HEALTH INSURANCE PREMIUM DETECTOR & ANALYSIS OF RELATION BETWEEN PARAMETERS

A Summer internship Report Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY IN
COMPUTER SCIENCE AND ENGINEERING – DATA SCIENCE

Submitted by

Miss. R. Vishnu Priya (21071A6744)

Miss. V. Sindhu Bhargavi (21071A6762)

Miss. Y. Sindhu Priya (21071A6763)

Under the guidance of

Dr . N. Sunanda
(Assistant Professor, Department of CSE- (CYS, DS) and AI&DS)

DEPARTMENT OF CSE - (CYS, DS) and AI&DS

VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

An Autonomous Institute, NAAC Accredited with 'A++' Grade NBA Accredited for CE, EEE, ME, ECE, CSE, EIE, IT, AME B. Tech Courses Approved by AICTE, New Delhi, Affiliated to JNTUH Recognized as "College with Potential for Excellence" by UGC ISO 9001:2015 Certified, QS I GUAGE Diamond Rated

Vignana Jyothi Nagar, Pragathi Nagar, Nizampet (S.O), Hyderabad – 500 090, TS, India

VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

An Autonomous Institute, NAAC Accredited with 'A++' Grade NBA Accredited for CE, EEE, ME, ECE, CSE, EIE, IT, AME B. Tech Courses Approved by AICTE, New Delhi, Affiliated to JNTUH Recognized as "College with Potential for Excellence" by UGC ISO 9001:2015 Certified, QS I GUAGE Diamond Rated

Vignana Jyothi Nagar, Pragathi Nagar, Nizampet (S.O), Hyderabad – 500 090, TS, India

DEPARTMENT OF CSE- (CYS, DS) and AI&DS

CERTIFICATE

This is to certify that the project report entitled "**Health Insurance Premium Detector & analysis of relation between parameters"** is a bonafide work done under our supervision and is being submitted by **Miss. Vishnu Priya(21071A6744), Miss. Sindhu Bhargavi(21071A6762), Miss. Y. Sindhu Priya (21071A6763)** in partial fulfillment for the award of the degree of Bachelor of Technology in CSE-CYS, DS and AI&DS , of the VNRVJIET, Hyderabad during the academic year 2023-2024. Certified further that to the best of our knowledge the work presented in this thesis has not been submitted to any other University or Institute for the award of any Degree or Diploma.

Dr. N. Sunanda
Assistant Professor
Dept. of CSE- (CYS, DS) and AI&DS
VNR VJIET

Dr.M.Raja Sekar
Professor and Head
Dept. of CSE- (CYS, DS) and AI&DS
VNR VJIET

VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

An Autonomous Institute, NAAC Accredited with 'A++' Grade NBA Accredited for CE, EEE, ME, ECE, CSE, EIE, IT, AME B. Tech Courses Approved by AICTE, New Delhi, Affiliated to JNTUH Recognized as "College with Potential for Excellence" by UGC ISO 9001:2015 Certified, QS I GUAGE Diamond Rated

Vignana Jyothi Nagar, Pragathi Nagar, Nizampet (S.O), Hyderabad – 500 090, TS, India

DEPARTMENT OF CSE- (CYS, DS) and AI&DS

DECLARATION

We declare that the major project work entitled "**Health Insurance Premium Detector & analysis of relation between parameters"** submitted in the department of **CSE-(CYS, DS) and AI&DS**, Vallurupalli Nageswara Rao Vignana Jyothi Institute of Engineering and Technology, Hyderabad, in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology** in **CSE-(CYS, DS) and AI&DS** is a bonafide record of our own work carried out under the supervision of Dr. N . Sunanda, **Assistant Professor, Department of CSE- (CYS, DS) and AI&DS, VNRVJIET**. Also, we declare that the matter embodied in this thesis has not been submitted by us in full or in any part thereof for the award of any degree/diploma of any other institution or university previously.

Place: Hyderabad

R. Vishnu Priya	**V. Sindhu Bhargavi**	**Y. Sindhu Priya**
(21071A6744)	(21071A6762)	(21071A6763)

ACKNOWLEDGEMENT

Firstly, we would like to express our immense gratitude towards our institution VNR Vignana Jyothi Institute of Engineering and Technology, which created a great platform to attain profound technical skills in the field of Computer Science, thereby fulfilling our most cherished goal.

We are very much thankful to our Principal, **Dr. Challa Dhanunjaya Naidu,** and our Head of Department, **Dr. M.Raja Sekar**, for extending their cooperation in doing this project within the stipulated time.

We extend our heartfelt thanks to our guide, Dr.N.Sunanda **,** and the project coordinators **Mr.P.Veeranjaneyulu and Dr. T.Preethi** for their enthusiastic guidance throughout the course of our project.

Last but not the least, our appreciable obligation also goes to all the staff members of the Computer Science & Engineering department and to our fellow classmates who directly or indirectly helped us.

Miss.R. Vishnu Priya (21071A6744)

Miss. V. Sindhu Bhargavi (21071A6762)

Miss. Y. Sindhu Priya (21071A6763)

ABSTRACT

A policy that helps to cover all losses or lessen the loss in terms of costs brought on various hazards is insurance. One element of computational intelligence called ***machine learning (ML)*** has the potential to solve a variety of problems in a variety of systems & applications. The price of insurance is influenced by numerous factors. Predicting medical insurance costs using ***ML approaches*** is still a problem in the healthcare industry that requires investigation and improvement. Using a series of ML algorithms, this project provides a computational intelligence approach for predicting healthcare insurance costs. ***Forecasting*** insurance pricing based on certain characteristics allows insurance providers to attract clients while saving time in developing plans for each individual. ML has the potential to significantly reduce these person efforts in policymaking. This may help businesses improve their ***profitability. For*** this aim, a medical insurance cost dataset is obtained from the ***KAGGLE*** repository, & ML methods are utilized to demonstrate how different ***regression models*** can anticipate insurance prices and compare the ***models' accuracy.***

INDEX

List of Tables

1. INTRODUCTION

A policy that helps to cover all losses or lessen the loss in terms of costs brought on by various hazards is insurance. One element of computational intelligence called machine learning (ML) has the potential to solve a variety of problems in a variety of systems and applications. Machine learning (ML) and soft computing are well-known study fields in computational and applied mathematics. The price of insurance is influenced by numerous factors. The expression of the cost of an insurance policy is influenced by certain considerations of many aspects. Predicting medical insurance costs using ML approaches is still a problem in the healthcare industry that requires investigation & improvement. Using a series of machine learning algorithms, this project provides a computational intelligence approach for predicting healthcare insurance costs.

This reduces human work and resources while increasing the company's profitability. As a result, ML can enhance accuracy. To properly anticipate insurance prices based on people's data, such as age, BMI, smoking status and so on. Furthermore, we will discover the most essential variable determining insurance costs.

2. LITERATURE SURVEY/ EXISTING SYSTEM

2.1 FEASIBILITY STUDY

With the rapid change in technology, the development, and enhancement of hardware and software have substantially lowered developers' and users' costs. Educational information, learning environments, and communication techniques have also changed dramatically. Based on Machine Learning Algorithms and methods, this project serves as the best tool for predicting the health insurance of a person in the present world.

2.1.1 ORGANIZATIONAL FEASIBILITY

The project will serve its best for the organizational feasibility goals. The health insurance detection system project can be useful in many ways for various organizations. It helps the organization to analyze the insurance affordability of the people. Since the project is simple to use, the organization finds it very helpful in the present situation.

2.1.2 ECONOMIC FEASIBILITY

The project is financially feasible because it can be used without any charges. Since the model will be trained before using it, you don't even need an internet connection to use the project model. The software used for building the project is economically feasible too. The software used for building the project is Python, Jupyter notebook, Visual Studio which are handy, free and easy to use.

2.1.3 TECHNICAL FEASIBILITY

The Visual Studio Code and Python Programming Language are needed to be installed in the system to build this project. A minimum of 8GB ram is required to run all this software smoothly. Necessary modules and packages need to be installed to build and run the project. Internet connectivity is needed to download and install the packages. The user can download and install these packages and can use them with ease. Thus, this project is technically feasible.

2.1.4 BEHAVIORAL FEASIBILITY

Behavioral feasibility for health insurance detection involves analyzing individual behaviors and patterns to assess their eligibility and risk for health insurance. This includes examining lifestyle choices, medical history, and adherence to preventive measures. Machine learning algorithms can identify feasible indicators, such as consistent exercise routines, regular health check-ups, and adherence to medication plans, to predict health outcomes. This approach enables insurers to tailor coverage and pricing based on an individual's proactive health behaviors, promoting a more accurate and fair insurance model.

2.2 LITERATURE REVIEW

Kaushik, K. et al.[1] It discusses the integration of artificial intelligence (AI) and machine learning (ML) in predicting health insurance premiums. It highlights the impact of these technologies in streamlining healthcare processes, such as faster claim settlements and personalized insurance policies. The research focuses on training and evaluating an artificial neural network (ANN) regression model for predicting premiums, achieving an accuracy of 92.72%. The text emphasizes the potential of ML to enhance healthcare efficiency and reduce costs. Overall, it underscores the need for further exploration and research in the domain of health insurance premium prediction using machine learning.

Sun, J.J et al.[2] The project focuses on understanding the factors influencing health insurance premiums in the United States by employing predictive analytics and insurer attributes. Four key factors—BMI, smoke status, age, and children—demonstrate significant correlations with health insurance costs. The research employs three regression models and one statistical model for the main question and seven classification models for sub-questions. The comparison reveals that the Random Forest model performs the best with an 80% R-square value, followed by Support Vector Machine with 67% accuracy. The introduction contextualizes the project within the high costs and coverage issues of U.S. health insurance, emphasizing the need for understanding factors impacting costs. Key terms include machine learning models, predictive analytics, insurance premiums, R-square value, and accuracy. The overall goal is to provide meaningful insights for insurance companies to make accurate premium charges.
The project conducts a literature review on predictive model algorithms in American health insurance, aiming to enhance insurers' ability to identify high-risk customers and respond to market changes. It addresses a gap in existing literature and emphasizes the role of machine learning technologies for effective analysis. The study explores value chain restructuring, integration of big data and technology, and the development and evaluation of various predictive models, including Multiple Linear Regression, Random Forest, Support Vector Machine, Naïve Bayes, Decision Tree, Logistic Regression, and K-Nearest Neighbour. Findings highlight the industry's adoption of predictive analytics, with 67% of insurers globally using these tools. The research also reveals an increase in lapse rates from 65.7% to 69.9% between 2010 and 2014, prompting the use of predictive modeling, particularly Generalized Linear Models, to reduce lapse rates. The project underscores the growing role of machine learning in the health insurance sector for more effective risk assessment and policy planning.

Karlsson, M. et al.[3] The study re-evaluates methods for detecting adverse and advantageous selection in insurance contracts based on unused variables in premium calculations. It emphasizes that existing approaches can lead to incorrect conclusions if estimated coefficients are influenced by different population segments, advocating for accounting for parameter heterogeneity. Using simulated data and real-world evidence from the private health insurance market in England, the study highlights the empirical relevance of parameter heterogeneity in assessing the efficiency of insurance markets. The research discusses the limitations of the "positive correlation test" and provides a comprehensive overview of commonly used testing procedures in insurance markets, emphasizing the need for accurate empirical methods to understand individual behavior and market functioning.

