**AI PROJECT**

**Healthcare Resource Demand Prediction and Patient Segmentation**

**Problem Statement:**

The objective of this project is to predict the demand for healthcare resources and segment patients based on their demographics. Additionally, we aim to build machine learning models to predict resource demand and patient health outcomes.

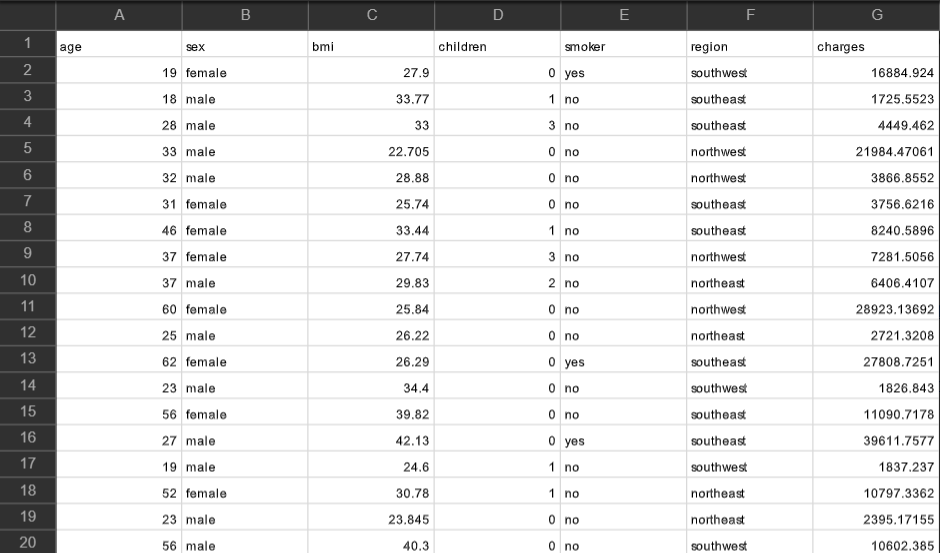
**Dataset**

Our data set named as insurance.csv consists of 7 columns with following fields

1. Age
2. Sex
3. Bmi
4. Children
5. Smoker
6. Region
7. Charges

**Target Variable:** The column represented by charges is taken as target variable.

Below is a screenshot of how our dataset looks like.



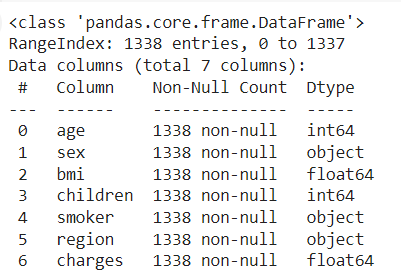
**Data Exploration and Preprocessing**

* The code uses **pd.read\_csv('insurance.csv')** to load the dataset from a CSV file named 'insurance.csv' into a pandas DataFrame called **data\_frame**.



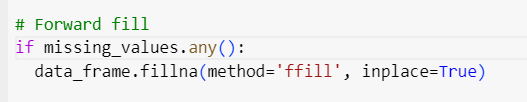
**Loading the dataset and exploring its structure and features**

**data\_frame.info()**: The **.info()** method provides a concise summary of the DataFrame, including the number of entries, column names, data types, and non-null counts. Below is the screenshot of the output of this operation:



**Handling missing values**

Missing values in any of the columns is removed through the method of forward fill as shown:



* Forward Fill: The **fillna()** method with **method='ffill'** fills missing values with the last observed non-null value in each column. The **inplace=True** parameter ensures that the changes are made directly to **data\_frame** without creating a new DataFrame.

