DSE CAPSTONE PROJECT

PREDICTING HOSPITAL STAY DURATION USING PATIENT AND ADMISSION DATA: A DATA-DRIVEN APPROACH TO OPTIMIZE HEALTHCARE RESOURCES

Batch	DSE - FT - Chennai, May 2024 - G2
Team Members	1. Abinash Balasubramanian 2. Akash Thiruveedula 3. Harshitha Babu 4. RajKumar R 5. Harish R
Domain	Healthcare Analytics
Group Number	2
Team Leader	Abinash Balasubramanian
Mentor Name	Jatinder Bedi

Date:3-12-2024

B.abinash

Signature of the Mentor **Signature of the Team Leader**

TABLE OF CONTENTS

INTRODUCTION:

DATA SET BACKGROUND:

In the healthcare industry, managing hospital resources effectively is crucial for delivering high-quality patient care. One significant challenge faced by healthcare providers is the unpredictable length of patient stays, which complicates bed management, staffing, and overall operational efficiency. Traditionally, hospitals have relied on historical averages or manual estimations to forecast the duration of stays; however, these methods often lack accuracy and fail to consider the unique factors influencing each patient's situation.

Recent advancements in data analytics and machine learning present an opportunity to improve the prediction of hospital stays. By analyzing various data points—from patient demographics to admission types—hospitals can gain insights that allow for more precise forecasting. This project leverages these advancements to develop a model that predicts hospital stay durations, ultimately aiming to facilitate better resource management and enhance patient outcomes.

ABSTRACT:

The increasing complexity of healthcare demands efficient resource management to enhance patient care and operational efficiency in hospitals. This project aims to develop a predictive model to estimate the duration of hospital stays using patient and admission data, including demographics, health conditions, and severity of illness. By employing advanced machine learning techniques, the project seeks to uncover patterns in historical data that can inform better decision-making processes in hospital management. The outcomes are anticipated to reduce wait times, optimize bed utilization, and ultimately improve patient satisfaction. This predictive approach is vital for addressing the challenges posed by unpredictable patient stays, leading to enhanced healthcare outcomes and resource allocation.

SCOPE OF THE PROJECT:

Predictive Model Development: The primary focus is to create a machine learning model that accurately predicts hospital stay durations using various patient demographics and admission data.

Data Analysis: The project will analyze a comprehensive set of data points, including patient demographics, admission types, health conditions, and severity of illness, to identify patterns that influence hospital stay durations.

Resource Optimization: By leveraging the predictive model, the project aims to provide actionable insights to hospital administrators to optimize bed utilization, staffing levels, and resource planning.

Improving Patient Outcomes: The initiative seeks to enhance patient care by reducing wait times for beds and ensuring timely treatment, ultimately leading to improved healthcare outcomes and increased patient satisfaction.

Model Validation: The project will include a robust evaluation of the model's performance to ensure its effectiveness and applicability in real-world healthcare settings, allowing for future scalability and integration into existing hospital systems.

CURRENT CHALLENGES:

Unpredictable Length of Stay: Hospitals often struggle with accurately predicting the duration of patient stays, which complicates bed management and resource allocation. Ineffective

Traditional Methods: Historical averages and manual estimations for forecasting patient stays are often inaccurate, as they fail to account for individual patient factors and the complexity of healthcare dynamics.

Data Complexity: The integration and analysis of diverse data points, including demographics, health conditions, and severity of illness, can be challenging. Ensuring the quality and completeness of this data is critical for accurate predictions.

Implementation of Advanced Analytics: Hospitals may face difficulties in adopting advanced data analytics and machine learning techniques due to resource constraints, lack of technical expertise, and potential resistance to change in established processes.

Validation and Real-World Applicability: Ensuring that the developed predictive model performs well in real-world settings requires thorough evaluation and validation, which can be resource-intensive.

INDUSTRY REVIEW:

OVERVIEW OF THE HEALTHCARE INDUSTRY:

The healthcare industry is a complex network of organizations, providers, and institutions dedicated to delivering medical services to patients. It encompasses a wide range of services, including preventive, diagnostic, therapeutic, and rehabilitative care. As one of the largest and fastest-growing sectors globally, the healthcare industry plays a crucial role in improving population health outcomes.