Alzoubi. et al. [4] The study aims to predict medical insurance costs in the US based on biological and demographic factors using Machine Learning Regression techniques. Four models, namely Gradient Boosting Regressor, AdaBoostRegressor, Lasso, and Elastic Net Regression, were

applied to a US-based dataset. The analysis revealed that boosting techniques outperformed regularization techniques in terms of maximum R2 and minimized loss. The proposed system seeks to help organizations design more public-oriented medical insurance policies, benefiting users and improving organizational revenue. Suggestions for improving the study include data quality and preprocessing, a comprehensive model evaluation, feature importance analysis, ethical considerations, and providing policy implications and recommendations for future work.

2.3 EXISTING SYSTEM

- Most of insurance data is checked by staff of company i.e. manually
- Existing ML models function as "black boxes" with complex internal workings that are difficult to understand. Particularly in sensitive industries like healthcare, where people may wish to understand the variables driving their insurance forecasts, this lack of openness can be problematic.

2.4 DRAWBACKS OF THE EXISTING SYSTEM

- Analyzing is a time taking process as there are so many records to check.
- Professional staff should have to investigate which will affect the company.
- If wrong analysis is done which misleads people can lead to loss for company

3. SOFTWARE REQUIREMENT ANALYSIS

3.1 INTRODUCTION

Following elicitation, requirement analysis is an important and necessary process. It involves identifying, understanding the health insurance predictions and their parameters relations and documenting the essential features, performance criteria.

3.1.1 DOCUMENT PURPOSE

The goal is to Understand the parameters that influence insurance and its prediction.

3.1.2 DEFINITIONS

Machine Learning

Machine learning is the analysis of computer algorithms that learn and develop on their own based on experience and data. Machine learning (ML) in healthcare are approaches to make people's lives easier by anticipating and diagnosing diseases more swiftly than most medical experts. There is a direct link between the insurer and the policyholder when the distance between an insurance business and the consumer is reduced to zero with the use of technology, especially digital health insurance

Data Preprocessing

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.

- Getting the dataset
- Importing libraries
- Importing datasets
- Finding Missing Data
- Encoding Categorical Data
- Splitting dataset into training and test set
- Feature scaling

One Hot Encoding:

One-hot encoding is used in machine learning as a method to quantify categorical data. In short, this method produces a vector with length equal to the number of categories in the data set.

One hot encoding is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction. In this case each value becomes a new column.

Label Encoder:

Features which define a category are Categorical Variables. E.g. Color (red, blue, green), Gender(Male, Female). Machine learning models expect features to be either floats or integers therefore categorical features like color, gender etc. need to be converted to numerical values. Label encoder converts categorical feature to integers.

K-Fold Cross Validation:

To evaluate the performance of a health insurance prediction model on a dataset, we need to measure how well the predictions made by the model match the observed health insurance data. One commonly used method for doing this is known as k-fold cross-validation, which uses the following approach:

- Randomly divide the health insurance dataset into k groups, or "folds," of roughly equal size.
- Choose one of the folds to be the holdout set. Fit the health insurance prediction model on the remaining k-1 folds. Calculate the test Mean Squared Error (MSE) on the observations in the fold that was held out.
- Repeat this process k times, using a different set each time as the holdout set.
- Calculate the overall test MSE to be the average of the k test MSEs.

This approach helps ensure that the health insurance prediction model is evaluated on diverse subsets of the dataset, providing a more robust assessment of its generalization performance. The use of k-fold cross-validation helps in estimating how well the model is likely to perform on unseen data.

Ordinal Data:

Ordinal data classifies data while introducing an order, or ranking. For instance, measuring economic status using the hierarchy: ‘wealthy’, ‘middle income’ or ‘poor.’ However, there is no clearly defined interval between these categories.

Nominal Data:

Nominal data is the simplest data type. It classifies data purely by labeling or naming values e.g. measuring marital status, hair, or eye color. It has no hierarchy to it.

3.2 SYSTEM ARCHITECTURE

It is a conceptual model that describes the structure, viewpoints, and behavior of a system. A formal description and representation of a system organized in a way that facilitates reasoning about the system's structures and behaviors is known as an architecture description. A system architecture consists of system components and created sub-systems that work together to implement the overall system.The proposed System, System architecture starts with detecting the chances of diseases they might face by using attributes like bmi ,smoking etc features in the database. After detecting it, it checks the percentage of risk . The system first checks attributes then risk and gives an amount of insurance for different people based on risk. After preprocessing , the system extracts the features from it , after calculating risk it gives the range of amount for insurance which people have to pay.

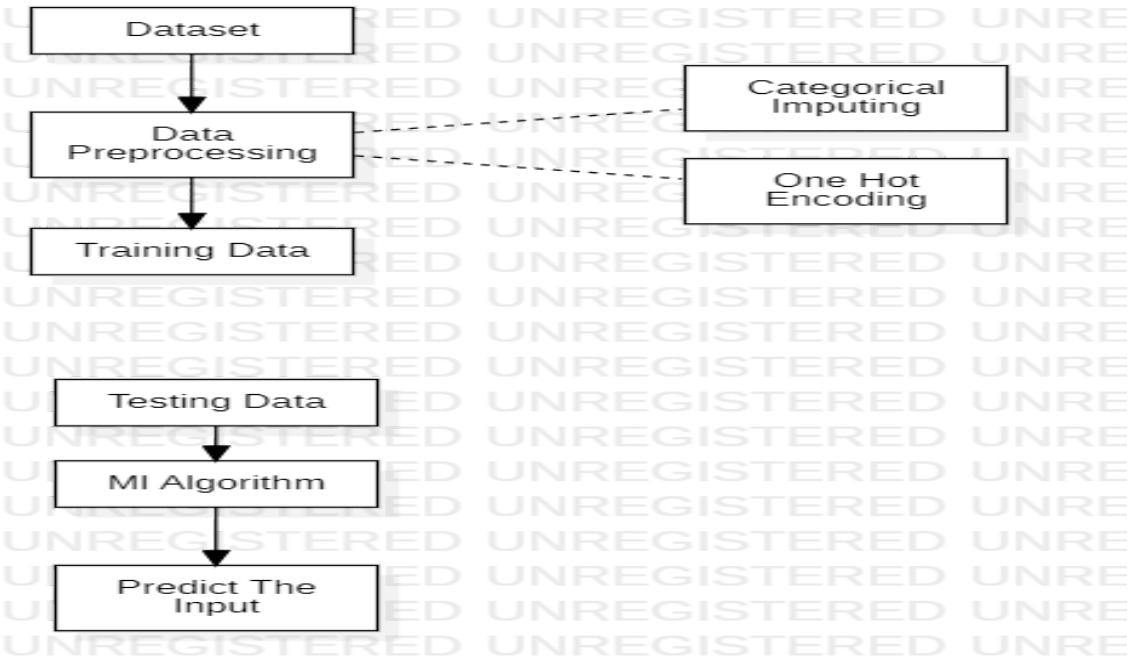


Fig 3.2.1: System Architecture

3.3 FUNCTIONAL REQUIREMENTS

- data preprocessing
- detect fraud cases
- The plot area

3.4 SYSTEM ANALYSIS

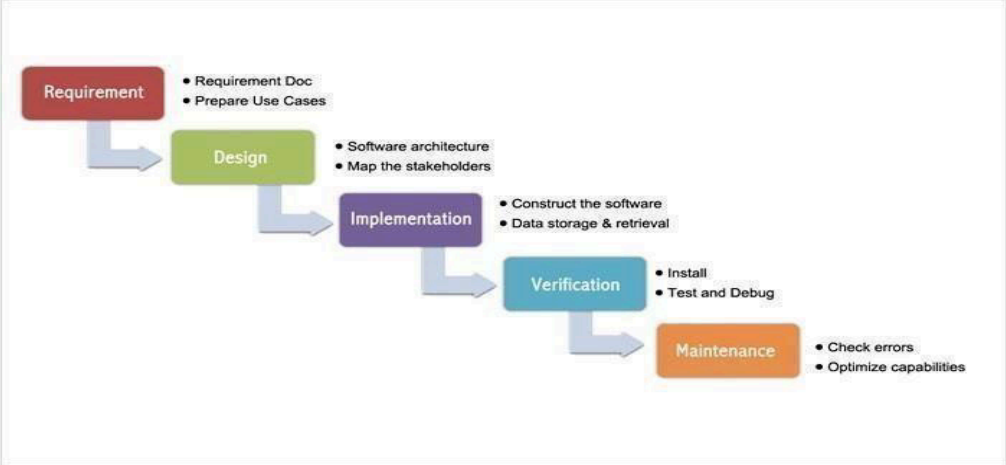


Fig 3.4.1: System Analysis

It was the first Process Model to be introduced. A linear sequential life cycle model is another name for it. It's quite simple to use and understand. Phases do not overlap in this paradigm, and each phase must be finished before the next one begins. The first SDLC approach used during software development was the waterfall model.

The model shows that the development of software is linear and is a sequential process. Only after one phase of the development is completed, we can go to the next phase. In this waterfall paradigm, the phases do not overlap.

The steps in the waterfall model are explained below.

Requirements: The search has become more intense and concentrated on the software's requirements at this time. To comprehend the nature of the programs to be developed, the software engineer must first comprehend the software's information domain, which includes the required functionalities, user interface, and so on. The customer must be informed about the second activity, which must be recorded and presented.

Design: This step is used to transform the above criteria as a representation in the form of "blueprint" software before coding begins. The design must be able to meet the criteria laid out in the previous stage.

Implementation: The design was converted into a machine-readable format in order for it to be interpreted by a computer in some circumstances, i.e., through the coding process into a programming language. This was the stage in which the programmer will put the technical design phase into action.

Verification: It, like anything else constructed, must first be put to the test. The same may be said for software. To ensure that the application is error-free, all functions must be checked, and the results must closely comply to the previously specified requirements.

Maintenance: Software maintenance, including development, is essential since the software that is being generated is not always exactly like that. It may still have minor faults that were not identified previously when it runs, or it may require additional capabilities that were not previously available in the software.

Useful factors: The Waterfall model has its advantages like it is simple to use. Additionally, while using the model all the system requirements can be defined as a whole, explicitly and at the start the product can run without many issues.

It is economic to make changes in the early stages of the project when there are problems with system requirements then when the problems which arise in later stages.

3.3 NON-FUNCTIONAL REQUIREMENTS

Ease of Use

- The system is simple, user friendly.

Extensibility

- The system can be easily extended to incorporate additional functionality.

Security

- The system is secure because it doesn't share any information of the client without his/her consent.

Maintainability

- The system will be as self-contained as possible to allow for ongoing maintenance.

Reliability

- Most efficient model is chosen that is Random Forest or gradient booster with 80% accuracy.

3.6 SOFTWARE REQUIREMENT SPECIFICATION

JUPYTER NOTEBOOK

The Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at Project Jupyter.

Jupyter Notebooks are a spin-off project from the IPython project, which used to have an IPython Notebook project itself. The name, Jupyter, comes from the core supported programming languages that it supports: Julia, Python, and R. Jupyter ships with the IPython kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that you can also use.

GOOGLE COLAB

Colab is a free Jupyter notebook environment that runs entirely in the cloud. Most importantly, it does not require a setup and the notebooks that you create can be simultaneously edited by your team members - just the way you edit documents in Google Docs. Colab supports many popular machine learning libraries which can be easily loaded in your notebook.