**Handling outliers, and performing necessary data transformations**

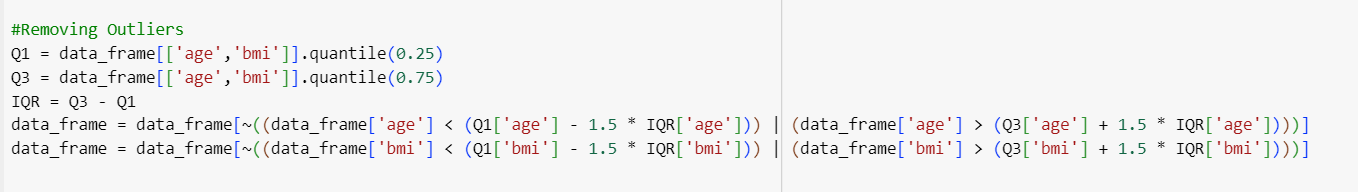
For removing any of the outliers we have used IRQ method. We are majorly dealing with two coloumns that are bmi and age for clustering and thus IRQ method is applied on these coloumns.

* **IRQ Method:**

The IQR is a measure of statistical dispersion, specifically a measure of the spread of a dataset. It is calculated as the difference between the 75th percentile (Q3) and the 25th percentile (Q1) of the data.

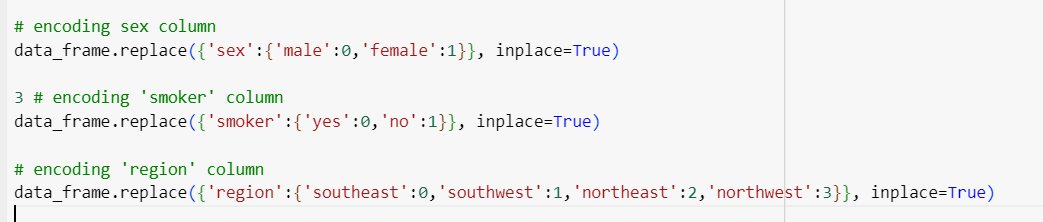
**Mathematically, IQR = Q3 - Q1.**

Any value that falls below Q1 - 1.5 \* IQR or above Q3 + 1.5 \* IQR is considered an outlier. These values are then removed from the dataset to prevent any analysis problem



**Encoding of different columns**

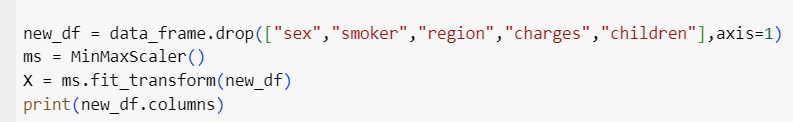
Encoding allows values to be represented numerically, which can be useful in certain types of analyses or machine learning models that require numerical inputs. We have used following encoding schemes:



* Code replaces 'male' with 0 and 'female' with 1.
* 'yes' is replaced with 0 and 'no' with 1 in smokers column.
* 'region' column is encoded based on a mapping where different regions ('southeast', 'southwest', 'northeast', 'northwest') are assigned numerical values (0, 1, 2, 3).

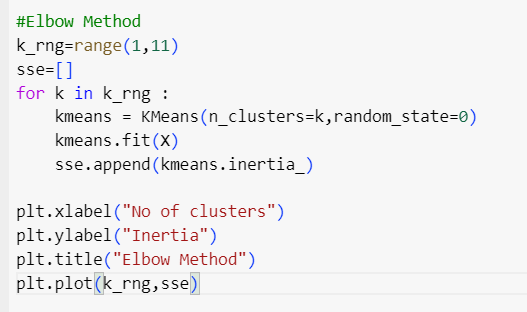
**Feature selection process and chosen features**

For the better results we choose the 2 best features i.e bmi and age and the remaining features are dropped for further execution as shown below.



