Nataliya Shakhovska1 uses the stacking is developed using K Nearest Neighbours (KNN), Support Vector Machine (SVM), Regression Tree, Linear Regression, Stochastic Gradient Boosting. The random forest (RF) algorithm is used to combine the predictions. It produces the accuracy of 96%

[4] Chinthala Shreekar uses the most efficient model is polynomial regression, which has an accuracy of 80.97%, to improve outcomes, the health insurance system looks into predictive modelling.

# CHAPTER-III
# METHODOLOGY

## 3.1 Regression Models

The following Regression models are used in this analysis

- Multiple Linear Regression (MLR)
- Random Forest Regressor (RFR)
- XGBoost (XG)
- Decision Tree (DT)
- Bayesian Ridge Regressor (BR)
- k-Nearest Neighbours (KNN)
- Stochastic Gradient Boosting (SGB)
- Voting Regressor (VR)

### 3.1.1 Multiple Linear Regression

Multiple Linear Regression (MLR) is a statistical approach used to examine the connection between a dependent variable and numerous independent factors. It is an extension of simple linear regression that takes into account only one independent variable. MLR is an effective tool for predicting the value of a dependent variable based on the values of two or more independent variables. Main focus of univariate regression is analysing the relationship between a dependent variable and one independent variable and formulates the linear relation equation between dependent and independent variable. The dependent variable is denoted Y in the MLR model, and the independent variables are denoted X1, X2, ... Xt. The MLR model's goal is to determine the best collection of weights ($\alpha_0$, $\alpha_1$, $\alpha_2$, ..., $\alpha_t$) that may be used to predict the value of the dependent variable Y given the values of the independent variables X1, X2, ..., Xt. The MLR model implies that the dependent variable and each independent variable have a linear relationship. The model also presumes that the residuals are normally distributed with a mean of zero. The residuals are the disparities between the actual and expected values of the dependent variable based on the model.

The Formula for Multiple Linear Regression is

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip} + \epsilon$$

where, for i=n observations:

$y_i$ = Dependent variables

$x_i$ = Explanatory variables

$\beta_0$ = y – Intercept (constant term)

$\beta p$ = slope coefficients for each explanatory variable

$\epsilon$ = The model's error term (also known as the residuals)

### 3.1.2 Random Forest Regressor

Random Forest Regressor is a machine learning algorithm that belongs to the family of ensemble methods. It is a variation of the popular random group classification, but instead of predicting categorical labels, it predicts sequential numeric values. The algorithm builds multiple decision trees, each of which is trained with a random subset of the input attributes and a random subset of the training data. Each decision tree makes a prediction based on the characteristics it was trained on, and the final prediction of the random forest is the average of the predictions of all decision trees. Random Forest Regressors are useful for a variety of regression problems, especially when dealing with large datasets with many input features. They handle missing values, noisy data, and outliers quite well. They are also resistant to overfitting, which can be a problem with other machine learning algorithms when working with complex data sets. The benefits of using a random forest regressor include Robustness, Flexibility, Speed. Overall, random forest regressors are a powerful tool for building accurate regression models and are widely used in various fields, including finance, healthcare, and marketing. As in the following equation, a random model for forest regressors can be expressed

$$g(x)=f0(x)+f1(x)+f2(x)+ \ldots\ldots +fn(x)$$

where g is the final model that is the sum of all models. Each model f(x) is a decision tree.

The benefits of using a random forest regressor include:

- Robustness: Random Forest regressors handle outliers and noisy data well.
- Flexibility: They can be used with different data types, including categorical and continuous variables.
- Speed: The algorithm can efficiently process large datasets with many input features.

- Easy to interpret: Random Forest regressors provide measures of attribute importance that can help identify key variables in the model.

Overall, random forest regressors are a powerful tool for building accurate regression models and are widely used in various fields, including finance, healthcare, and marketing.

### 3.1.3 XGB Regressor

XGBoost (Extreme Gradient Boosting) is a cutting-edge gradient boosting method that has attracted a lot of attention in recent years. It is a scalable and distributed Advanced Gradient Boosting (GBDT) machine learning library. It provides parallel tree boosting and is the leading machine learning library for regression, classification, and classification problems. It is an ensemble learning approach that uses the combined power of numerous decision trees to create correct predictions. The technique operates by adding weak students to the model iteratively, focusing on misclassified examples from the preceding iteration each time. The model learns to fix its faults and increase its accuracy in this way. XGB Regressor is a sophisticated XGBoost implementation intended exclusively for regression workloads. It can handle huge datasets with many characteristics as well as missing values in data. Many hyper settings in XGB Regressor may be changed to boost performance.

N estimators, max_depth, learning_rate, subsample, colsample_bytree, Gamma, reg_alpha reg_lambda are some of the main XGB Regressor hyperparameters. XGBoost was chosen as the predictor model for forecasting medical insurance expenses in this study owing to its effectiveness in managing missing values and its potential for enhancing prediction accuracy. The work intends to create a more exact and efficient way for calculating medical insurance expenses by utilising XGBoost.

$$\hat{Y}_i = \sum^{k}_{k=1} f_k(x_i)$$

$\hat{Y}_i$ is the predicted value for the $i^{th}$ instance
$f_k$ is the prediction of the $k^{th}$ tree on the $i^{th}$ instance
k is the total number of trees in the model

### 3.1.4 Decision Tree Regressor

Decision trees are a popular machine learning approach for classification and regression issues. Decision tree regression is a sort of regression technique that uses input information to predict a continuous value. It operates by constructing a tree-like model of decisions and their potential outcomes. The decision tree is constructed by the algorithm by dividing the data into smaller groups based on the value of one of the input characteristics. As the decision node, the feature that best divides the data into subgroups with the lowest variance is picked. To construct the decision tree, the method selects a feature j and a split points that minimise the sum of squared errors (SSE) of the output variables $y_i$ in each ensuing partition:

$$\min\_SSE\ (j, s) = \sum^{n}_{i=1}(y_i - \hat{y}(R_i))^2$$

$\hat{y}(Ri)$ represents the mean of the output variables $y_i$ in region $R_i$.