- Write and execute code in Python
- Document your code that supports mathematical equations
- Create/Upload/Share notebooks

- Import/Save notebooks from/to Google Drive
- Import/Publish notebooks from GitHub
- Import external datasets e.g. from Kaggle
- Integrate PyTorch, TensorFlow, Keras, OpenCV
- Free Cloud service with free GPU

3.4 SOFTWARE REQUIREMENTS

- Software : jupyter notebook or google colab
- Operating System : Windows family
- Technology : Machine Learning

3.5 HARDWARE REQUIREMENTS

- Minimum 8GB Ram Laptop
- Internet Connection

4. SOFTWARE DESIGN

4.1 UML DIAGRAMS

The Device Architecture Manual describes the application requirements, operating state, application and subsystem functionality, documents and repository setup, input locations, yield types, human-machine interfaces, management reasoning, and external interfaces. The Unified Modeling Language (UML) assists software developers in expressing an analysis model through documents that contain a plethora of syntactic and semantic instructions. A UML context is defined as five distinct viewpoints that present the system from a particularly different point of view.

The components are similar to modules that can be combined in a variety of ways to create a complete UML diagram. As a result, comprehension of the various diagrams is essential for utilizing the knowledge in real-world systems. The best method to understand any complex system is to draw diagrams or images of it. These designs have a bigger influence on our understanding. Looking around, we can see that info-graphics are not a new concept, but they are frequently utilized in a variety of businesses in various ways.

User Model View

The perspective refers to the system from the clients' point of view. The exam's depiction depicts a situation of utilization from the perspective of end-clients. The user view provides a window into the system from the perspective of the user, with the system's operation defined in light of the user and what the user wants from it.

Structural model view

This layout represents the details and functionality of the device. This software design maps out the static structures. This view includes activity diagrams, sequence diagrams and state machine diagrams

Behavioral Model View

It refers to the social dynamics as framework components, delineating the assortment cooperation between various auxiliary components depicted in the client model and basic model view. UML Behavioral Diagrams illustrate time-dependent aspects of a system and communicate the system's dynamics and how they interact. Behavioral diagrams include interaction diagrams, use case diagrams, activity diagrams and state–chart diagrams.

Implementation Model View

The essential and actions as frame pieces are discussed in this when they are to be manufactured. This is also referred to as the implementation view. It uses the UML Component diagram to describe system components. One of the UML diagrams used to illustrate the development view is the Package diagram.

Environmental Model View

The systemic and functional component of the world where the program is to be introduced was expressed within this. The diagram in the environmental view explains the software model's after-deployment behavior. This diagram typically explains user interactions and the effects of software on the system. The following diagrams are included in the environmental model: Diagram of deployment.

The UML model is made up of two separate domains:

- Demonstration of UML Analysis, with a focus on the client model and auxiliary model perspectives on the framework.
- UML configuration presenting, which focuses on demonstrations, usage, and natural model perspectives.

4.1.1 USE CASE DIAGRAM

The objective of a use case diagram is to portray the dynamic nature of a system. However, because the aim of the other four pictures is the same, this description is too broad to characterize the purpose. We'll look into a specific purpose that distinguishes it from the other four diagrams.

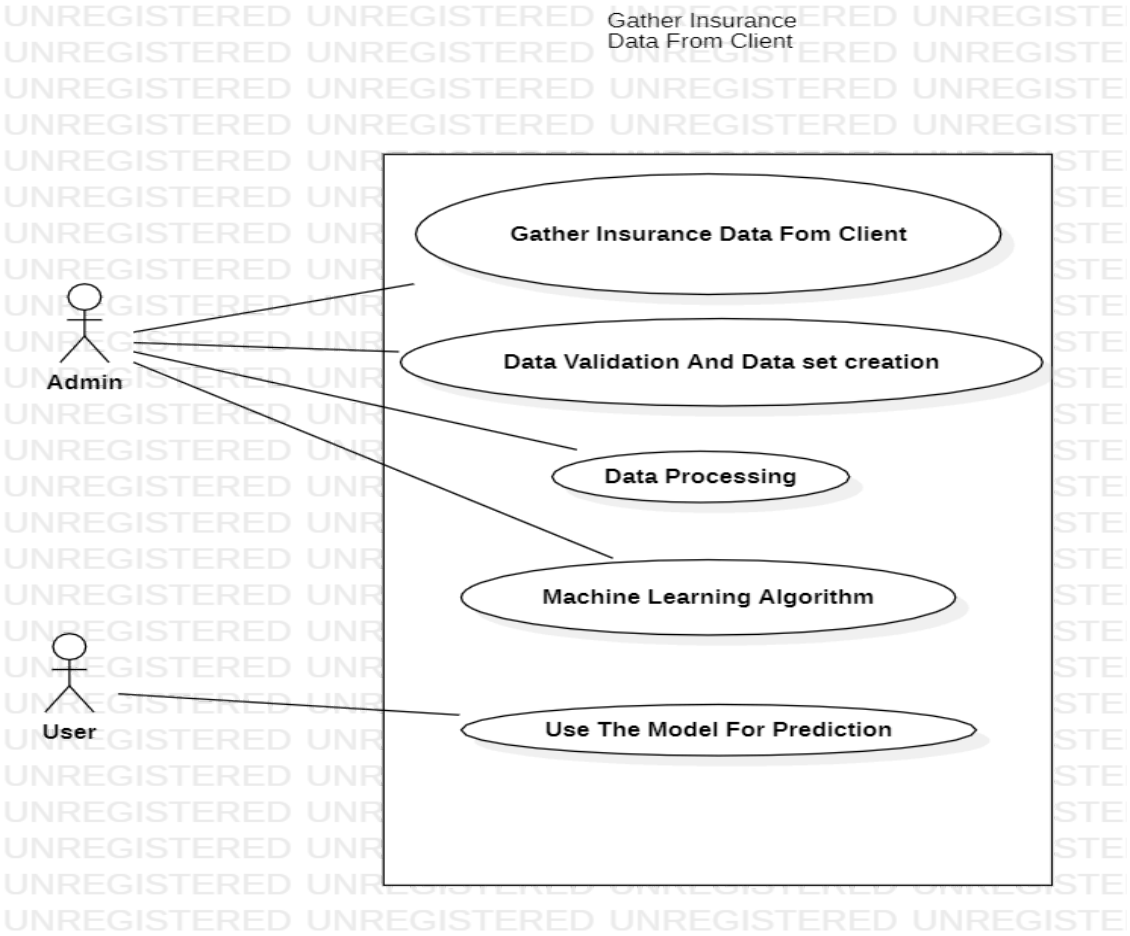


Fig 4.1.1.1: Use Case diagram for the application

Actors:

- Admin
- User

Use Cases:

- Gather Insurance Data From Client
- Data Validation And Data Set Creation
- Data Processing
- Machine Learning Algorithm
- Use The Model For Prediction

Connections:

- Admin must gather data required to choose which algorithm to be used
- Then the admin will be processing the data and choose the algorithm based on accuracy
- User will be using the efficient model for prediction on the data

4.1.2 SEQUENCE DIAGRAM

Because it illustrates how a group of items interact with one another, a sequence diagram is a form of interaction diagram. These diagrams are used by software engineers and business people to comprehend the requirements for a new system or to document a current process. Sequence diagrams are sometimes known as event diagrams or event scenarios. Sequence diagrams can be useful as a reference for businesses and other organizations. Make the diagram to show:

- Describe the specifics of a UML use case.
- Create a model of the logic of a complex procedure, function, or operation.
- Examine how objects and components interact with one another in order to complete a process.
- Plan and comprehend the specific functionality of a current or future scenario.

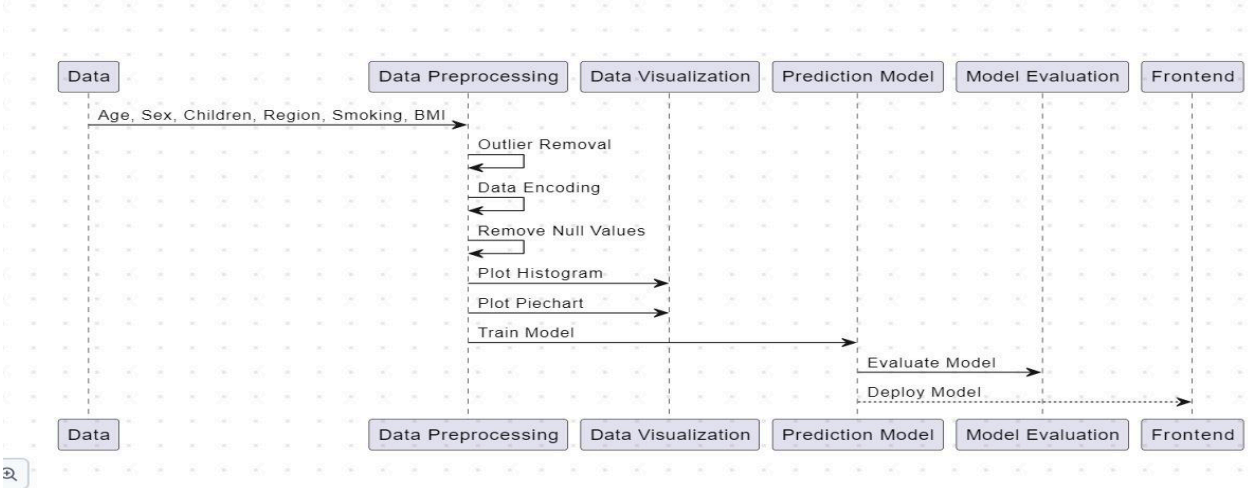


Fig 4.1.2.1: Sequence Diagram

In the above sequence diagram, the lifelines are:

- Client
- Data-Preprocessing
- Data Visualization
- Prediction Model
- Model Evaluation
- Frontend

The sequence starts from the client sending the data required for the prediction process that includes data-validation, data-preprocessing.

4.1.1 ACTIVITY DIAGRAM

An activity diagram is a flowchart that displays the movement of information from one action to the next. A system operation can be used to describe the activity.

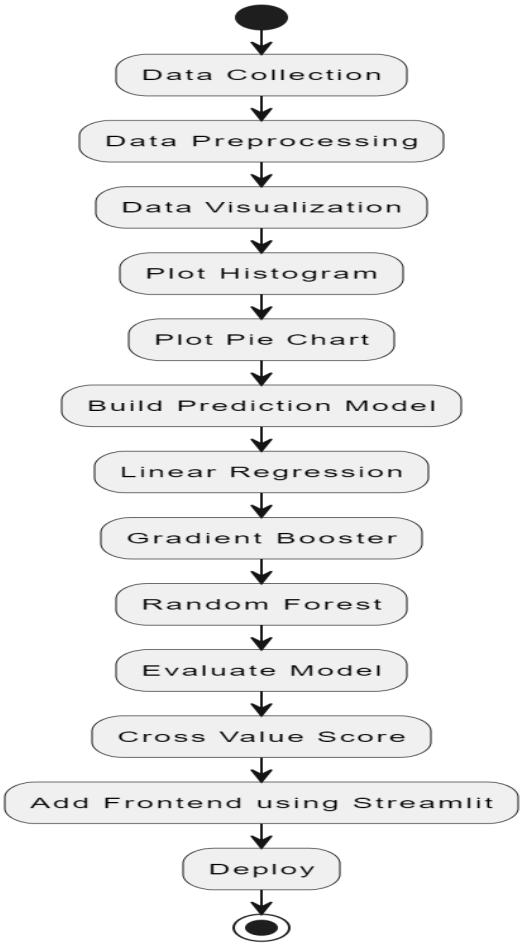


Fig 4.1.3.1: Activity Diagram

This activity diagram shows the whole activity of the system. The Activity starts with the client submitting the data required for prediction and data validation, data preprocessing followed by predicting the results.