**Patient Segmentation**

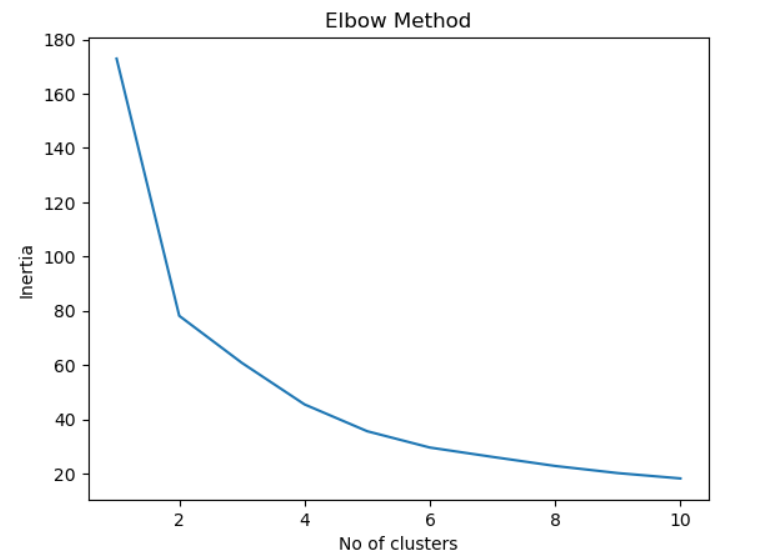
We have used K-means clustering for patient segmentation using elbow method.



We have used Elbow Method to determine the optimal number of clusters for K-means clustering. It iterates through a range of values for the number of clusters (**k\_rng=range (1,11)**) and calculates the Sum of Squared Errors (SSE) for each value of k.

**Execution of Elbow Method**

**sse=[]** initializes an empty list to store the SSE values. A loop is used to fit K-means clustering models with different numbers of clusters (**k**) and compute their inertia (SSE), which is then appended to the **sse** list. The Elbow Method plots the number of clusters against the SSE to find the "elbow point" where the SSE starts to decrease at a slower rate.

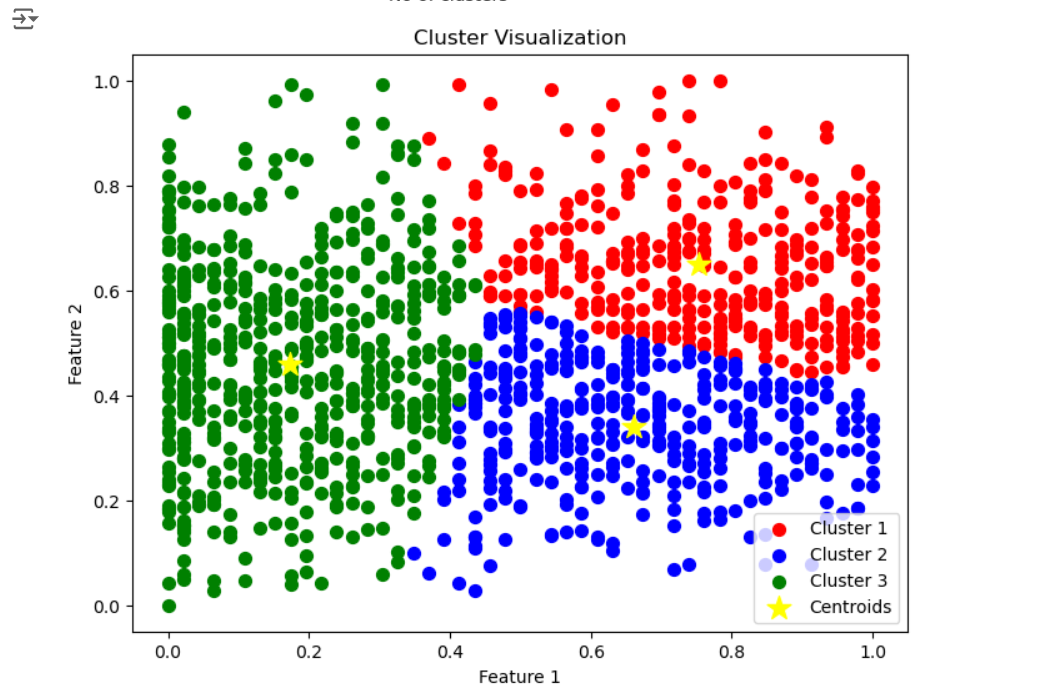


**K-means Clustering**

* K-means clustering is performed with the chosen number of clusters (**n\_clusters=3**).
* **y\_pred = km.fit\_predict(X)** fits the K-means model to the data (**X**) and assigns cluster labels to each data point based on the centroids.