The decision tree is constructed iteratively until a stopping requirement, such as a minimum number of samples per leaf node or a maximum depth of the tree, is fulfilled. After constructing the tree, the method may anticipate the output value for a new input by traversing the tree from root to leaf node. The prediction is then the value of the leaf node. The target variable is predicted in decision tree regression by taking the average of the values at the leaf node. This value is used to anticipate the next input value. The technique can handle both continuous and categorical input characteristics and may be utilised for basic and sophisticated regression situations. In summary, decision trees are a powerful tool for predictive modelling, classification, and data analysis. They are used in various applications and industries due to their easy interpretation and understandability. To build an accurate and reliable decision tree, it is essential to collect relevant data, select the right characteristics, and use the right criteria to break down the data into each node. Decision trees are an essential tool for machine learning and data science, and their popularity is expected to continue to grow in the future.

### 3.1.5 Bayesian Rigid regressor

Bayesian regression is a statistical modeling technique used to predict continuous values in machine learning. It is based on Bayesian principles, which define and estimate statistical models. Bayesian regression is especially useful when there is not enough data in the dataset or when the data is poorly distributed. The output of a Bayesian regression model is obtained from a probability distribution rather than a single value of each attribute. Bayesian regression assumes that the parameters and model output come from a distribution. The goal of Bayesian regression is to find the posterior distribution for the model parameters. The posterior distribution is proportional to the probability of the data multiplied by the prior probability of the parameters.

Bayesian crest regression is a specific type of Bayesian regression that uses a Gaussian distribution to model the output. Bayesian Ridge Regression is implemented using a Gamma distribution before the alpha parameter and a Gamma distribution before the lambda parameter. However, the Bayesian approach can be used with any programming language that supports statistical modeling.

Bayesian regression allows a natural mechanism to survive insufficient data or poorly distributed data by formulating linear regression using probability distributors rather than point estimates. The output or response 'y' is assumed to drawn from a probability distribution rather than estimated as a single value. Mathematically, to obtain a fully probabilistic model the response y is assumed to be Gaussian distributed around $X_W$ as follows

$$p\ (y|X, w, \alpha) = N\ (y|X_W, \alpha)$$

One of the most useful types of Bayesian regression is Bayesian Ridge regression which estimates a probabilistic model of the regression problem. Here the prior for the coefficient w is given by spherical Gaussian as follows –

$$p\ (w \mid \lambda) = N\ (w|0, \lambda_{-1}|p)$$

### 3.1.6 k-Nearest Neighbours (KNN)

The supervised learning method K-Nearest Neighbours (KNN) Regressor is utilised for regression applications. A given data point's K-nearest neighbours are located in the training set, and the output variable of the new data point is predicted as the average of its K-nearest neighbours' output variables. The KNN Regressor uses feature vectors to represent the data points, and based on the distance between the feature vectors, the algorithm determines the K nearest neighbours. Euclidean distance is the most often used measure of distance, but other distance metrics, like Manhattan distance and Minkowski distance, can also be applied. K is a hyperparameter that must be chosen prior to training the model. A lower K value results in a more flexible model with higher variance and lower bias, whereas a higher K value results in a more rigid model with lower variance and higher bias. KNN Regressor is a non-parametric algorithm, which means it makes no assumptions about the underlying data distribution. It is also a lazy learning algorithm, which means that no training time is required and that all training data is stored in memory. KNN Regressor can be used to predict property values, stock prices, and medical diagnoses, among other things. However, KNN regression also has some limitations: it can be computationally expensive, especially for large datasets. The performance of the algorithm strongly depends on the choice of the number of neighbours k. It can be sensitive to the magnitude of the input variables.

**Distance functions**

Euclidean $$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

Manhattan $$\sum_{i=1}^{k}|x_i - y_i|$$

Minkowski $$\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$$

### 3.1.7 Stochastic Gradient Boosting

Stochastic Gradient Boosting Regression is a regression task ensemble learning method that combines numerous weak regression models to build a strong prediction model. It is based on the boosting concept, in which weak models are sequentially trained and integrated to build a final model with higher predictive performance. The "stochastic" in Stochastic Gradient Boosting Regression refers to the fact that a random fraction of the training data is used to train the weak model at each iteration. This contributes to less overfitting and better generalisation performance.

Furthermore, the approach employs a learning rate parameter to regulate the contribution of each weak model to the final ensemble. A slower learning rate produces a smoother learning curve and greater generalisation performance, but it takes more cycles to achieve the best results.

Overall, Stochastic Gradient Boosting Regression is a sophisticated and adaptable regression technique capable of handling intricate interactions between input data and the target variable. However, careful tuning of the hyperparameters may be required to attain the optimal performance, and it may be computationally expensive for large datasets.The formula for the prediction of SGR is:

$$\hat{y} = w0 + w1x1 + w2x2 + ... + wnxn$$

where $\hat{y}$ is the predicted value, w0 is the intercept, w1, w2, ..., wn are the coefficients of the model, and x1, x2, ..., xn are the features of the input data.

The loss function used in SGR is typically the mean squared error (MSE), which is defined as:

$$MSE = 1/n * \sum(y_i - \hat{y}_i)^2$$

where n is the number of samples, $y_i$ is the actual value of the i-th sample, and $\hat{y}_i$ is the predicted value of the i-th sample.

The update rule for the coefficients is:

$$wi = wi - \alpha * \partial MSE/\partial wi$$

and the update rule for the intercept is:

$$w0 = w0 - \alpha * \partial MSE/\partial w0$$

where α is the learning rate.

### 3.1.8 Voting Regressor

Voting Regressor is an ensemble learning technique that mixes different regression models in order to improve the model's overall performance. It operates by combining the predictions of several regression models to produce a single prediction. The Voting Regressor averages the predictions of numerous regression models to combine them. It can be used with any regression model, including linear regression, decision tree regression, and support vector regression, among others.

The Voting Regressor has two applications:
Hard Voting: In this approach, each model's output is treated as a vote, and the final forecast is based on the majority vote. For example, if three regression models are used, and two of them predict the output as 3, and one predicts the output as 4, the final prediction will be 3.

Soft Voting: In this method, each model's output is weighted based on its confidence or accuracy. The final forecast is the weighted average of the guesses. Each model's weight is determined by its performance on the training data.