4.1.2 CLASS DIAGRAM

A static diagram is also referred to as a class diagram. It depicts the static view of an application. A class diagram can be used to visualize, describe, and document various parts of a system, as well as to create executable code for a software programmer.

The traits and activities of a class, as well as the constraints, are described in a class diagram.

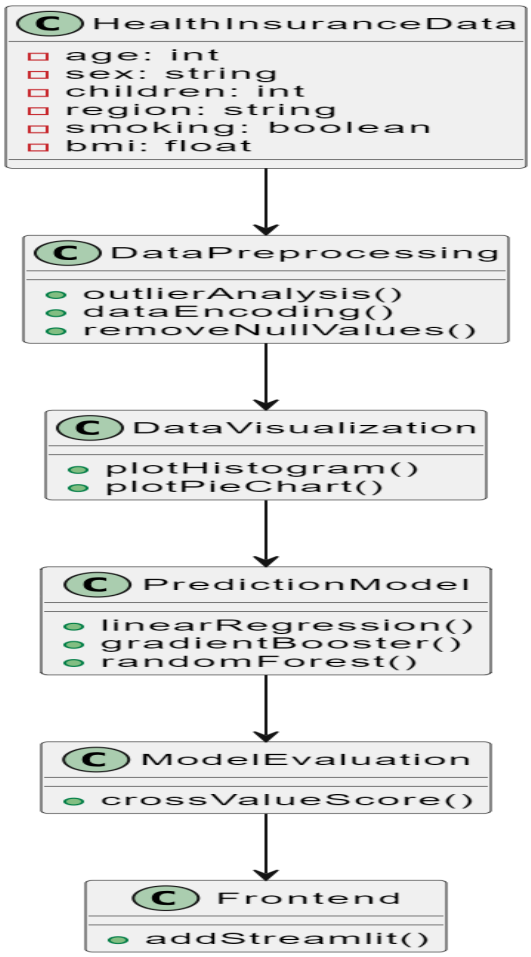
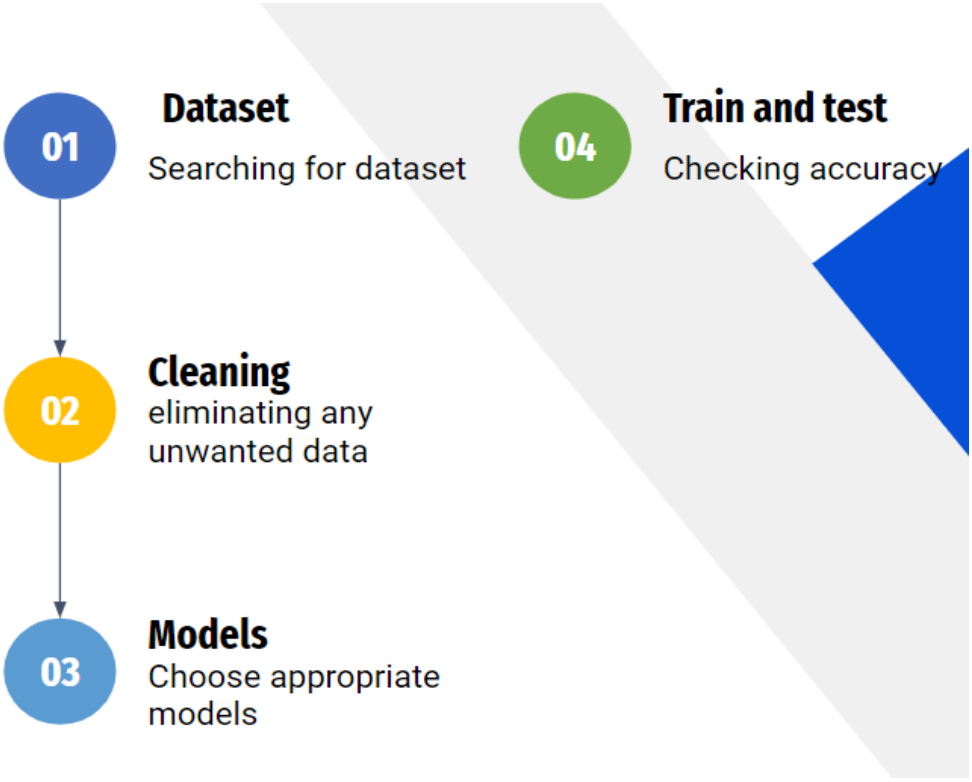


Fig 4.1.4.1: Class Diagram

The main class in this class diagram is about health insurance data which places a major role in predicting the insurance amount for people. The other classes are about data validation and model prediction which suits best and after model the front-end is deployed for the user.

5 PROPOSED SYSTEM

5.1 METHODOLOGY



Data Set:
Data is crucial in finding patterns and using that pattern to predict the outcome in our case . For our project we give a CSV file to train the model.
Data set consists of 1338 rows and 7 columns

Cleaning:

Data cleaning is the process of preparing data for analysis by weeding out information that is irrelevant or incorrect.

This is generally data that can have a negative impact on the model or algorithm it is fed into by reinforcing a wrong notion.

Data cleaning not only refers to removing chunks of unnecessary data, but it's also often associated with fixing incorrect information within the train-validation-test dataset and reducing duplicates

Here are some key takeaways on the best practices you can employ for data cleaning:

1. Identify and drop duplicates and redundant data
2. Detect and remove inconsistencies in data by validating with known factors
3. Maintain a strict data quality measure while importing new data.
4. Fix typos and fill in missing regions with efficient and accurate algorithms

Models:

A machine learning model is defined as a mathematical representation of the output of th*e training process*. Machine learning is the study of different algorithms that can improve automatically through experience & old data and build the model. A machine learning model is similar to computer software designed to recognize patterns or behaviors based on previous experience or data. The learning algorithm discovers patterns within the training data, and it outputs an ML model which captures these patterns and makes predictions on new data.

In Our Project we compared different machine learning models like

- Linear Regression
- Gradient Booster
- Random Forest

Train and Test:
Machine Learning is one of the booming technologies across the world that enables computers/machines to turn a huge amount of data into predictions. However, these predictions highly depend on the quality of the data, and if we are not using the right data for our model, then it will not generate the expected result. In machine learning projects, we generally divide the original dataset into training data and test data. We train our model over a subset of the original dataset, i.e., the training dataset, and then evaluate whether it can generalize well to the new or unseen dataset or test set. Therefore, train and test datasets are the two key concepts of machine learning, where the training dataset is used to fit the model, and the test dataset is used to evaluate the model.

5.2 FUNCTIONALITIES

5.2.1 Importing Data:
- we import CSV file
- file contain data sent by client

5.2.2 Preprocessing data:
- Removing inconsistencies in data like null values and outliers
- Dealing with categorical data

5.2.3 Choosing Model:
- Test each model and choose best model according to accuracy

5.2.4 Analysis:
- Data which is displayed can be summarized.

5.3 ADVANTAGES OF PROPOSED SYSTEM:

- Easy to maintain large datasets.
- Takes less time as compared to traditional methods
- Less human effort

6. CODING AND IMPLEMENTATIONS

6.1 DATASET:

It is used to train the model. The model learns from this data set and gives the outputs based on the learning during the testing.
We have data consisting of 1338 rows and 7 columns, each column has its own significance to find patterns in data. Each row consists of person who are insured

6.2 Understanding Data

Initially we imported all libraries which are required like pandas ,matplotlib seaborn etc. Data set is in the CSV format. Data consists of both categorical and numerical data . Some columns have null values which have to be preprocessed.

Data types we have in data are:

int64 data
object data
floatdata

```
1]: import pandas as pd
    data=pd.read_csv('insurance.csv')
    data
```

1]:

	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
...	...	...	...	...	...	...	...
1333	50	male	30.970	3	no	northwest	10600.54830
1334	18	female	31.920	0	no	northeast	2205.98080
1335	18	female	36.850	0	no	southeast	1629.83350
1336	21	female	25.800	0	no	southwest	2007.94500
1337	61	female	29.070	0	yes	northwest	29141.36030

1338 rows × 7 columns

```
copydata.dtypes
```

```
age            int64
sex           object
bmi          float64
children       int64
smoker        object
region        object
charges      float64
dtype: object
```

The above figure shows the datatypes available in the dataset .The original dataset contains data types of integer,object,float etc.

```
In [77]: copydata.columns
```

```
Out[77]: Index(['age', 'sex', 'bmi', 'children', 'smoker', 'region', 'charges'], dtype='object')
```

```
copydata.shape
```

```
(1338, 7)
```

Data set consists of 1338 rows and 7 columns,and it contains different attributes like age,sex,bmi,children,smoker,region,charges.

```
copydata.nunique()
```

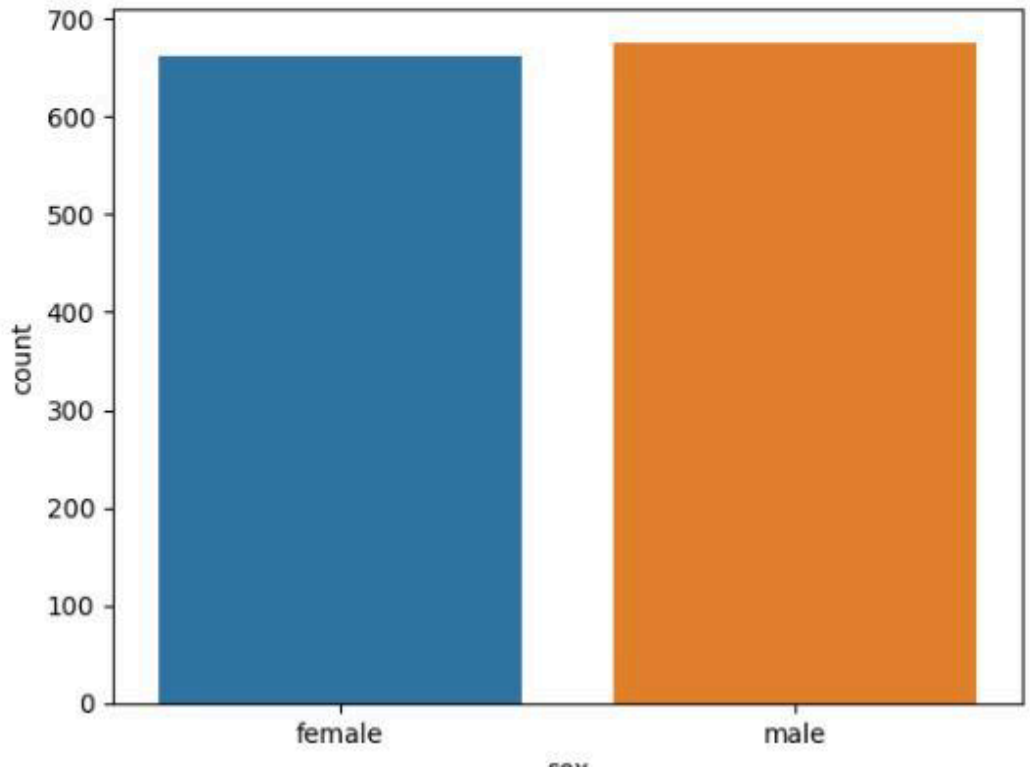
```
age            47
sex             2
bmi           548
children        6
smoker          2
region          4
charges      1337
dtype: int64
```

nunique functions help us to find unique values present in the single column. Some columns have a high number of unique elements and some have less unique elements. Seeing the number of unique numbers is important to see whether many unique elements are needed for prediction . And we can see no column has a number of unique elements equal to 1 which means no need to delete or remove any feature because if a column has only one value then it contributes nothing to the model prediction .