* Each cluster is represented by a different color **red**, **blue**, **green** based on the cluster labels.
* Centroids of the clusters are plotted as yellow stars.



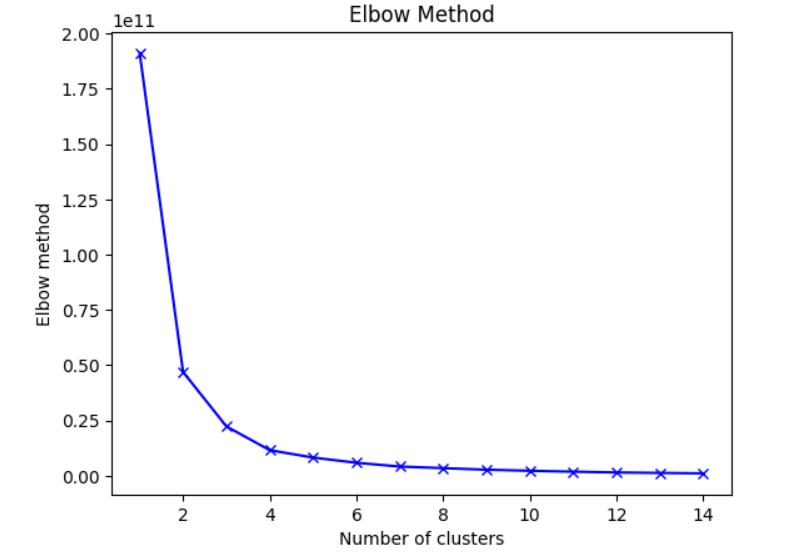
**Scatter plots showing healthcare charges vs. selected features.**

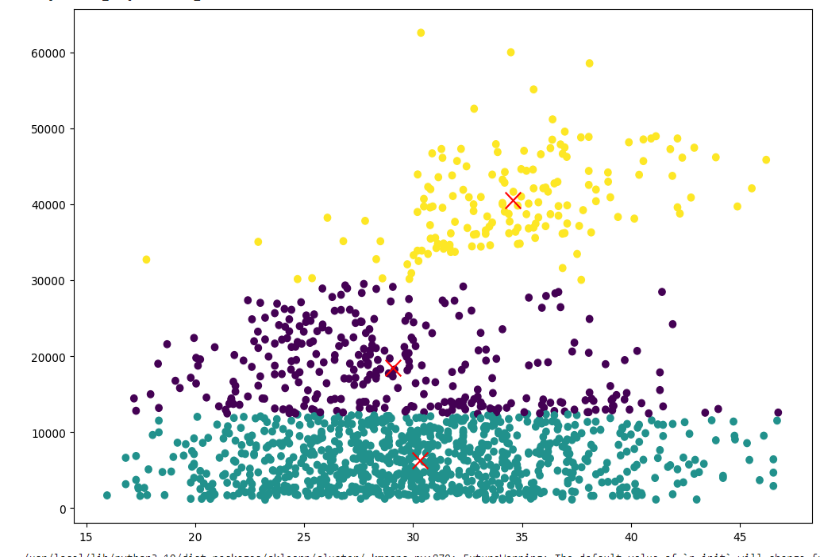
**Visualization between charges and BMI:**

Below code segment performs K-means clustering on the 'bmi' and 'charges' features, determines the optimal number of clusters using the Elbow Method, assigns cluster labels to the data points, and visualizes the clusters and centroids.



We determined the optimal number of clusters using the Elbow Method was 3.