The formula for the prediction of a Voting Regressor can be written as:

$$\hat{y} = w1\hat{y}1 + w2\hat{y}2 + ... + wk*\hat{y}k$$

where $\hat{y}$ is the predicted value, $\hat{y}1$, $\hat{y}2$, ..., $\hat{y}k$ are the predictions of k individual regressors, and w1, w2, ..., wk are the weights assigned to each individual regressor.
The weights w1, w2, ..., wk can be uniform (i.e., equal weights) or can be assigned based on the performance of each individual regressor on a validation set. The final prediction of the Voting Regressor is a weighted average of the individual predictions, which is given by:

$$\hat{y} = 1/k * \sum(wi * \hat{y}i)$$

where k is the number of individual regressors, wi is the weight assigned to the i-th regressor, and $\hat{y}i$ is the prediction of the i-th regressor.

## 3.2 Data Representation

In this study, we used machine learning algorithms on health insurance data from the Kaggle repository. The dataset contains 1338 entries with six attributes: "Age", "Sex", "BMI", "Children", "Smokers", and "Charges". However, the dataset was unsuitable for regression analysis without pre-processing. As a result, it was required to clean the dataset and choose the key features in order to use the data with different regression techniques. Age and smoking status were shown to have the largest impact on volume prediction, with smoking having the biggest effect. Following pre-processing, we performed feature engineering and divided the dataset into training and test datasets, utilising 70% of the total data for training and the remaining 30% for testing**.**

- **Data set Information**

```
<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
```

**Figure 3.2.1**

## 3.3 Data Pre-processing

In order to effectively apply machine learning algorithms to the dataset, the data must be carefully reviewed and updated. The data set consists of seven variables, all of which contribute to the estimate of the Charges, which is the dependent variable. First, all unknown values were adjusted and replaced with the mean of the respective variable. The target variable (Charges) was then examined to better understand its distribution

and possible relationships with the independent variables. The categorical values were converted to numeric values to explore the data set for regression analysis.

| Attribute | Description |
|---|---|
| Age | Present Age of the client |
| Sex | 0- Male<br>1- Female |
| BMI | Body Mass Index |
| Children | Number of Children to the client |
| Smoker | 0- No<br>1- Yes |
| Region | 1-northeast<br>2-northwest<br>3-southeast<br>4-southwest |
| Charges | Medical cost of the client per year |

**Table 3.3.1**

To better understand the correlation between the input or independent variables, age and BMI were divided into specific groups.

**AGE**

| Range | Number of Clients |
|---|---|
| 10-19 | 137 |
| 20-29 | 280 |
| 30-39 | 257 |
| 40-49 | 279 |
| 50-59 | 271 |
| 60-69 | 114 |

**Table 3.3.2**

**REGION**

| Region | Number of clients |
|---|---|
| North-East | 324 |
| North-West | 325 |
| South-East | 364 |
| South-West | 325 |

**Table 3.3.3**

**GENDER**

| Gender | Number of clients |
|---|---|
| Male | 676 |
| Female | 662 |

**Table 3.3.4**

**SMOKER**

| Type (Yes/No) | Number of clients |
|---|---|
| Yes | 274 |
| No | 1064 |

**Table 3.3.5**

**CHILDREN**

| Number of Children | Number of clients |
|---|---|
| 0 | 574 |
| 1 | 324 |
| 2 | 240 |
| 3 | 157 |
| 4 | 25 |
| 5 | 18 |

**Table 3.3.6**

**BMI**

| Category | Number of clients |
|---|---|
| Normal-Weight | 222 |
| Obese | 705 |
| Over-weight | 380 |
| Under-Weight | 20 |

**Table 3.3.7**

This analysis revealed crucial statistics in the data set, such as the number of male and female samples, smokers and non-smokers, and the prevalence of different age groups and BMI categories. It was discovered that the majority of the samples were non-smokers, and that a sizable component of the data set was made up of childless adults. Furthermore, the data looked to be evenly distributed among four areas, with somewhat more samples in the south-eastern region. Finally, based on the BMI ranges used in the study, the majority of samples were classified as obese.

### 3.3.1 Descriptive Statistics of Charges Variable

| Variable Name | Mean | Std | Min | 25% | 50% | 75% | Max |
|---|---|---|---|---|---|---|---|
| Charges | 13270.422 | 12110.011 | 1121.873 | 4740.287 | 9382.033 | 16639.912 | 63770.428 |

These statistics give information on the 'charges' variable's central tendency, variability, and range. The mean of 3270.422 is the average cost of health insurance. The 12110.011 standard deviation suggests that the charges vary greatly throughout the sample. The minimum and maximum values represent the range of charges, while the quartiles show how charges are distributed over the dataset. The median figure of 9382.033 reflects the midpoint of the charged data, with 50% of the data falling below and 50% falling above this amount. as seen in the preceding table.

### 3.3.2 Graphical Representation

#### 3.3.2.1 Age vs Charges

We can see in Figure 1 that with the growing age, the insurance charges are going to be increased. For example, when the age touches 64, the insurance charge is 23000, as shown in Figure 1. Age is shown on the x-axis, and charges are given on the y-axis.