In our data set their are 676 male and 662 female are there, with which we can train our model .

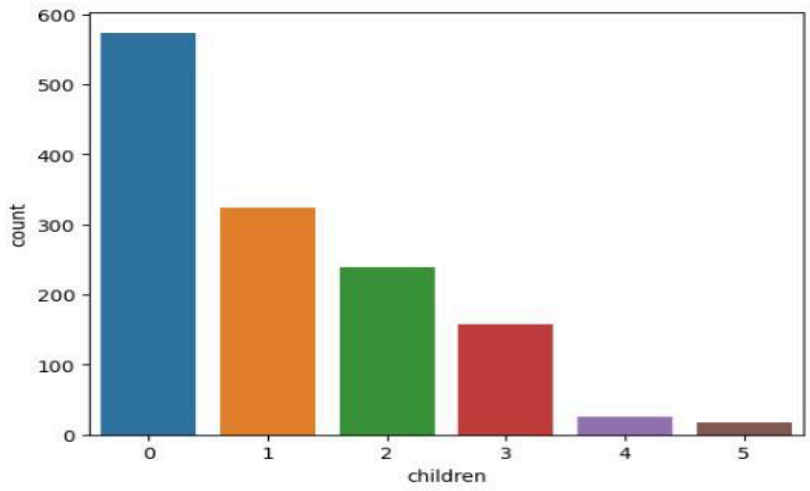
```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
sns.countplot(x=data['sex'])
```

```
<Axes: xlabel='sex', ylabel='count'>
```



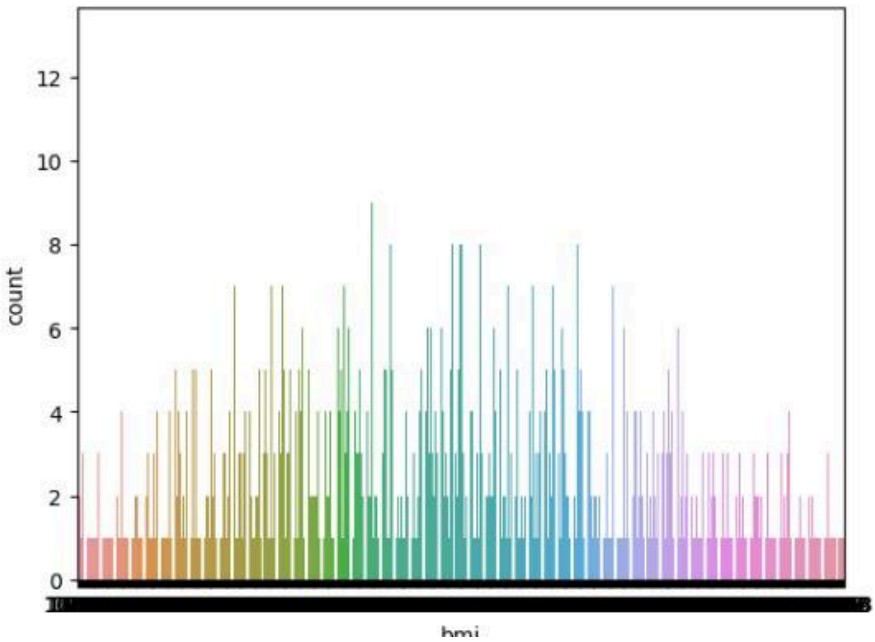
In the below diagram we have most of the people with 0 or 1, 2 children very few people have 3 children and negligible people have 4 or 5 children.

```
In [9]: sns.countplot(x=data['children'])
Out[9]: <Axes: xlabel='children', ylabel='count'>
```



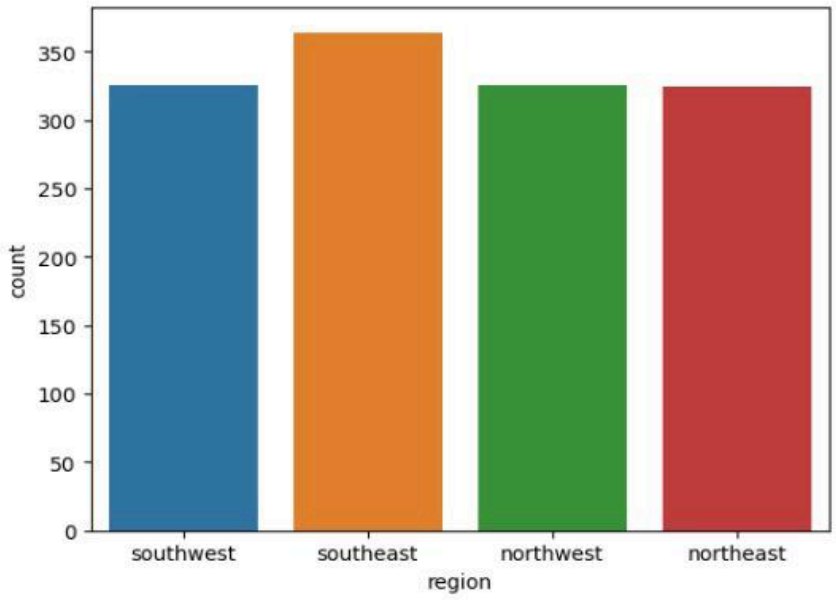
Below plot tells us about the counts of different values of bmi of the people here. The plot says that bmi values are numerical and hugely affects the accuracy of the model .So bmi is the most considerable factor for proper accuracy of the model.

```
In [10]: sns.countplot(x=data['bmi'])
Out[10]: <Axes: xlabel='bmi', ylabel='count'>
```



Below plot is about the attribute region which contains 4 regions namely : southeast,southwest,northeast,northwest.
The highest number are from the southeast region.

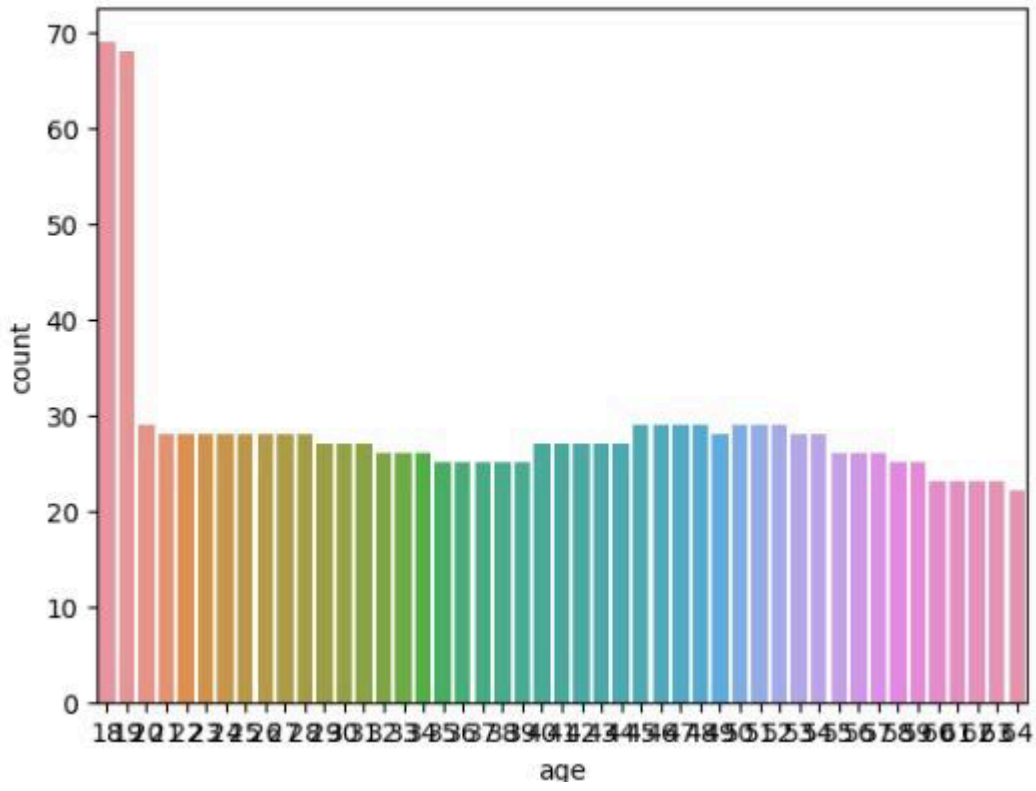
```
In [11]: sns.countplot(x=data['region'])
Out[11]: <Axes: xlabel='region', ylabel='count'>
```



The plot is about one of numerical parameter age; it is a continuous parameter so it also affects the model rather than other categorical variables it affects the most.

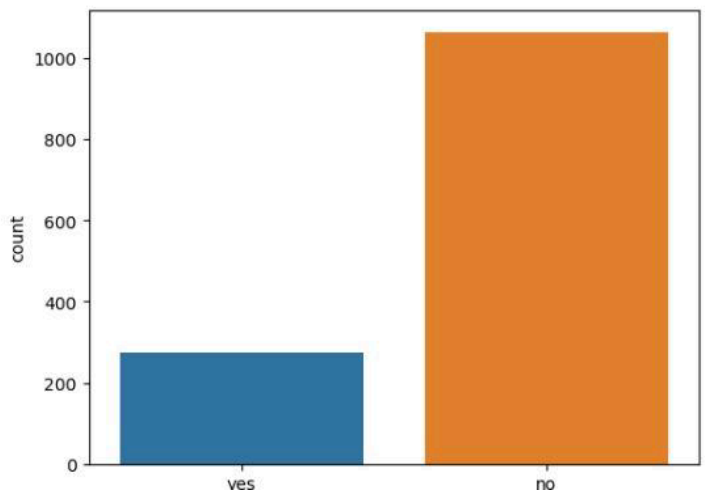
```
sns.countplot(x=data['age'])
```

```
<Axes: xlabel='age', ylabel='count'>
```



The below bar graph shows no of non smokers and smokers in the dataset there are about 300 smokers and 1000 nonsmokers in the dataset.