CURRENT PRACTICES:

Resource Management: Hospitals rely on historical data and manual estimates to predict patient stay durations, often leading to inefficiencies.

Data Utilization: Basic demographic and admission information is used, but lacks depth in personalized patient data analysis.

Technology Adoption: Limited integration of advanced technologies such as machine learning in operational practices for predicting hospital stay lengths.

Challenges: Inaccurate predictions contribute to bed shortages, staffing inefficiencies, and suboptimal patient care.

BACKGROUND RESEARCH:

Predictive Analytics: Studies highlight the effectiveness of machine learning in enhancing prediction accuracy for hospital stays.

Personalized Approaches: Research indicates that tailored models using patient-specific factors outperform traditional methods.

Healthcare Optimization: Effective prediction models have demonstrated potential in improving resource allocation and operational efficiency in hospitals.

Case Studies: Successful implementations of predictive models have shown reduced wait times and improved patient outcomes in various healthcare settings.

LITERATURE SURVEY:

TITLE	AUTHOR	DESCRIPTION	YEAR
Data-driven methodology to predict the ICU length of stay: A multicentre study of 99,492 admissions in 109 Brazilian units	Igor Tona Peres, Silvio Hamacher, Fernando Luiz Cyrino Oliveira, Fernando Augusto Bozza, Jorge Ibrain Figueira Salluh	The length of stay (LoS) is one of the most used metrics for resource use in Intensive Care Units (ICU). We propose a structured data-driven methodology to predict the ICU length of stay and the risk of prolonged stay, and its application in a large multicentre Brazilian ICU database.	2022
Data analytics for the sustainable use of resources in hospitals: Predicting the length of stay for patients with chronic diseases	Hamed M. Zolbanin, Behrooz Davazdahemami, Dursun Delen, Amir Hassan Zadeh	This study develops a deep learning model to predict COPD and pneumonia patients' length of stay, improving hospital resource optimization, bed utilization, operational efficiency, environmental impact, and patient satisfaction.	2022
Data-Driven Prediction for COVID-19 Severity in Hospitalized Patients	Abdulrahman A. Alrajhi, Osama A. Alswailem, Ghassan Wali, Khalid Alnafee, Sarah AlGhamdi, Jhan Alarifi, Sarab AlMuhaideb, Hisham ElMoaqet, Ahmad AbuSalah	This study details a Middle Eastern hospital's development of a real-time COVID-19 severity prediction tool, using machine learning on data from 1386 patients. The tool, validated by clinicians, supports efficient resource allocation. A random forest model achieved strong performance, with AUCs of 0.83 and 0.87 and high recall and precision across severity classes.	2022

A systematic review of the prediction of hospital length of stay: Towards a unified framework	Kieran Stone, Reyer Zwiggelaar, Phil Jones, Neil Mac Parthaláin	This paper reviews methods for predicting hospital length of stay (LoS), highlighting the benefits and limitations of current approaches. It proposes a unified framework to generalize prediction methods, enabling comparisons across hospitals. The study suggests using novel methods like fuzzy systems and enhancing model interpretability to improve LoS predictions.	2022
Data-driven optimization methodology for admission control in critical care units	Amirhossein Meisami, Jivan Deglise-Hawkinson, Mark E. Cowen, Mark P. Van Oyen	This study addresses critical care admissions by modeling the hospital as a complex queueing network, optimizing patient admissions through a Mixed Integer Programming model. By implementing a dual-threshold policy, admissions to ICUs and IMCs increased while reducing patient risk thresholds. The approach enhances care accessibility, personalizing resource allocation based on patient needs.	2018