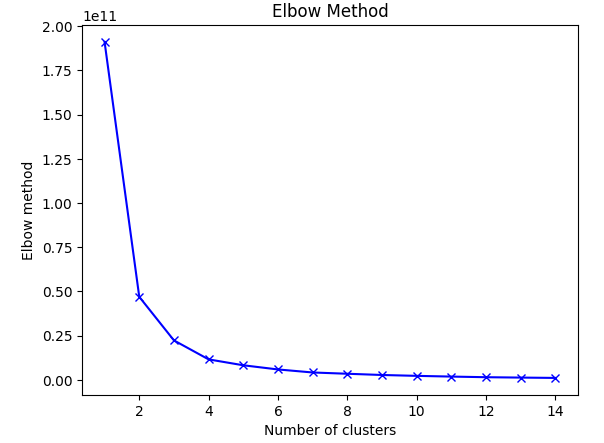
A K-means clustering model is created with **n\_clusters=3.**

**Visualization between charges and age:**

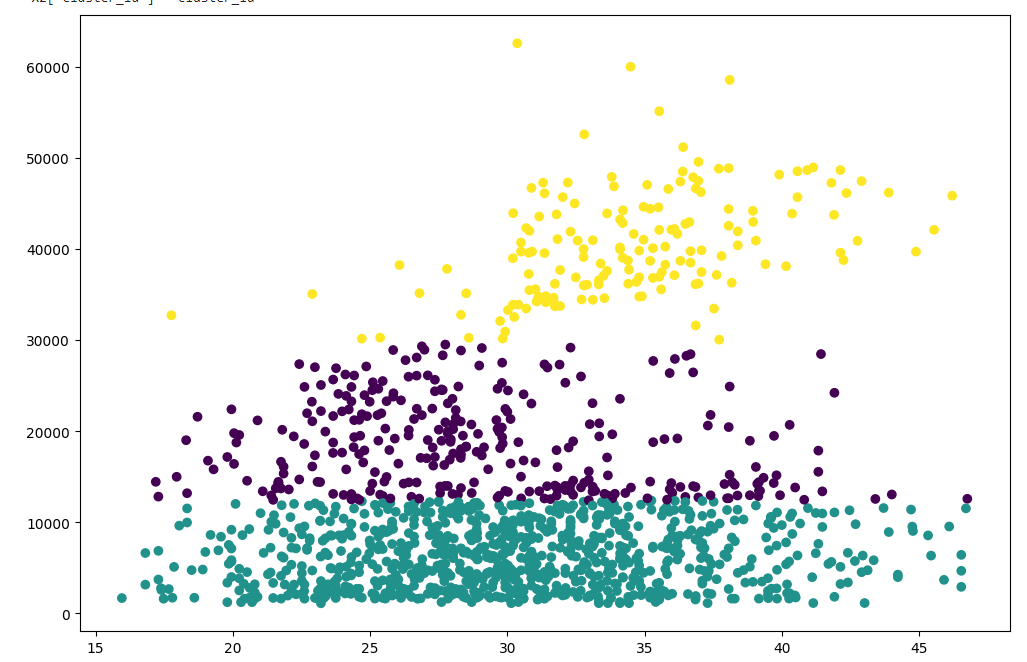
Below code segment performs K-means clustering on the 'age' and 'charges' features, determines the optimal number of clusters using the Elbow Method, assigns cluster labels to the data points, and visualizes the clusters and centroids.



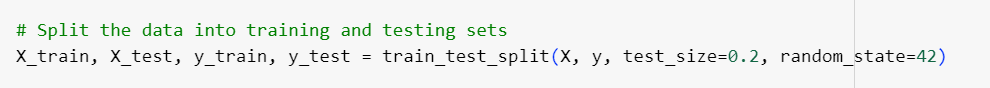
We determined the optimal number of clusters using the Elbow Method was 3.



A K-means clustering model is then created with **n\_clusters=3.**



**Splitting the data into training and testing sets**



* **X** contains the features (excluding the 'charges' column), and **y** contains the target variable ('charges').