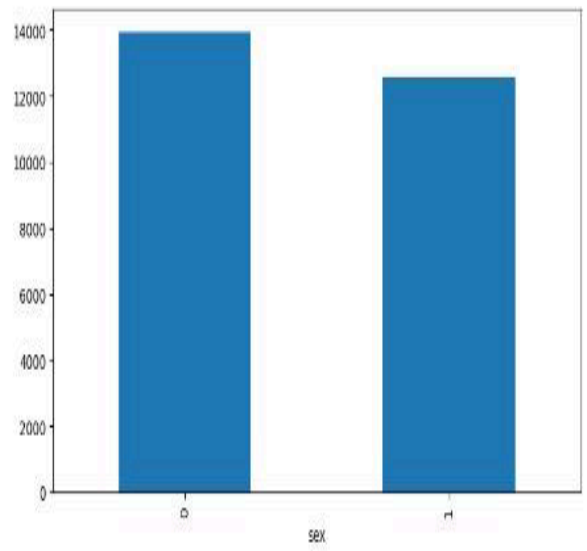
```
]: sns.countplot(x=data['smoker'])
]: <Axes: xlabel='smoker', ylabel='count'>
```

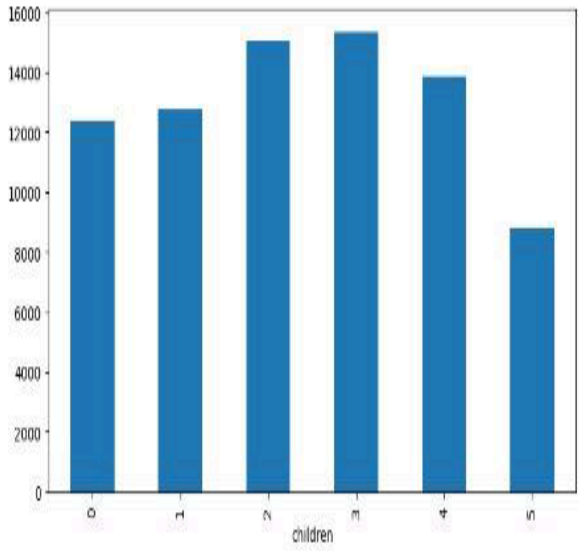


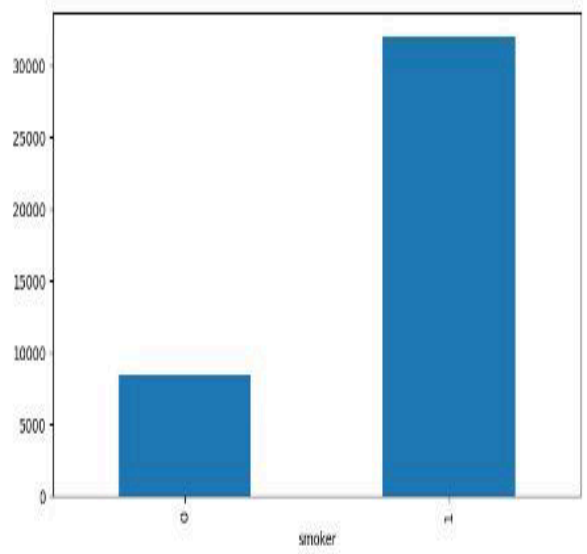
The below subplots tells about the bivariate analysis between output charges and input parameters.To know how each input affects the charges below analysis is essential.

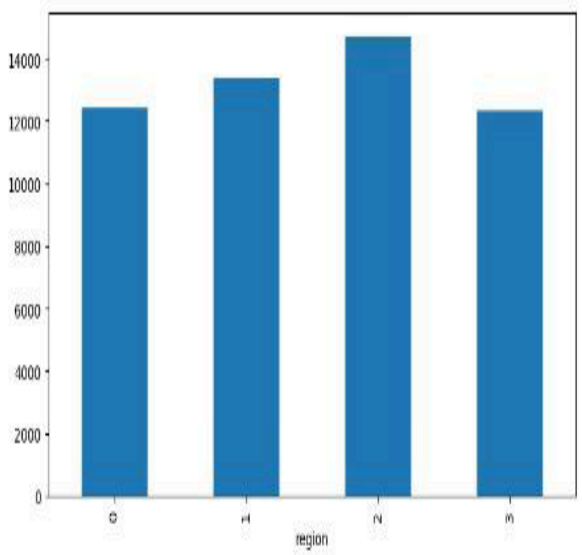
```
#bivaraite analysis between output charges and input parameters
features = ['sex', 'children', 'smoker', 'region']

plt.subplots(figsize=(20, 10))
for i, col in enumerate(features):
    plt.subplot(2, 2, i + 1)
    copydata.groupby(col).mean()['charges'].plot.bar()
plt.show()
```

```
C:\Users\sindhubhargavi\AppData\Local\Temp\ipykernel_29944\4160958757.py:6: MatplotlibDeprecationWarning: Auto-removal of overl
apping axes is deprecated since 3.6 and will be removed two minor releases later; explicitly call ax.remove() as needed.
  plt.subplot(2, 2, i + 1)
```





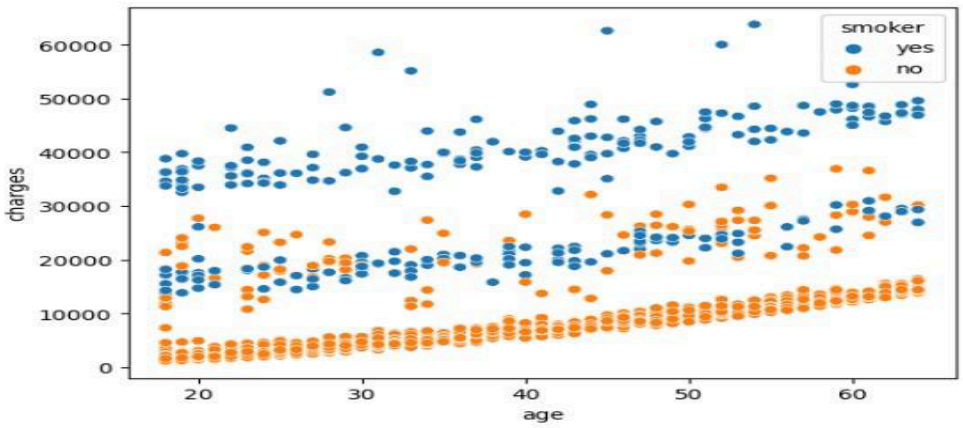




This plot tells about the variation of charges with respect to age in consideration of smoker or not.

```
sns.scatterplot(x=data['age'],y=data['charges'],hue=data['smoker'])
```

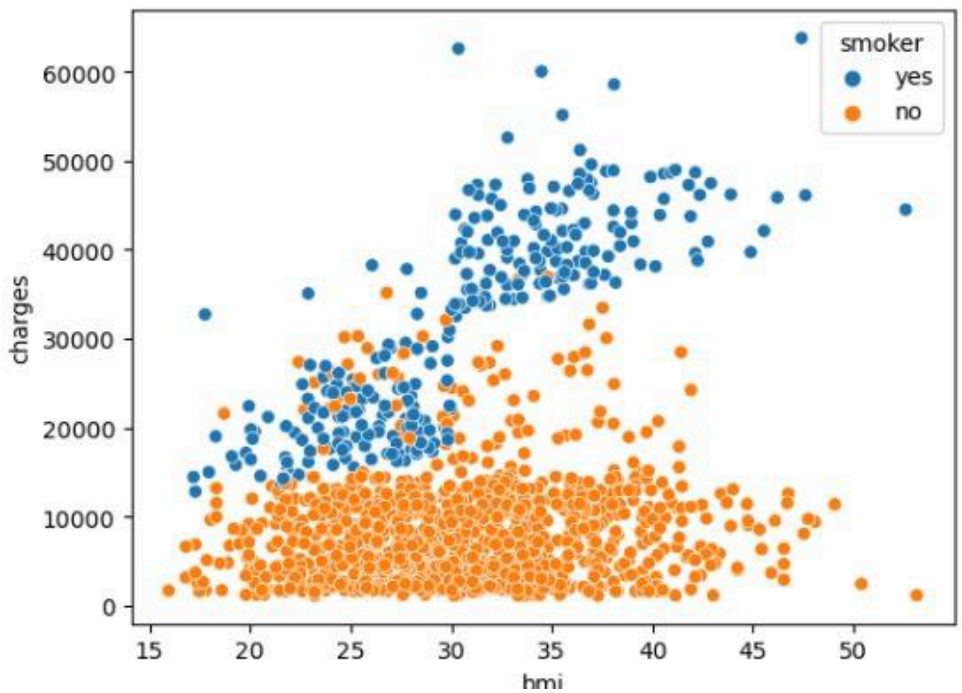
```
<Axes: xlabel='age', ylabel='charges'>
```



This plot tells about the variation of charges with respect to bmi in consideration of whether a smoker or not.

```
sns.scatterplot(x=data['bmi'],y=data['charges'],hue=data['smoker'])
```

```
<Axes: xlabel='bmi', ylabel='charges'>
```



6.3 Data Preprocessing:

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.When creating a machine learning project, it is not always the case that we come across clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put it in a formatted way. So for this, we use data preprocessing tasks.

Some preprocessing tasks we are going to do are:

1. removing unwanted features
2. replacing a feature with more meaning full feature
3. removing or replacing null(?) values
4. Dealing with Categorical data
5. Dealing with outliers

```
.3]: copydata.describe()
```

.3]:

	age	sex	bmi	children	smoker	region	charges
count	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	0.494768	30.663397	1.094918	0.204783	1.514948	13270.422265
std	14.049960	0.500160	6.098187	1.205493	0.403694	1.105572	12110.011237
min	18.000000	0.000000	15.960000	0.000000	0.000000	0.000000	1121.873900
25%	27.000000	0.000000	26.296250	0.000000	0.000000	1.000000	4740.287150
50%	39.000000	0.000000	30.400000	1.000000	0.000000	2.000000	9382.033000
75%	51.000000	1.000000	34.693750	2.000000	0.000000	2.000000	16639.912515
max	64.000000	1.000000	53.130000	5.000000	1.000000	3.000000	63770.428010

Here we are using the ordinal encoding to convert categorical data to numerical data.In the data set sex,smoker and region columns are categorical data they need to convert numerical data for model fitting and evaluation of model .So it is needed to convert categorical data to numerical data.

```
]: copydata['sex']=copydata['sex'].map({'male':0,'female':1})
   copydata['smoker']=copydata['smoker'].map({'yes':1,'no':0})
   copydata['region']=copydata['region'].map({'northwest':0, 'northeast':1,'southeast':2,'southwest':3})
   copydata
```

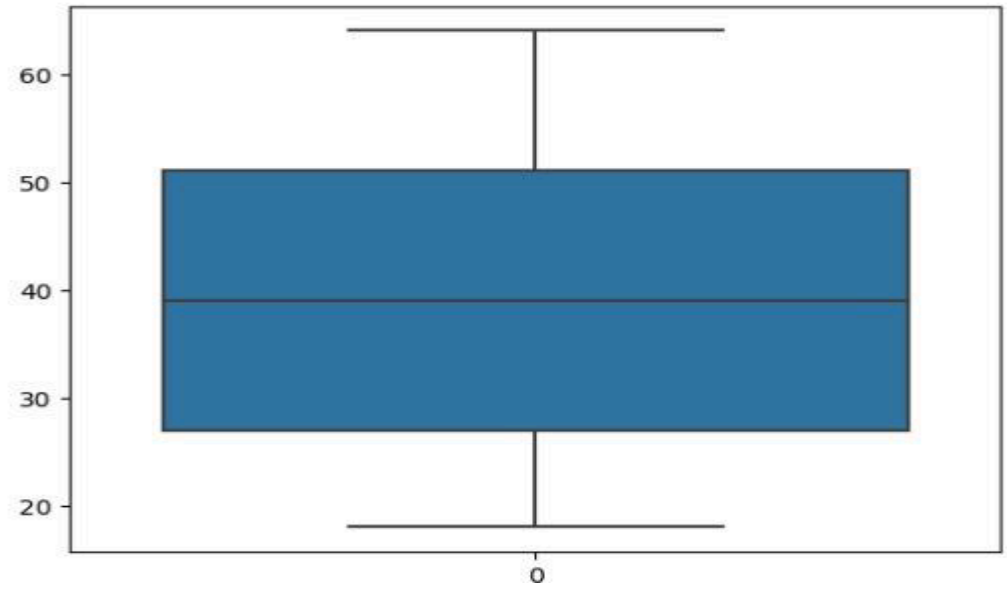
]:

	age	sex	bmi	children	smoker	region	charges
0	19	1	27.900	0	1	3	16884.92400
1	18	0	33.770	1	0	2	1725.55230
2	28	0	33.000	3	0	2	4449.46200
3	33	0	22.705	0	0	0	21984.47061
4	32	0	28.880	0	0	0	3866.85520
...	...	...	...	...	...	...	...
1333	50	0	30.970	3	0	0	10600.54830
1334	18	1	31.920	0	0	1	2205.98080
1335	18	1	36.850	0	0	2	1629.83350
1336	21	1	25.800	0	0	3	2007.94500
1337	61	1	29.070	0	1	0	29141.36030