DATASET AND DOMAIN

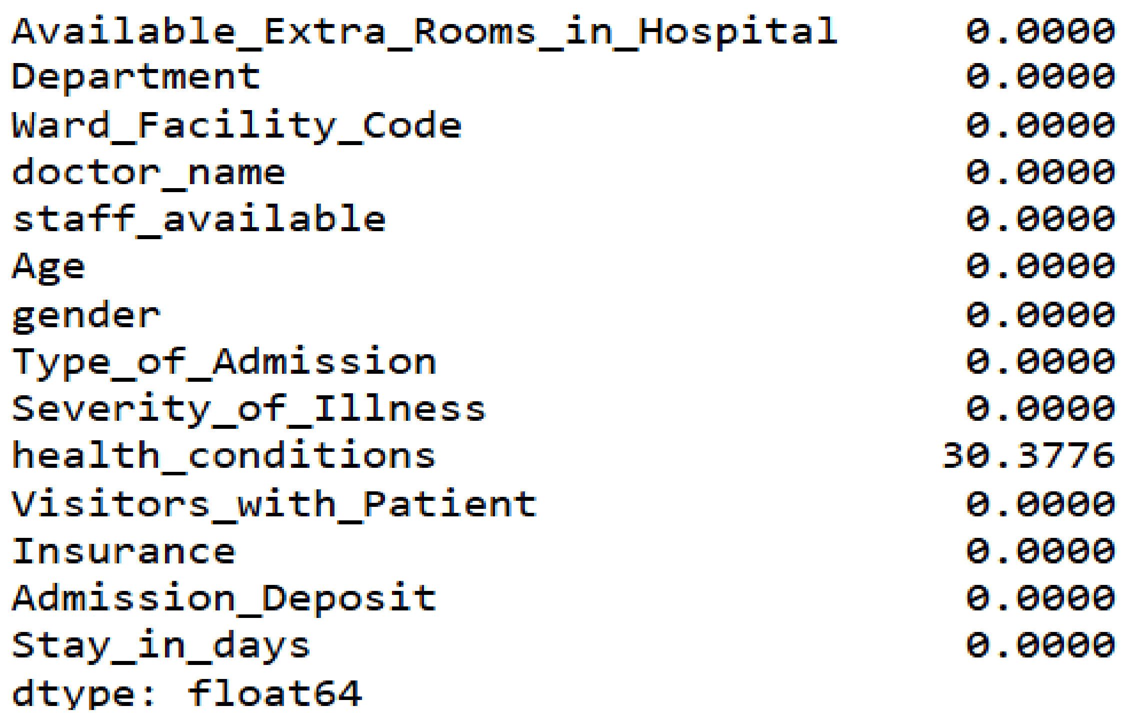
DATA DICTIONARY:

Feature Name	Description	Type
Available Extra Rooms in Hospital	Number of extra rooms available in the hospital at the time of admission	Integer
Department	The hospital department where the patient was admitted (e.g., Surgery, General Medicine)	Categorical
Ward_Facility_Code	Code of the ward where the patient was admitted	Categorical
doctor_name	Name of the attending doctor during admission	Categorical
staff_available	Number of medical staff available at the time of admission	Integer
patient_id	Unique identifier for each patient	Integer
Age	Age of the patient	Integer
gender	Gender of the patient (e.g., Male, Female)	Categorical
Type of Admission	Type of patient admission (e.g., Emergency, Routine)	Categorical
Severity of Illness	Severity level of the patient's condition at the time of admission (e.g., Mild, Moderate, Severe)	Categorical
health_conditions	Number of existing health conditions the patient has (e.g., diabetes, hypertension)	Categorical
Visitors with Patient	Number of visitors accompanying the patient during the hospital stay	Integer
Insurance	Whether the patient has insurance coverage (Yes/No)	Categorical
Admission_Deposit	Initial deposit amount paid at the time of admission (in currency)	Numeric
Stay (in days)	Duration of the patient's hospital stay (Target Variable)	Integer

VARIABLE CATEGORIZATION:

- There are 6 numerical features.
  - Available_Extra_Rooms_in_Hospital
  - staff_available
  - patientid
  - Visitors_with_Patient
  - Admission_Deposit
  - Stay_in_days
- There are 9 categorical features.
  - Department
  - Ward_Facility_Code
  - doctor_name
  - Age
  - gender
  - Type_of_Admission
  - Severity_of_Illness
  - health_conditions
  - Insurance

PRE PROCESSING DATA ANALYSIS:



```
Available_Extra_Rooms_in_Hospital     0.0000
Department                            0.0000
Ward_Facility_Code                    0.0000
doctor_name                           0.0000
staff_available                       0.0000
Age                                   0.0000
gender                                0.0000
Type_of_Admission                     0.0000
Severity_of_Illness                   0.0000
health_conditions                    30.3776
Visitors_with_Patient                 0.0000
Insurance                             0.0000
Admission_Deposit                     0.0000
Stay_in_days                          0.0000
dtype: float64
```