* **train\_test\_split** from sklearn is used to split the data into training (**X\_train**, **y\_train**) and testing (**X\_test**, **y\_test**) sets. The test size is set to 20% of the data.

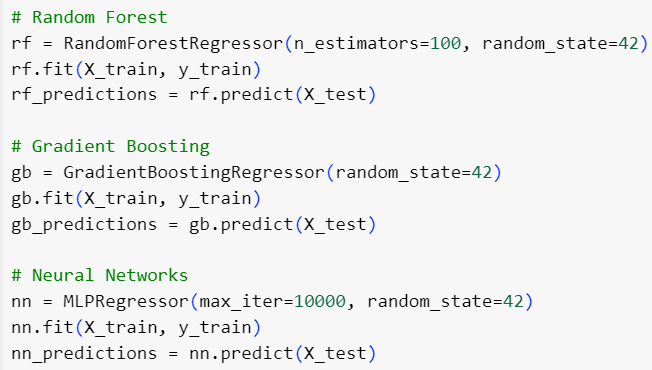
**Building Machine Learning Models**

**Random Forest:**

* A Random Forest Regressor model is initialized with 100 estimators and **random\_state=42** for reproducibility.
* The model is trained on the training data (**X\_train**, **y\_train**) using **rf.fit()**.
* Predictions are made on the test data (**X\_test**), and the predictions are stored in **rf\_predictions**.

**Gradient Bosting:**

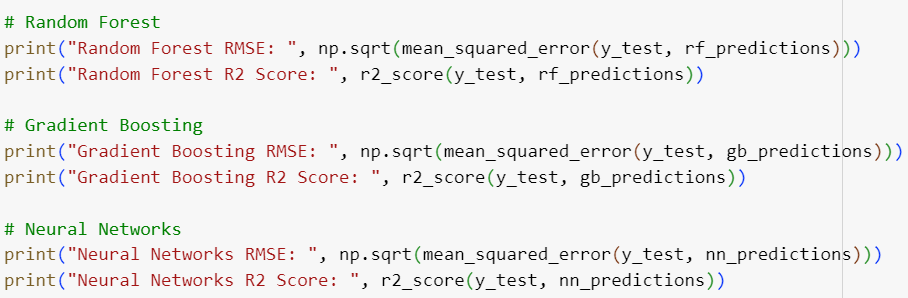
* A Gradient Boosting Regressor model is initialized with **random\_state=42** for reproducibility.
* The model is trained on the training data (**X\_train**, **y\_train**) using **gb.fit()**.
* Predictions are made on the test data (**X\_test**), and the predictions are stored in **gb\_predictions**.



**Neural Networks (Multi-layer Perceptron - MLP):**

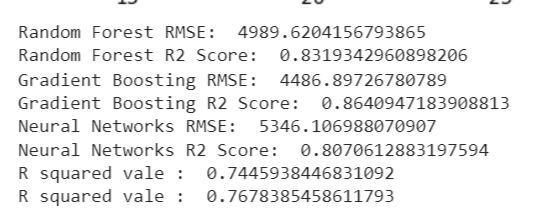
* An MLP Regressor model is initialized with a maximum of 10,000 iterations and **random\_state=42** for reproducibility.
* The model is trained on the training data (**X\_train**, **y\_train**) using **nn.fit()**.
* Predictions are made on the test data (**X\_test**), and the predictions are stored in **nn\_predictions**.

**Evaluating Models Performance**



* **RMSE** measures the average difference between predicted values and actual values, with lower values indicating better model performance.
* **R2 Score** quantifies the goodness of fit of the model to the data, with values closer to 1 indicating a better fit. It ranges from -∞ to 1, where 1 indicates a perfect fit.

**We obtained the following results:**

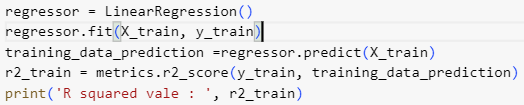


**Comparing the performance of different models**

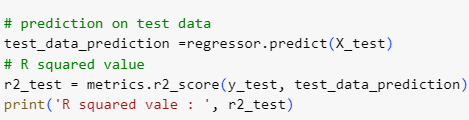
Among the models evaluated , the Gradient Boosting model performed better based on both RMSE and R2 score compared to Random Forest and Neural Networks.