1338 rows × 7 columns

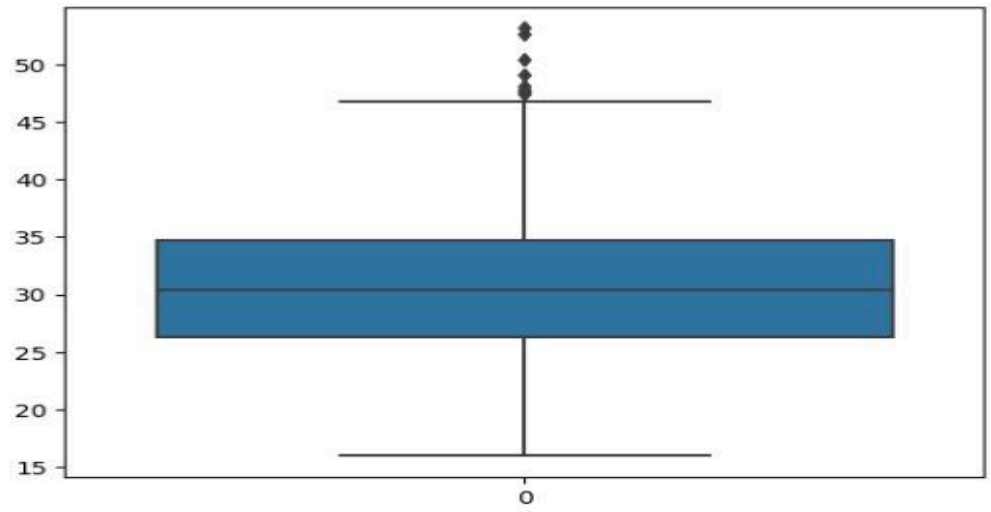
By plotting the box whisker plot against continuous columns like bmi,age is to be plotted.
The below diagram is box whisker plot of age is plotted it shows there are no outliers in the age column.

```
: sns.boxplot(copydata['age'])
: <Axes: >
```



The below plot is a box whisker plot against bmi here there is some outliers in the bmi column.

```
sns.boxplot(copydata['bmi'])
<Axes: >
```



The above diagram represents there are outliers in bmi column so to remove outliers in the bmi column the method used is finding Inter quartile region and then finding the minimum and maximum value by formula Q1-1.5*IQR and Q3+1.5*IQR and after finding the minimum and maximum values remove the values which are above maximum and below minimum and copy values other than that in another dataframe.

```
In [29]: percentile25=copydata['bmi'].quantile(0.25)
         percentile75=copydata['bmi'].quantile(0.75)

In [30]: percentile25
Out[30]: 26.29625

In [31]: percentile75
Out[31]: 34.69375

In [32]: iqr=percentile75-percentile25
         iqr
Out[32]: 8.3975

In [33]: lower=percentile25-1.5*iqr
         lower
Out[33]: 13.7

In [34]: higher=percentile75+iqr*1.5
         higher
Out[34]: 47.290000000000006

In [35]: copydata[copydata['bmi']<lower]
Out[35]:
```

	age	sex	bmi	children	smoker	region	charges

```
]: copydata[copydata['bmi']>higher]
]:
```

	age	sex	bmi	children	smoker	region	charges
116	58	0	49.06	0	0	2	11381.32540
286	46	1	48.07	2	0	1	9432.92530
401	47	0	47.52	1	0	2	8083.91980
543	54	1	47.41	0	1	2	63770.42801
847	23	0	50.38	1	0	2	2438.05520
860	37	1	47.60	2	1	3	46113.51100
1047	22	0	52.58	1	1	2	44501.39820
1088	52	0	47.74	1	0	2	9748.91060
1317	18	0	53.13	0	0	2	1163.46270

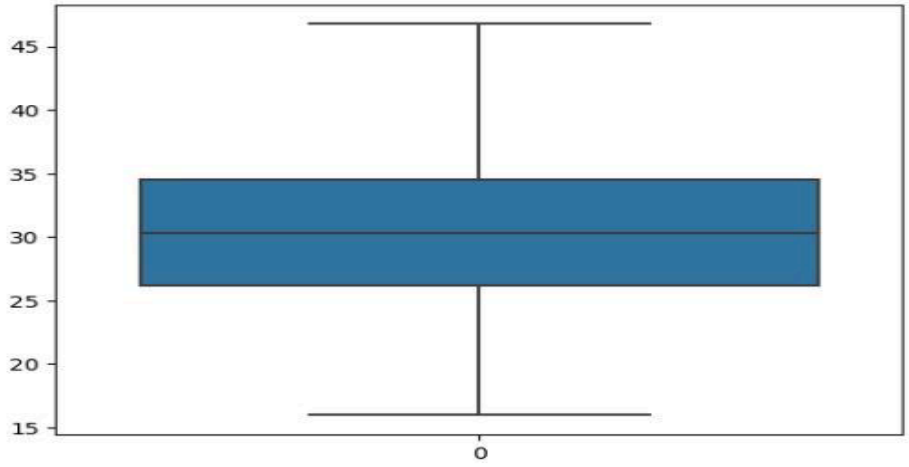
```
]: newcopydata=copydata[copydata['bmi']<higher]
   newcopydata
]:
```

	age	sex	bmi	children	smoker	region	charges
0	19	1	27.900	0	1	3	16884.92400
1	18	0	33.770	1	0	2	1725.55230
2	28	0	33.000	3	0	2	4449.46200
3	33	0	22.705	0	0	0	21984.47061
4	32	0	28.880	0	0	0	3866.85520
...	...	...	...	...	...	...	...
1333	50	0	30.970	3	0	0	10600.54830
1334	18	1	31.920	0	0	1	2205.98080
1335	18	1	36.850	0	0	2	1629.83350
1336	21	1	25.800	0	0	3	2007.94500
1337	61	1	29.070	0	1	0	29141.36030

1329 rows × 7 columns

After removal of outliers from the bmi column again there is need of plotting box whisker to check if the outliers are removed from the data in the new dataframe.

```
[38]: newcopydata.shape
t[38]: (1329, 7)

[39]: import seaborn as sns
      sns.boxplot(newcopydata['bmi'])
t[39]: <Axes: >
```




```
[40]: #check the skewness of the data
      newcopydata['bmi'].skew()
t[40]: 0.15717963249230826

[41]: newcopydata['age'].skew()
t[41]: 0.058413301736796895
```

To know about how each variable is related to another variable and how strong it is related corr() function is used on dataframe.and heatmap is plotted to represent correlation between variables pictorially.

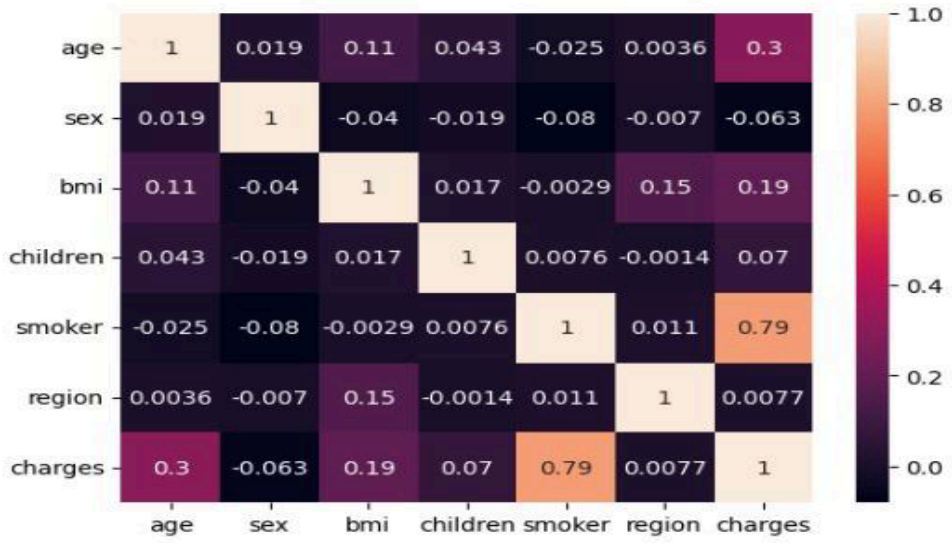
```
[43]: newcopydata.corr()
```

[43]:

	age	sex	bmi	children	smoker	region	charges
age	1.000000	0.019143	0.114686	0.043041	-0.024505	0.003597	0.302022
sex	0.019143	1.000000	-0.039958	-0.019281	-0.079854	-0.007047	-0.063394
bmi	0.114686	-0.039958	1.000000	0.017355	-0.002871	0.152816	0.193196
children	0.043041	-0.019281	0.017355	1.000000	0.007577	-0.001403	0.069739
smoker	-0.024505	-0.079854	-0.002871	0.007577	1.000000	0.010690	0.785912
region	0.003597	-0.007047	0.152816	-0.001403	0.010690	1.000000	0.007693
charges	0.302022	-0.063394	0.193196	0.069739	0.785912	0.007693	1.000000

```
sns.heatmap(data=newcopydata.corr(),annot=True)
```

```
<Axes: >
```



6.4 Choosing model

To predict the insurance charges different models are trained and tested.
The below figure shows about linear regression model training and testing here we divided 20% of data for testing and 80% data for training purpose.

```
#linear regression model
from sklearn.model_selection import train_test_split,cross_val_score
x=newcopydata.drop(['charges'],axis=1)
y=newcopydata[['charges']]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

```
x_train
```

	age	sex	bmi	children	smoker	region
759	18	0	38.170	0	1	2
1052	49	0	29.830	1	0	1
312	43	0	35.970	3	1	2
991	38	1	27.835	2	0	1
582	39	0	45.430	2	0	2
...	...	...	...	...	...	...
733	48	1	27.265	1	0	1
1293	46	0	25.745	3	0	0
906	27	0	32.585	3	0	1
1022	47	0	36.080	1	1	2
253	27	0	30.300	3	0	3

1063 rows × 6 columns

```
834    5377.45780

266 rows × 1 columns
```

```
In [50]: from sklearn.linear_model import LinearRegression
         from sklearn.metrics import mean_squared_error,r2_score
         reg=LinearRegression()
         reg.fit(x_train,y_train)

Out[50]: LinearRegression()

In [51]: y_pred=reg.predict(x_test)

In [52]: y_pred
```

The below figure is about Random forest model training and testing.To implement random forest model import random forest regressor from sklearn package,and no of estimators should be fixed for perfect analysis.