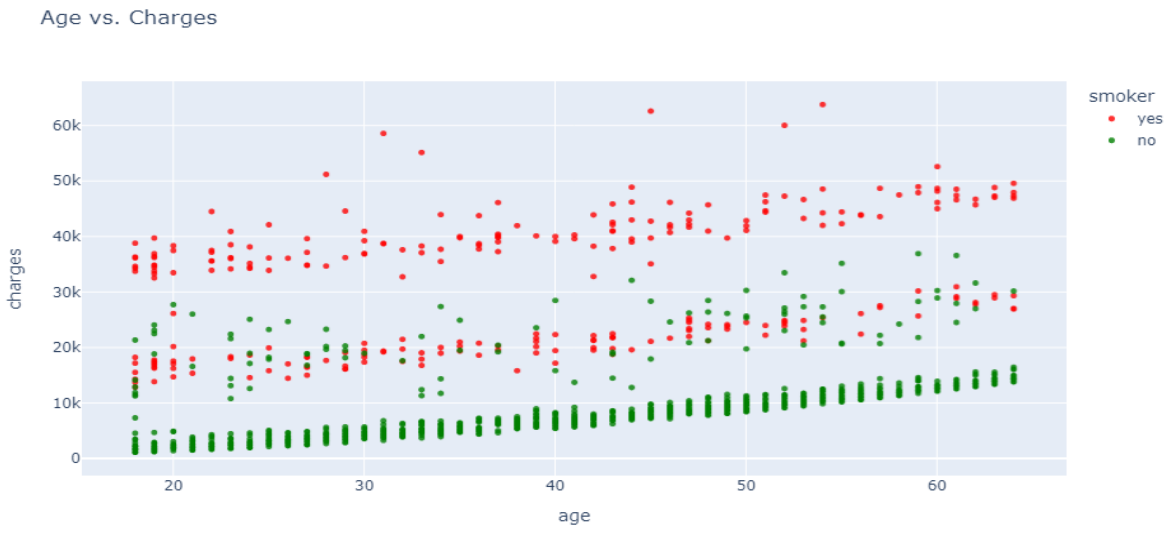



**Figure 1 Age vs Charges**

### 3.3.2.2 Region vs Charges

Insurance charges vary concerning certain regions as shown in Figure 2. The health insurance charges in the southeast are greater than in other regions. The region count is displayed on the *y-axis*, and charges are shown on the *x-axis*.

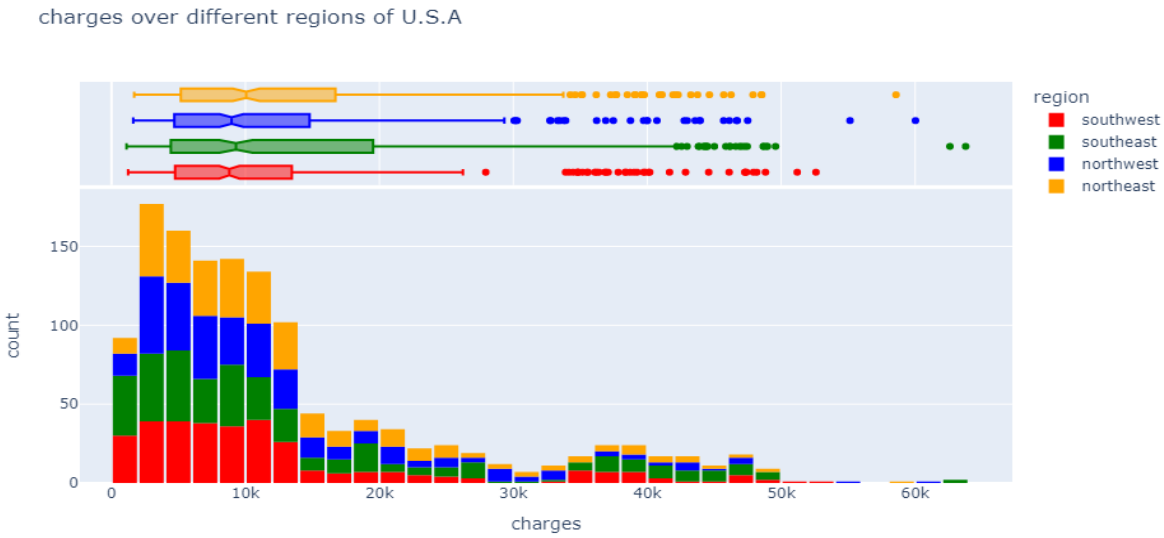



**Figure 2 Region vs Charges**

### 3.3.2.3 BMI vs Charges

In Figure 3, the zero value is used to represent the females and one value is used for the males. The BMI values of smoker types (Yes and No) are given in the *x-axis*, and the charges are presented in the *y-axis*. It can be clearly seen that when the values of BMI are varied, the insurance charges will vary accordingly as shown in Figure 3.

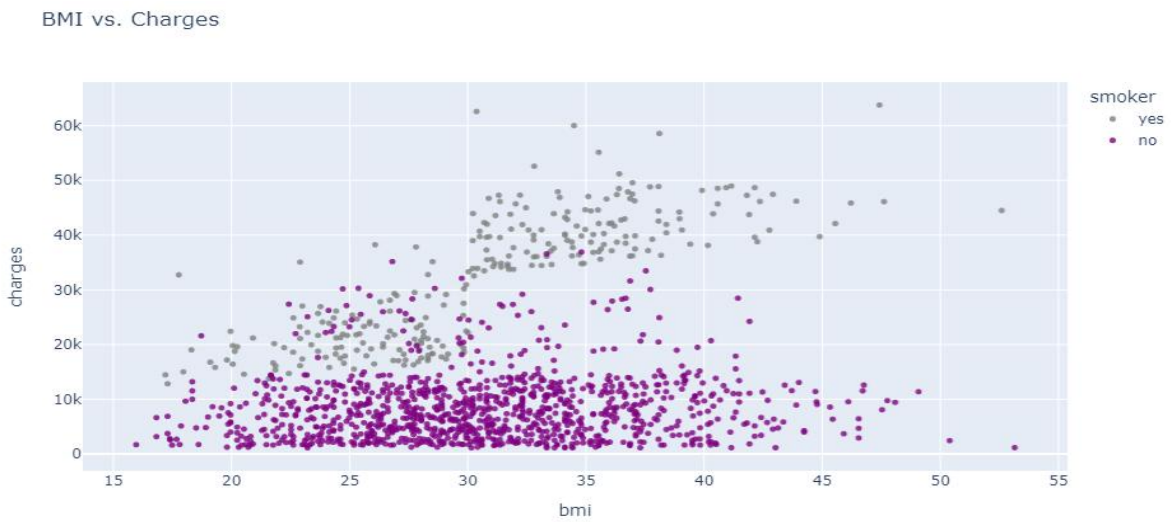



**Figure 3 BMI vs Charges**

### 3.3.2.4 Smoker vs Charges

However, men are more addicted and passionate to smoking as compared to women so the health insurance cost for females is greater as compared to the males. We can see in Figure 4 that with the increase of smoking habits, the insurance charges are going to be decreased for men and increased for women. Smokers based on gender values are shown on the *x-axis*, and charges are shown on the *y-axis*.