**Simple and multiple linear regression model implementation and evaluation**

* A Linear Regression model is initialized and fitted to the training data (X\_train, y\_train) using regressor.fit().
* X\_train contains the independent variables, and y\_train contains the dependent variable.



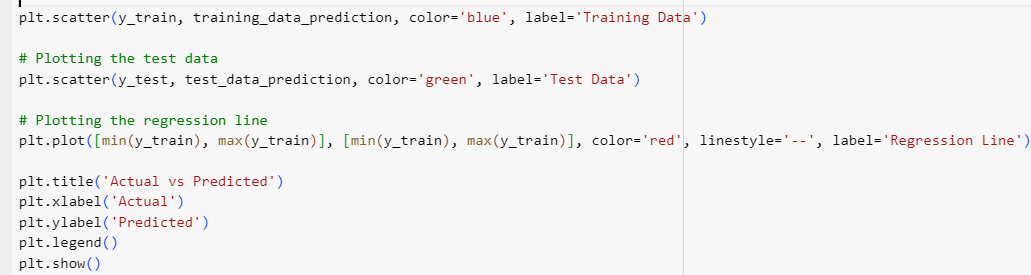
* Predictions are made on the training data (**X\_train**) using **regressor.predict()**, and the predicted values are stored in **training\_data\_prediction**.
* The R-squared (R2) score is calculated to evaluate the model's performance on the training data. The R2 score measures how well the regression model fits the actual data, with higher values indicating a better fit.



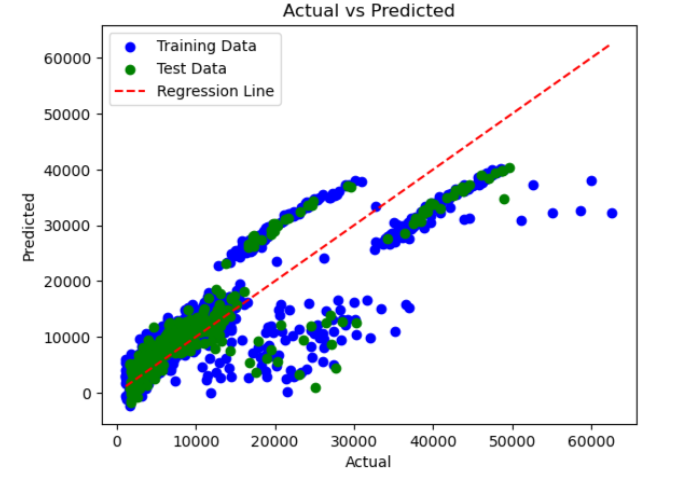
* Similarly, predictions are made on the test data (**X\_test**), and the R2 score is calculated to evaluate the model's performance on unseen data (test data). This helps assess how well the model generalizes to new data.

**Visualization of Model Performance**

* The below code segment creates a scatter plot to visualize the actual target values (**y\_train** and **y\_test**) against the predicted values (**training\_data\_prediction** and **test\_data\_prediction**).



* The red dashed line represents the regression line, indicating the relationship between actual and predicted values.