- Count of Missing values: In health_conditions: 151888/500000 (30%)

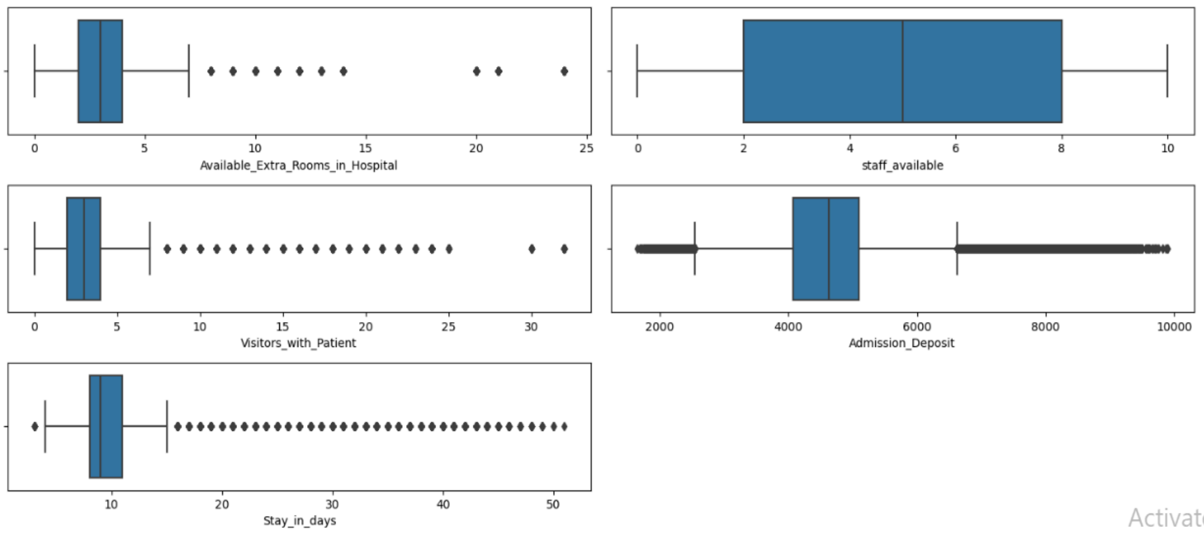
Every data point is important in the healthcare context. Removing the entire "health_conditions" column would result in the loss of valuable information. Therefore, we opted to drop only the null values from this column, preserving the remaining data for analysis.

- Redundant columns: Patient ID
- There are no duplicate rows and columns.

OUTLIER TREATMENT:

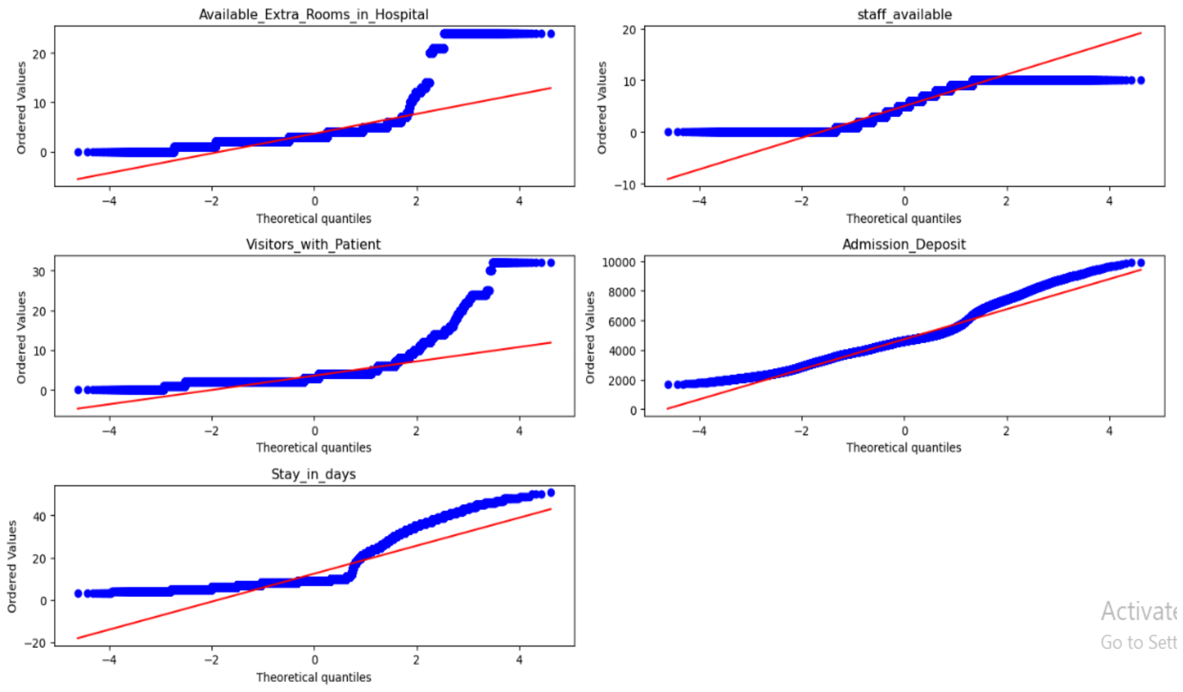
```
Available_Extra_Rooms_in_Hospital      -1.000000
staff_available                        -7.000000
Visitors_with_Patient                  -1.000000
Admission_Deposit                    2541.867254
Stay_in_days                            3.500000
dtype: float64
Available_Extra_Rooms_in_Hospital       7.000000
staff_available                        17.000000
Visitors_with_Patient                   7.000000
Admission_Deposit                    6621.459994
Stay_in_days                           15.500000
dtype: float64
```

The dataset contains approximately 34% outliers, which is a significant proportion. However, due to the nature of the healthcare domain, where every piece of data is crucial for analysis, we cannot remove or cap these outliers. Therefore, we will proceed to the next steps in our analysis while retaining the outliers as they may provide valuable insights.



DATA EXPLORATION (EDA)

1.SKEWNESS OF DATA:



1. **Available_Extra_Rooms_in_Hospital:**

   Shows curvature at the tails, indicating potential non-normality (heavy tails or skewness).

2. **staff_available:**

   The plot suggests the presence of steps, possibly due to a limited number of distinct values (discrete data).

3. **Visitors_with_Patient:**

   Deviates significantly at the tails, indicating heavy-tailed or skewed data.