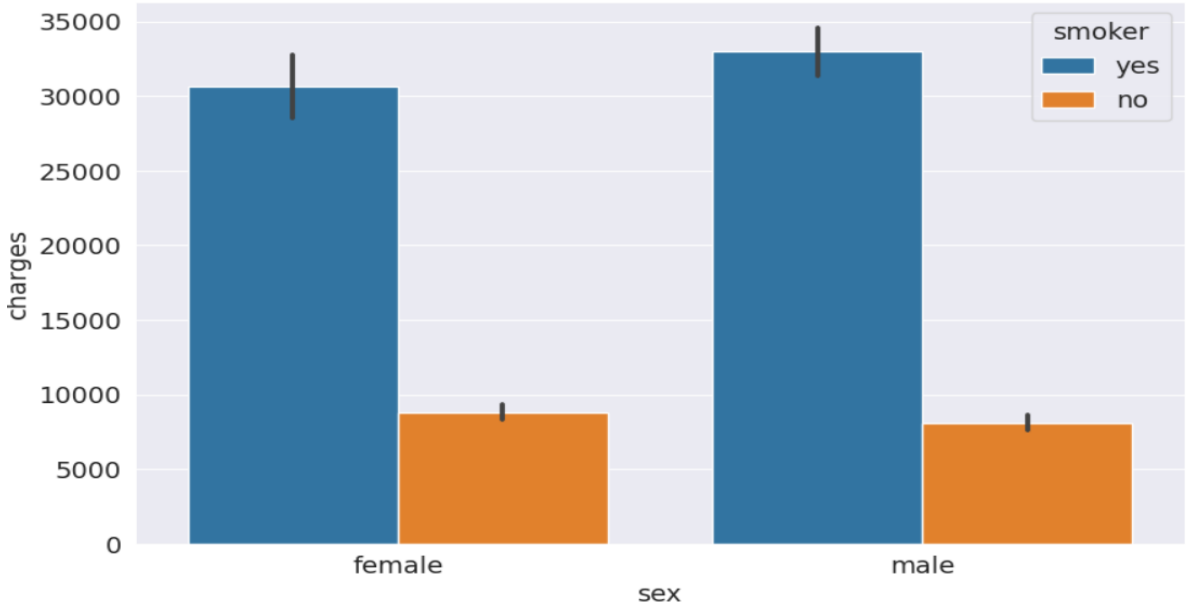



**Figure 4 Smoker vs Charges**

### 3.3.2.5 Sex vs Charges

The medical insurance charges for the female gender are always greater than for the male as shown in Figure 5. It gives the sex types on the *y-axis* and the charges on the *x-axis.* The figure illustrates that the insurances charges for the female are 14000, and for the male, the charges are around 13000.

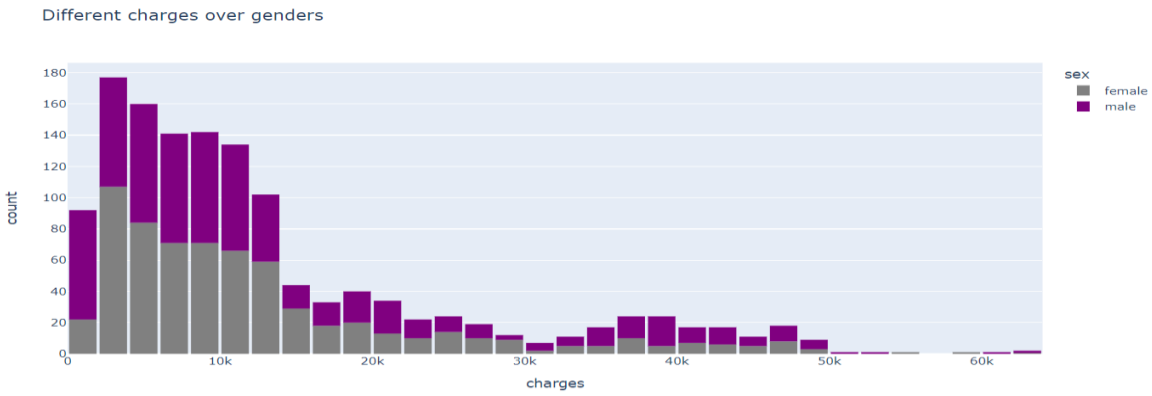



**Figure 5 Sex vs charges**

### 3.3.2.6 Feature Engineering and Correlation Matrix

When it comes to machine learning, feature engineering is the process of extracting features from raw data while applying domain expertise in order to improve the performance of ML algorithms. In the medical insurance cost dataset, attributes such as smoker, BMI, and age are the most important factors that determine charges. Also, we see that sex, children, and region do not affect the charges. We might drop these 3 columns as they have less correlation by plotting the heat map graph to see the dependency of dependent value on independent features. The heat map makes it easy to identify which features are most related to the other features or the target variable. Outcomes are shown in Figure 6.