**Conclusion:**

**The following libraries are used in the code:**

* **import pandas as pd:** Data manipulation and analysis library in Python.
* **from sklearn. cluster import KMeans:** Unsupervised learning algorithm for clustering data points.
* **from sklearn. preprocessing import MinMaxScaler:** Scales features to a specified range (usually 0 to 1).
* **from sklearn. preprocessing import StandardScaler:** Standardizes features by removing the mean and scaling to unit variance.
* **from sklearn. decomposition import PCA**: Principal Component Analysis for dimensionality reduction.
* **import matplotlib. pyplot as plt:** Data visualization library for creating plots and charts
* **import seaborn as sns:** Data visualization library built on top of matplotlib, providing additional aesthetics
* **import NumPy as np:** Numerical computing library for handling arrays and mathematical operations.
* **from sklearn. Model\_selection import train\_test\_split:** Splits data into training and testing sets for model evaluation.
* **from sklearn. Ensemble import RandomForestRegressor:** Ensemble learning method for regression tasks.
* **from sklearn. Metrics import mean\_squared\_error, r2\_score**: Evaluation metrics for regression models.
* **from sklearn. Ensemble import GradientBoostingRegressor:** Ensemble learning method for regression tasks, boosting based.
* **from sklearn. linear\_model import LinearRegression:** Linear regression model for predicting continuous outcomes.
* **from sklearn. neural\_network import MLPRegressor**: Multi-layer Perceptron model for regression tasks.
* **from sklearn import metrics:** Library for various machine learning metrics and evaluation functions.

**Glossary:**

* **IRQ Method:** The IQR is a measure of statistical dispersion, specifically a measure of the spread of a dataset.
* **K-means clustering:** unsupervised machine learning algorithm that partitions data into K clusters based on similarity, aiming to minimize intra-cluster variance.
* **Elbow Method:** technique used in K-means clustering to determine the optimal number of clusters (K)
* **Centroids:** refer to the central points of clusters in k-mean algorithm
* **Random Forest:** It builds multiple decision tree models during training and combines their predictions to improve accuracy and reduce overfitting
* **Gradient Boosting:** It builds an ensemble of weak decision trees sequentially, where each new learner corrects errors made by the previous ones.
* **Neural Networks (Multi-layer Perceptron - MLP):** deep learning models consisting of interconnected layers of nodes (neurons) that process and transform input data through non-linear activation functions
* **Linear Regression model:** statistical method used for modeling the relationship between a dependent variable and one or more independent variables

## Conclusion:

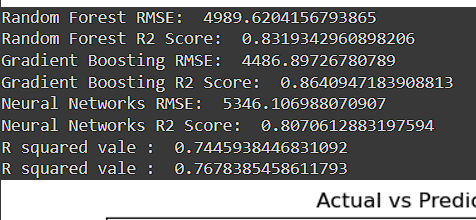
Comparing Linear Regression with Neural Networks:

* **R Squared Value**: Linear Regression has slightly lower R squared values compared to Neural Networks, both in training and testing. This suggests that Neural Networks capture more variance in the data and provide better predictions of resource demand and patient health outcomes.
* **RSME:** Neural Networks have a higher RSME compared to Linear Regression, indicating that they have higher prediction errors.

Comparing with Gradient Boosting:

* **R squared Value:** Gradient Boosting also achieves the highest R squared value, indicating that it captures the most variance in the data and provides the best fit to the target variable.
* **RSME:** Gradient Boosting has the lowest RSME, followed by Random Forest and then MLRegressor.This indicates Gradient Boosting provides the most accurate prediction, Followed by Random Forest.
* **Random Forest:** Random Forest exhibit a relatively low RSME compared to other models, indicating its effectiveness in minimizing prediction errors, Additionally, it demonstrates a commendable R squared value, suggesting a high degree of variance explained by the model, making it a robust choice for predicting resource demand and patient health outcomes in this context.

In summary, Gradient Boosting appears to be the best-performing model among the compared models, followed by Random Forest.



## Importance of Feature Selection:

By carefully selecting relevant variables and excluding irrelevant ones, predictive model can focus on the most influential factors for healthcare resource demand prediction and patient segmentation. This process streamlines the model’s focus, enhances predictive power, and reduces noise and overfitting.In the provided code, feature selection is exemplifies by excluding variables like ‘sex’, ‘smoker’, ‘region’ and ‘children, thereby refining the models predictive capabilities, Through techniques like Kmeans clustering, the data is further distilled to identify meaningful patterns, facilitating more precise patient segmentation and resource allocation