```
]: from sklearn.ensemble import RandomForestRegressor
   from sklearn.model_selection import train_test_split
   from sklearn.metrics import mean_squared_error ,r2_score
    X = newcopydata.drop(['charges'], axis=1)
   Y= newcopydata['charges']

']: from sklearn.ensemble import RandomForestRegressor
   xtrain,xtest,ytrain,ytest=train_test_split(X,Y,test_size=0.2,random_state=42)

]: xtest

]:
        age  sex     bmi  children  smoker  region

: rf_regressor = RandomForestRegressor(n_estimators=100, random_state=42)

: rf_regressor.fit(xtrain, ytrain)

:       RandomForestRegressor
   RandomForestRegressor(random_state=42)

:
   y_pred = rf_regressor.predict(xtest)
   y_pred
```

```
from sklearn.ensemble import GradientBoostingRegressor
X = newcopydata.drop(['charges'], axis=1)
Y= newcopydata['charges']

x1train,x1test,y1train,y1test=train_test_split(X,Y,test_size=0.2,random_state=i)
```

The Other model used in above diagram is gradient booster model,this is better than above mentioned 2 models,as the accuracy of this model is higher compared to other models.to implement gradient booster model import gradient boosting regressor from sklearn package.

```
]: grad = GradientBoostingRegressor(
       loss='squared_error',
       learning_rate=0.2,
       n_estimators=15,
       max_depth=5,
       random_state=42
   )
   grad
```

```
]: GradientBoostingRegressor
GradientBoostingRegressor(learning_rate=0.2, max_depth=5, n_estimators=15,
                          random_state=42)
```

```
]: grad.fit(x1train,y1train)
   res=grad.predict(x1test)
   res
```

To choose which model to use we have to see and compare performance of different models.
In our project we compared different models namely

1. Linear Regression
2. Random forest
3. Gradient booster

As there is only one data set to know the performance correctly, the K-Fold method is useful. In the K-fold cross validation method we divide limited data sets into K number of train data and test data , For each train and test data is used to find accuracy of model from which we get k number of accuracies.

After training on different models we came to know that gradient booster has more accuracy than linear regression and random forest.

6.5 Train And Test Of Gradient Booster Model

```
from sklearn.ensemble import GradientBoostingRegressor
X = newcopydata.drop(['charges'], axis=1)
Y= newcopydata['charges']

x1train,x1test,y1train,y1test=train_test_split(X,Y,test_size=0.2,random_state=i)
```

```
]: grad = GradientBoostingRegressor(
       loss='squared_error',
       learning_rate=0.2,
       n_estimators=15,
       max_depth=5,
       random_state=42
   )
   grad
```

```
]: GradientBoostingRegressor
   GradientBoostingRegressor(learning_rate=0.2, max_depth=5, n_estimators=15,
                             random_state=42)
```

```
]: grad.fit(x1train,y1train)
   res=grad.predict(x1test)
   res
```

The Other model used in above diagram is gradient booster model,this is better than above mentioned 2 models,as the accuracy of this model is higher compared to other models.to implement gradient booster model import gradient boosting regressor from sklearn package.

7. TESTING

In machine learning, testing is mainly used to validate raw data and check the ML model's performance. The main objectives of testing machine learning models are:

- Quality Assurance
- Detect bugs and flaws

Once your machine learning model is built (with your training data), you need unseen data test your model. This data is called testing data, and you can use it to evaluate the performance and progress of your algorithms' training and adjust or optimize it for improved results.

Testing data has two main criteria. It should:

- Represent the actual dataset
- Be large enough to generate meaningful predictions

7.1 TYPES OF TESTING

7.1.1 MANUAL TESTING

Manual Testing is a type of software testing in which test cases are executed manually by a tester without using any automated tools. The purpose of Manual Testing is to identify the bugs, issues, and defects in the software application. Manual software testing is the most primitive technique of all testing types and it helps to find critical bugs in the software application.

Any new application must be manually tested before its testing can be automated. Manual Software Testing requires more effort but is necessary to check automation feasibility. Manual Testing concepts does not require knowledge of any testing tool. One of the Software Testing Fundamental is "**100% Automation is not possible**". This makes Manual Testing imperative.

7.1.2 AUTOMATED TESTING

Automation Testing is a software testing technique that performs using special automated testing software tools to execute a test case suite. On the contrary, Manual Testing is performed by a human sitting in front of a computer carefully executing the test steps.

The automation testing software can also enter test data into the System Under Test, compare expected and actual results and generate detailed test reports. Software Test Automation demands considerable investments of money and resources.

7.2 TESTING LEVELS

7.2.1 NON-FUNCTIONAL TESTING

Non-functional testing is a type of software testing to test non-functional parameters such as reliability, load test, performance and accountability of the software. The primary purpose of non-functional testing is to test the reading speed of the software system as per non-functional parameters. The parameters of non-functional testing are never tested before the functional testing. Non-functional testing is also very important as functional testing because it plays a crucial role in customer satisfaction.

7.2.1.1 PERFORMANCE TESTING

Performance testing is a form of software testing that focuses on how a system running the system performs under a particular load. This is not about finding software bugs or defects. Different performance testing types measure according to benchmarks and standards. Performance testing gives developers the diagnostic information they need to eliminate bottlenecks.

7.2.1.2 STRESS TESTING

Stress Testing is a type of software testing that verifies stability & reliability of software application. The goal of Stress testing is measuring software on its robustness and error handling capabilities under extremely heavy load conditions and ensuring that software doesn't crash under crunch situations. It even tests beyond normal operating points and evaluates how software works under extreme conditions.

7.2.1.3 SECURITY TESTING

Security Testing is a type of Software Testing that uncovers vulnerabilities of the system and determines that the data and resources of the system are protected from possible intruders. It ensures that the software system and application are free from any threats or risks that can cause a loss. Security testing of any system is focused on finding all possible loopholes and weaknesses of the system which might result in the loss of information or repute of the organization.

7.2.1.4 PORTABILITY TESTING

Portability Testing is one of Software Testing which is carried out to determine the degree of ease or difficulty to which a software application can be effectively and efficiently transferred from one hardware, software or environment to another one. The results of portability testing are measurements of how easily the software component or application will be integrated into the environment and then these results will be compared to the non-functional requirement of portability of the software system.

7.2.1.5 USABILITY TESTING

Usability Testing also known as User Experience (UX) Testing, is a testing method for measuring how easy and user-friendly a software application is. A small set of target end-users, use software application to expose usability defects. Usability testing mainly focuses on user's ease of using application, flexibility of application to handle controls and ability of application to meet its objectives. This testing is recommended during the initial design phase of SDLC, which gives more visibility on the expectations of the users.

7.2.2 FUNCTIONAL TESTING

It is a type of software testing which is used to verify the functionality of the software application, whether the function is working according to the requirement specification. In functional testing, each function tested by giving the value, determining the output, and verifying the actual output with the expected value. Functional testing performed as black-box testing which is presented to confirm that the functionality of an application or system behaves as we are expecting. It is done to verify the functionality of the application. Functional testing also called as black-box testing.

7.3 TEST CASES

Sl.no	Testcase	Expected Result	Actual Result	Pass/ Fail
1.	After Training testing with test sample data	most of the out comes are right upto 80 % accuracy	detect fraud or not	PASS

8. RESULTS

```
In [105]: print(np.sqrt(mean_squared_error(y_test,y_pred)))
          print(reg.score(x_train,y_train))
          print(reg.score(x_test,y_test))
          cvscore=cross_val_score(reg,x,y,cv=5)
          print(cvscore.mean())

          5673.216672425889
          0.7337139559302677
          0.8015840795746242
          0.7463262858031846
```

```
          3494.7073117 , 14443.7344477 , 40960.3490283 , 14488.423923  ,
          5325.6438796 , 10639.0856124 ])

In [116]: #print(r2_score(xtrain,ytrain))
          print(rf_regressor.score(xtest,ytest))
          print(rf_regressor.score(xtrain,ytrain))
          print(cross_val_score(rf_regressor,X,Y,cv=5,).mean())

          0.8297264587116508
          0.9755927519384534
          0.8344447927764229
```

```
126]: print(grad.score(x1test,y1test))
       print(grad.score(x1train,y1train))
       print(cross_val_score(grad,X,Y,cv=5).mean())

       0.8542711525783899
       0.9096374033998812
       0.8500255094806718
```

From Above we can see Random Forest and Gradient Booster have high accuracies. So either of them can be used for health insurance prediction. As Gradient Booster have low standard deviation so we are considering the model(Gradient Booster) and test it with data for accuracy.

9. CONCLUSION AND FURTHER WORK

In today's world to find the most accurate and up-to-date information, it's recommended to check with insurance providers in India or relevant financial and insurance websites. Additionally, you may want to consult with insurance agents or financial advisors who can provide personalized guidance based on your specific situation and needs. So, this model is useful for individuals to know the up-to-date information of Insurance policies dependent on individual situation and needs along with comparison among various Insurance companies.

Model Performance Evaluation:

The application of k-fold cross-validation provides a robust and unbiased assessment of the health insurance prediction model's performance.

The calculated test Mean Squared Error (MSE) serves as a reliable metric to gauge the accuracy of predictions, taking into account the diversity of the dataset.

Parameter Importance:

The parameters utilized in the health insurance prediction model, such as age, gender, pre-existing conditions, and coverage type, play a significant role in determining the accuracy of predictions.

Understanding the impact of each parameter on the model's performance is crucial for refining the predictive capabilities of the model.

Generalization Ability:

The model's ability to generalize across different subsets of the health insurance dataset is a key consideration. The use of k-fold cross-validation ensures that the model performs well on unseen data, enhancing its reliability in real-world scenarios.

Optimization Opportunities:

Further optimization of the model parameters may be explored to enhance predictive accuracy. This could involve fine-tuning the weights assigned to different parameters or incorporating additional relevant features to improve overall model performance.

Decision Support Tool:

The developed health insurance prediction model can serve as a valuable decision support tool for insurance providers, policy-makers, and individuals. It offers insights into potential coverage outcomes based on various input parameters.

Further Work:

Feature Engineering:

Explore additional relevant features that could improve the predictive capabilities of the model. This might include socioeconomic factors, lifestyle choices, or geographic considerations.

Model Refinement:

Continue refining the health insurance prediction model by experimenting with different algorithms, regularization techniques, and hyperparameter tuning to achieve optimal performance.

Interpretability:

Enhance the interpretability of the model by employing techniques such as SHAP (SHapley Additive exPlanations) values or LIME (Local Interpretable Model-agnostic Explanations). This helps in understanding the contribution of each parameter to the model's predictions.

External Validation:

Validate the model's performance on external datasets to ensure its generalizability across diverse populations and healthcare systems.

User Interface Development:

Develop a user-friendly interface for the health insurance prediction model, making it accessible and interpretable for a broader audience, including insurance professionals and individuals seeking coverage.

Ethical Considerations:

Investigate and address any ethical considerations related to the use of the model, ensuring fairness, transparency, and accountability in its predictions, especially in the context of sensitive health-related data.

REFERENCES

[1] Kaushik, K., Bhardwaj, A., Dwivedi, A.D. and Singh, R., 2022. Machine learning-based regression framework to predict health insurance premiums. *International Journal of Environmental Research and Public Health*, *19*(13), p.7898.

[2] Sun, J.J., 2020. *Identification and Prediction of Factors Impact America Health Insurance Premium* (Doctoral dissertation, Dublin, National College of Ireland).

[3] Karlsson, M., Klohn, F. and Rickayzen, B., 2018. The role of heterogeneous parameters for the detection of selection in insurance contracts. *Insurance: Mathematics and Economics*, *83*, pp.110-121.

[4] Alzoubi, H.M., Sahawneh, N., AlHamad, A.Q., Malik, U., Majid, A. and Atta, A., 2022, October. Analysis Of Cost Prediction In Medical Insurance Using Modern Regression Models. In *2022 International Conference on Cyber Resilience (ICCR)* (pp. 1-10). IEEE.