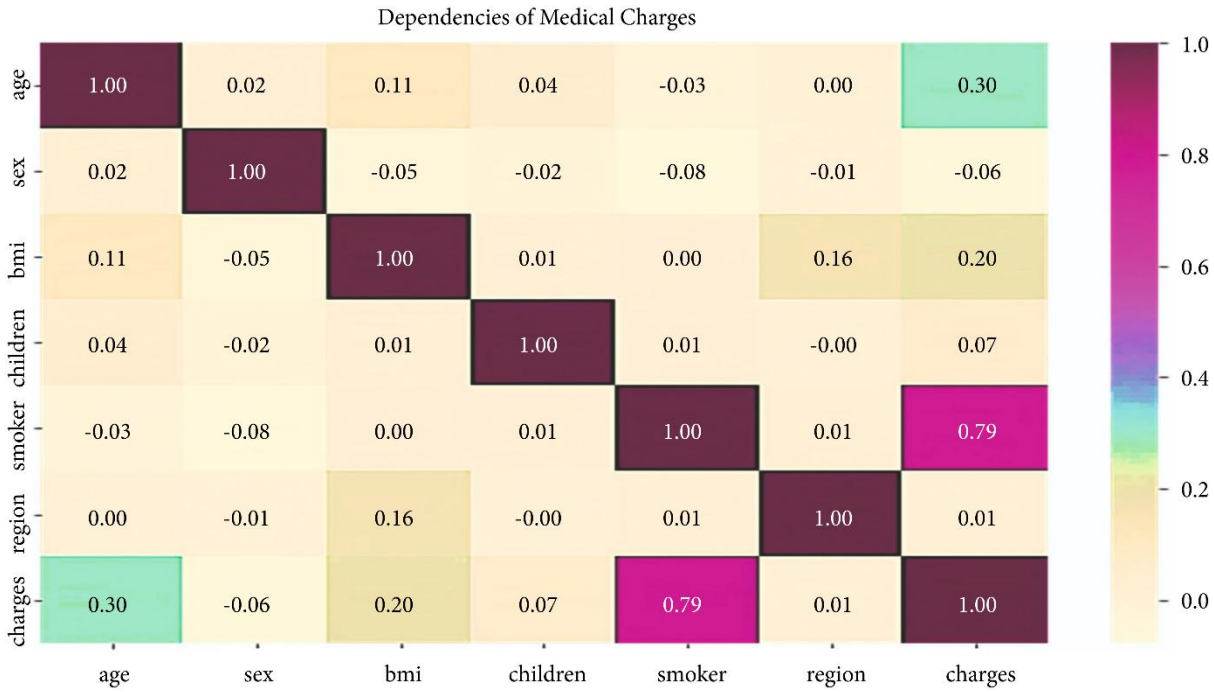



**Figure 6**

## 3.4 Proposed Method

In this study, regression techniques were employed to establish the relationship between a target or dependent variable and a set of independent or predictor variables. It is assumed that both the target and predictor variables have numerical values and that they are related in some way. The following actions were taken for each model:

**Data Preparation:** The dataset was gathered and pre-processed to account for missing values, categorical variables, and outliers. As appropriate, feature engineering techniques such as feature scaling and feature selection were used.

**Model Training:** Using a suitable technique, such as k-fold cross-validation, the dataset was divided into training and testing sets. 70% of the data is in the training model, while the rest is in the testing model. The regression models were then trained on the training data using the appropriate techniques and hyperparameters.

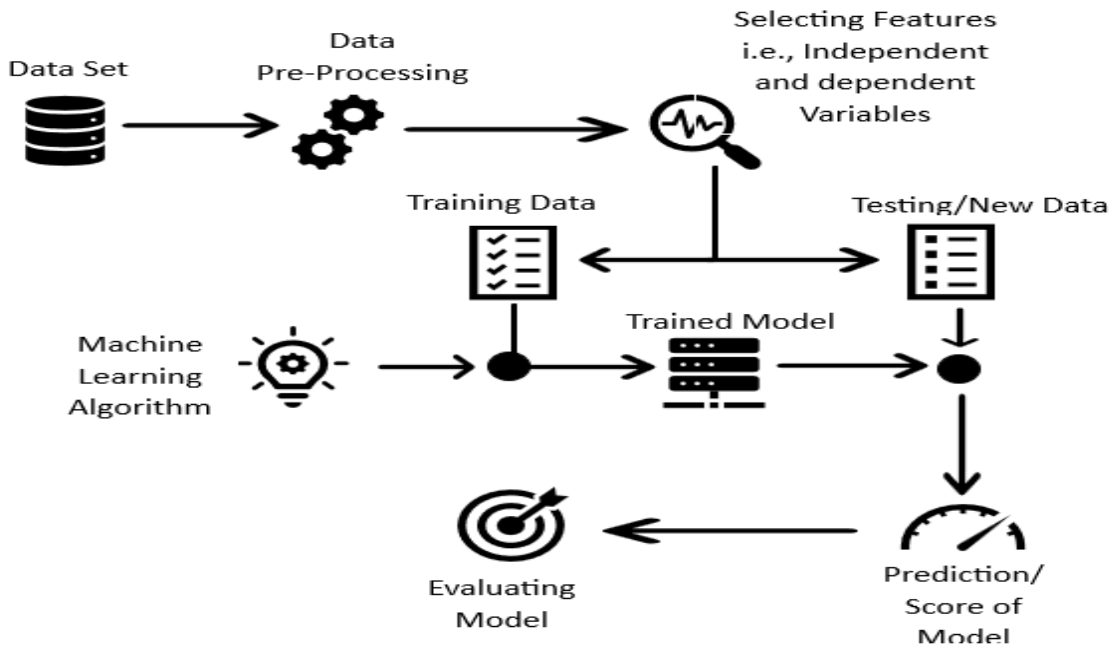



**Figure 3.4.1**

**Model Evaluation:** To determine predictive performance, the trained models were tested on the testing set using appropriate evaluation metrics such as mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), and R-squared (R2) score.

- **Mean absolute Error**

$$MSE = (1/n) * \Sigma (y - \hat{y})^2$$

  A lower MSE indicates higher predictability
- **Mean Square Error**

$$MAE = (1/n) * \Sigma |y - \hat{y}|$$

  When compared to MSE, MAE is less susceptible to outliers.
- **Root mean squared error**

$$RMSE=\sqrt{MSE}$$

- **R2-Score**

$$R2\_Score = 1 - (\Sigma (y - \hat{y})^2 / \Sigma (y - \bar{y})^2)$$

# CHAPTER-IV

# IMPLEMENTATION

## 4.1 CODE DEVELOPED FOR ANALYSIS

### // IMPORTING REQUIRED LIBRARIES

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

### // Data Uploading

```
data=pd.read_csv("insurance.csv")
data.head()
```

| | 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 |

```
print("Size of the data set:",data.shape)
```

Size of the data set: (1338, 7)

```
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
```

### // Data Pre-Processing

```
data.replace({'sex':{'male':1,'female':0}},inplace=True)
```

```
data.replace({'smoker':{'yes':1,'no':0}},inplace=True)
data.replace({'region':{'northeast':1,'northwest':2,'southeast':3,'southwest':4}},inplace=
True)
data
```

| | age | sex | bmi | children | smoker | region | charges | cbmi | Age-Category |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 19 | 0 | 27.900 | 0 | 1 | 4 | 16884.92400 | over-weight | 10-19 |
| 1 | 18 | 1 | 33.770 | 1 | 0 | 3 | 1725.55230 | obese | 10-19 |
| 2 | 28 | 1 | 33.000 | 3 | 0 | 3 | 4449.46200 | obese | 20-29 |
| 3 | 33 | 1 | 22.705 | 0 | 0 | 2 | 21984.47061 | normal-weight | 30-39 |
| 4 | 32 | 1 | 28.880 | 0 | 0 | 2 | 3866.85520 | over-weight | 30-39 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 1333 | 50 | 1 | 30.970 | 3 | 0 | 2 | 10600.54830 | obese | 50-59 |
| 1334 | 18 | 0 | 31.920 | 0 | 0 | 1 | 2205.98080 | obese | 10-19 |
| 1335 | 18 | 0 | 36.850 | 0 | 0 | 3 | 1629.83350 | obese | 10-19 |
| 1336 | 21 | 0 | 25.800 | 0 | 0 | 4 | 2007.94500 | over-weight | 20-29 |
| 1337 | 61 | 0 | 29.070 | 0 | 1 | 2 | 29141.36030 | over-weight | 60-69 |