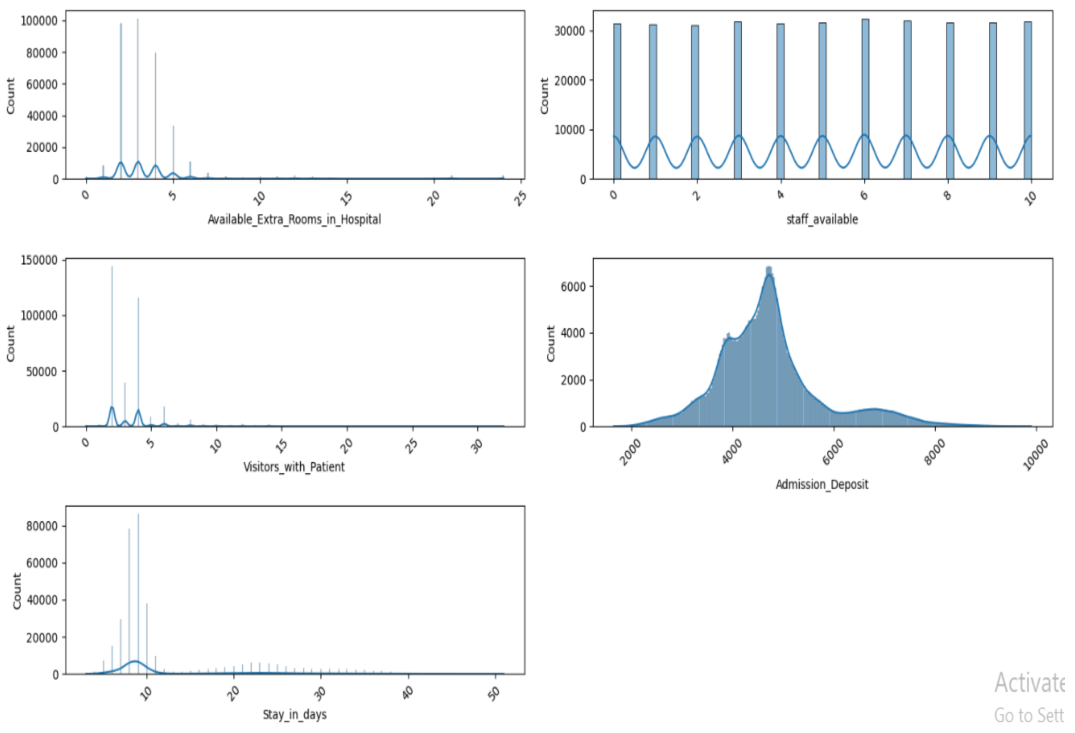
4. **Admission_Deposit:**

   Shows a deviation in the upper tail, suggesting skewness or outliers.

5. **Stay_in_days:**

   Also deviates from the straight line, indicating that it is not normally distributed.

2. DISTRIBUTION OF DATA:



1. **Available_Extra_Rooms_in_Hospital**:

   Distribution is highly skewed towards smaller values, with most data concentrated between 0 and 5.

2. **Staff_available:**

   Appears to have discrete uniform-like distribution, likely representing categories or limited options.

3. **Visitors_with_Patient:**

   Skewed towards smaller counts, with peaks at specific intervals, indicating discrete data.