1338 rows × 9 columns

**// Splitting of data into test and train**

```
x=data.drop(['charges','cbmi','Age-Category'],axis=1)
y=data['charges']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=100)
```

**//Importing the Regression Evaluation metrics Libraries**

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

**//Fitting Multiple linear Regression Model**

```
from sklearn.linear_model import LinearRegression
model1=LinearRegression()
model1.fit(x_train,y_train)
pred1=model1.predict(x_test)
```

**//Model Evaluation**

```
mae = mean_absolute_error(y_test, pred1)
print("MAE:", mae)
# Calculate MSE
mse = mean_squared_error(y_test, pred1)
print("MSE:", mse)
# Calculate RMSE
rmse = np.sqrt(mse)
print("RMSE:", rmse)
```

```
# Calculate R-squared
r_squared = r2_score(y_test, pred1)
print("R-squared:", r_squared)
```

```
MAE: 3935.564251349733
MSE: 32191230.36571921
RMSE: 5673.731608537649
R-squared: 0.778293784216272
```

**// Fitting Random Forest Regressor**

```
from sklearn.ensemble import RandomForestRegressor
model2= RandomForestRegressor(n_estimators = 100, random_state = 100)
model2.fit(x_train, y_train)
pred2=model2.predict(x_test)
```

**// Model Evaluation**

```
mae = mean_absolute_error(y_test, pred2)
print("MAE:", mae)
# Calculate MSE
mse = mean_squared_error(y_test, pred2)
print("MSE:", mse)
# Calculate RMSE
rmse = np.sqrt(mse)
print("RMSE:", rmse)
# Calculate R-squared
r_squared = r2_score(y_test, pred2)
print("R-squared:", r_squared)
```

```
MAE: 2965.769582498216
MSE: 22899447.048167832
RMSE: 4785.336670305218
R-squared: 0.8422878004067951
```

**//Fitting XGB Regressor**

```
from xgboost import XGBRegressor
model3=XGBRegressor()
model3.fit(x_train,y_train)
pred3=model3.predict(x_test)
```

//**Model Evaluation**

```
mae = mean_absolute_error(y_test, pred3)
print("MAE:", mae)
# Calculate MSE
mse = mean_squared_error(y_test, pred3)
print("MSE:", mse)
# Calculate RMSE
rmse = np.sqrt(mse)
```

```
print("RMSE:", rmse)
# Calculate R-squared
r_squared = r2_score(y_test, pred3)
print("R-squared:", r_squared)
```

```
MAE: 3072.546452664014
MSE: 23948217.564613968
RMSE: 4893.691609063037
R-squared: 0.8350647480479628
```

**//Fitting Decision Tree Regressor**

```
from sklearn.tree import DecisionTreeRegressor
model4=DecisionTreeRegressor()
model4.fit(x_train,y_train)
pred4=model4.predict(x_test)
```

**//Model Evaluation**

```
mae = mean_absolute_error(y_test, pred4)
print("MAE:", mae)
# Calculate MSE
mse = mean_squared_error(y_test, pred4)
print("MSE:", mse)
# Calculate RMSE
rmse = np.sqrt(mse)
print("RMSE:", rmse)
# Calculate R-squared
r_squared = r2_score(y_test, pred4)
print("R-squared:", r_squared)
```

```
MAE: 3386.6319191393036
MSE: 49573878.193277046
RMSE: 7040.871976770849
R-squared: 0.6585766741100028
```

**//Fitting Bayesian Rigid Regressor**

```
from sklearn.linear_model import BayesianRidge
model5 = BayesianRidge()
model5.fit(x_train, y_train)
pred5=model5.predict(x_test)
```

//**Model Evaluation**

```
mae = mean_absolute_error(y_test, pred5)
print("MAE:", mae)
# Calculate MSE
mse = mean_squared_error(y_test, pred5)
```

```
print("MSE:", mse)
# Calculate RMSE
rmse = np.sqrt(mse)
print("RMSE:", rmse)
# Calculate R-squared
r_squared = r2_score(y_test, pred5)
print("R-squared:", r_squared)
```

```
MAE: 3940.4465832296605
MSE: 32176130.625101298
RMSE: 5672.4007814241495
R-squared: 0.7783977785747878
```

**//Fitting KNN Regressor**

```
from sklearn.neighbors import KNeighborsRegressor
model6=KNeighborsRegressor()
model6.fit(x_train,y_train)
pred6=model6.predict(x_test)
```

//**Model Evaluation**

```
mae = mean_absolute_error(y_test, pred6)
print("MAE:", mae)
# Calculate MSE
mse = mean_squared_error(y_test, pred6)
print("MSE:", mse)
# Calculate RMSE
rmse = np.sqrt(mse)
print("RMSE:", rmse)
# Calculate R-squared
r_squared = r2_score(y_test, pred6)
print("R-squared:", r_squared)
```

```
MAE: 8156.719261987562
MSE: 134774192.65380326
RMSE: 11609.22877084448
R-squared: 0.07178831317980161
```

**//Fitting Stochastic Gradient Boosting**

```
from sklearn.ensemble import GradientBoostingRegressor
model8 = GradientBoostingRegressor()
model8.fit(x_train,y_train)
```

//**Model Evaluation**

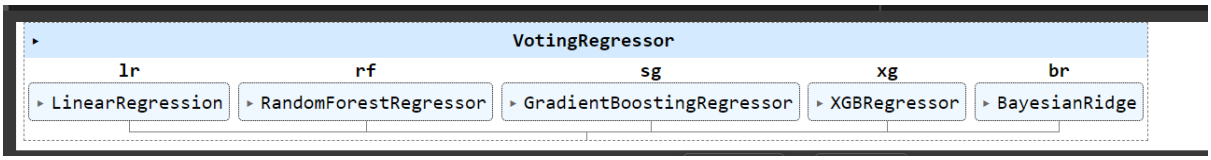
```
pred8=model8.predict(x_test)
mae = mean_absolute_error(y_test, pred8)
print("MAE:", mae)
# Calculate MSE
mse = mean_squared_error(y_test, pred8)
```

```
print("MSE:", mse)
# Calculate RMSE
rmse = np.sqrt(mse)
print("RMSE:", rmse)
# Calculate R-squared
r_squared = r2_score(y_test, pred8)
print("R-squared:", r_squared)
```

```
MAE: 2426.4623055908432
MSE: 15996757.66340632
RMSE: 3999.594687391001
R-squared: 0.8898277398511628
```

**// Fitting Voting Regressor**

```
from sklearn.ensemble import VotingRegressor
vc=VotingRegressor(estimators=[("lr",model1),("rf",model2),("sg",model8),("xg",mo
del3),("br",model5)])
vc.fit(x_train,y_train)
```




**//Model Evaluation**

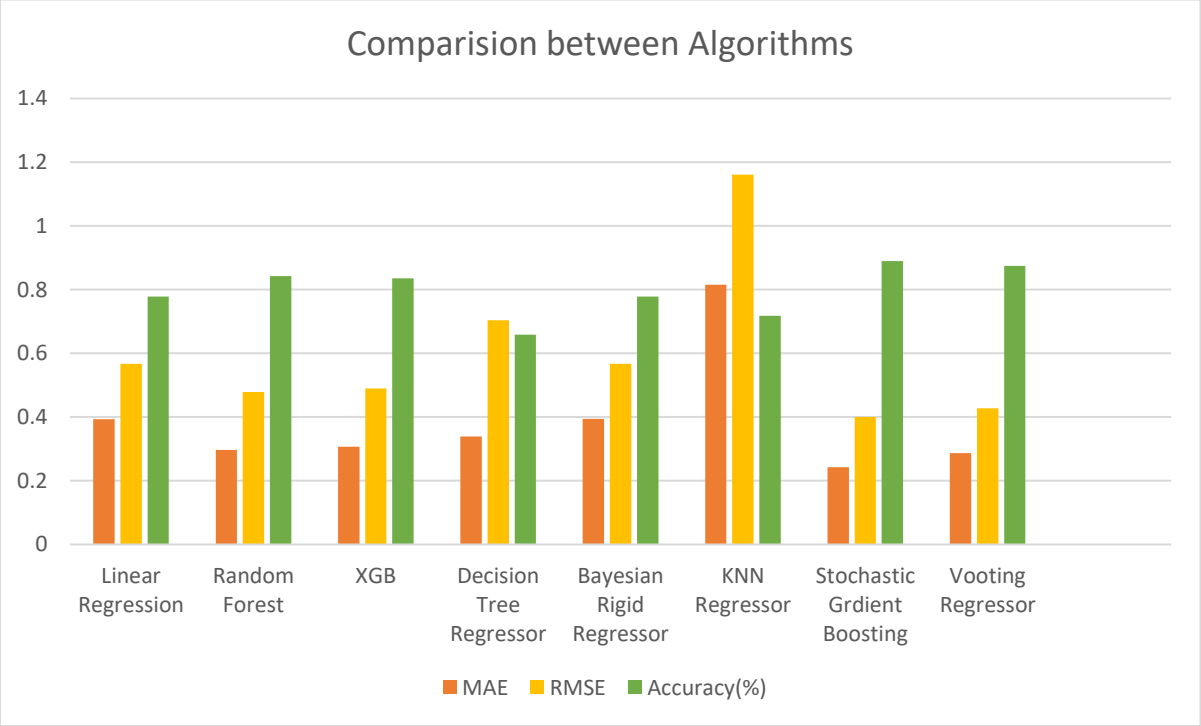
```
pred7=vc.predict(x_test)
mae = mean_absolute_error(y_test, pred7)
print("MAE:", mae)
# Calculate MSE
mse = mean_squared_error(y_test, pred7)
print("MSE:", mse)
# Calculate RMSE
rmse = np.sqrt(mse)
print("RMSE:", rmse)
# Calculate R-squared
r_squared = r2_score(y_test, pred7)
print("R-squared:", r_squared)
```

```
MAE: 2868.3914108846448
MSE: 18251248.13418052
RMSE: 4272.147953217505
R-squared: 0.874300698942279
```

# CHAPTER-V
# RESULT AND DISCUSSION

Using the previously stated methodology, the results of the above-mentioned machine learning algorithms are provided in a table with their MAE, MSE, RMSE, and R2 scores.

| Model Name | MAE | MSE | RMSE | R2-Score | Accuracy (%) |
|---|---|---|---|---|---|
| Linear Regression | 3935.56 | 32191230.36 | 5673.73 | 0.7782 | 77.82 |
| Random Forest | 2965.76 | 22899447.04 | 4785.336 | 0.8422 | 84.22 |
| XGB Regressor | 3072.54 | 23948217.56 | 4893.69 | 0.8350 | 83.50 |
| Decision Tree Regressor | 3386.63 | 49573878.19 | 7040.87 | 0.6585 | 65.85 |
| Bayesian Rigid Regressor | 3940.44 | 32176130.62 | 5672.40 | 0.7783 | 77.83 |
| KNN Regressor | 8156.71 | 134774192.65 | 11609.22 | 0.7178 | 71.78 |
| Voting Regressor | 2868.39 | 18521248.13 | 4272.14 | 0.8743 | 87.43 |
| Stochastic Gradient Boosting | 2426.46 | 15996757.66 | 3999.59 | 0.8898 | 88.98 |



# CHAPTER-IV
# CONCLUSION

Based on the results Stochastic Gradient Boosting model has the greatest accuracy score of 88.98% on the medical insurance data set, followed by the Voting Regressor model with an accuracy score of 87.43%. The accuracy ratings for the Random Forest and XGB Regressor models are likewise relatively good, at 84.22% and 83.50%, respectively.

The Decision Tree Regressor and KNN Regressor models, on the other hand, have lower accuracy ratings of 65.85% and 71.78%, respectively. The Bayesian Rigid Regressor model and Linear Regression model both have reasonable accuracy ratings of 77.83% and 77.82%.

Finally, the Stochastic Gradient Boosting and Voting Regressor models are the most accurate for forecasting medical insurance charges based on the data set. When choosing the appropriate model for a certain use case, various aspects such as model complexity, interpretability, and computing efficiency must be considered.