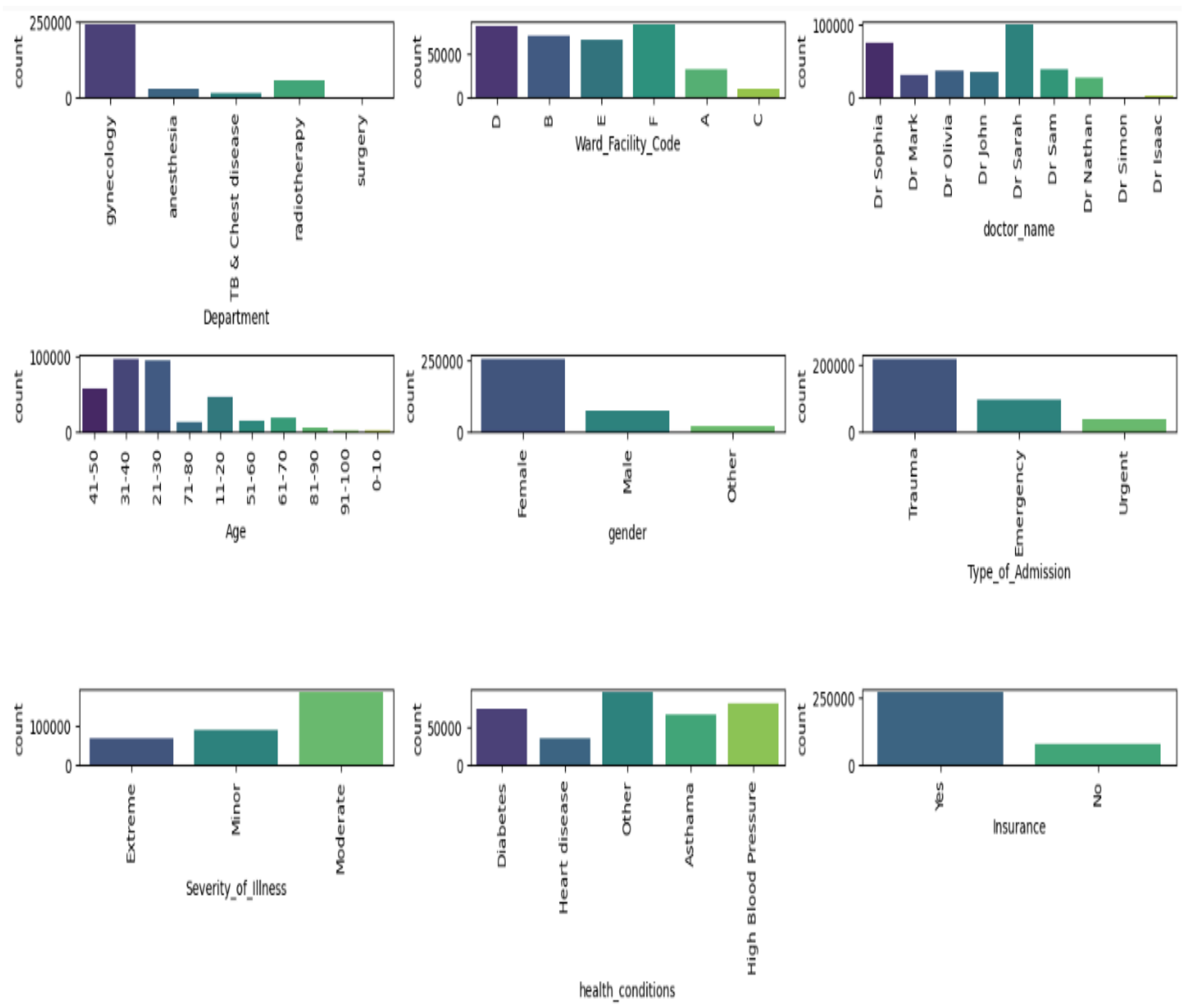
4. **Admission_Deposit:**

   Bimodal or multimodal distribution, suggesting there may be distinct groups in the data (e.g., different patient or hospital categories).

5. **Stay_in_days:**

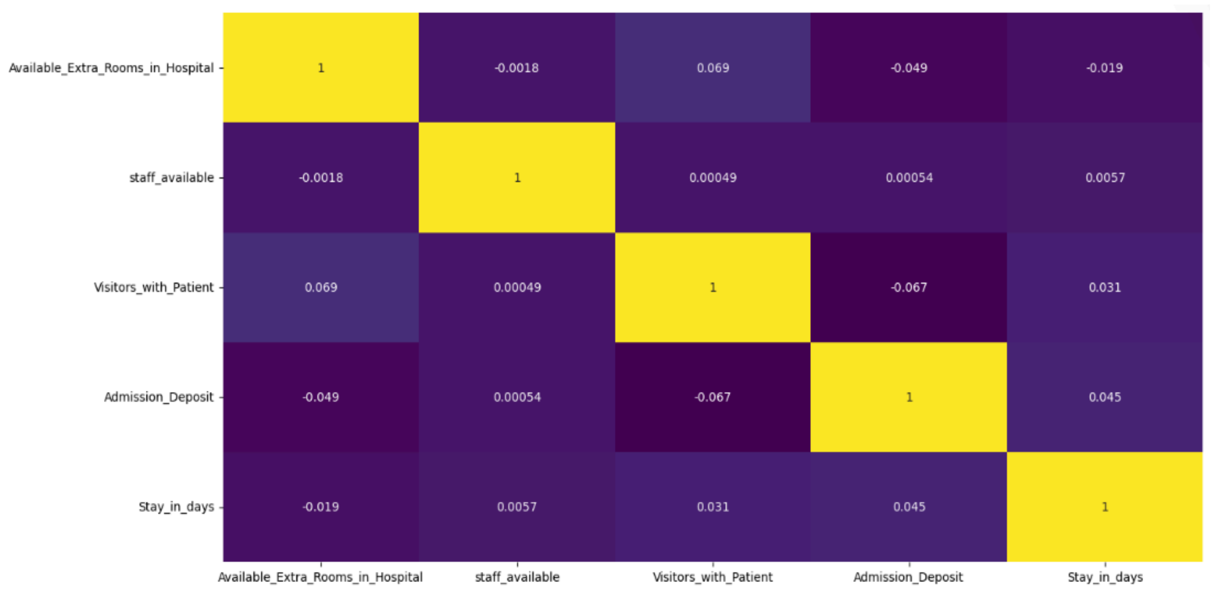
   Right-skewed, with most values concentrated at lower days of stay and a long tail extending towards higher values.

3.UNIVARIATE ANALYSIS:



This visualization highlights key trends in hospital data: Gynecology dominates among departments, with the majority of patients being females aged 31-50. Most admissions are for trauma cases with moderate illness severity, and High Blood Pressure is a common health condition. A significant portion of patients have insurance, and certain doctors like Dr. Sophia and Dr. Sarah handle the highest patient loads. These patterns can guide resource allocation and targeted healthcare programs.

4.MULTIVARIATE ANALYSIS:



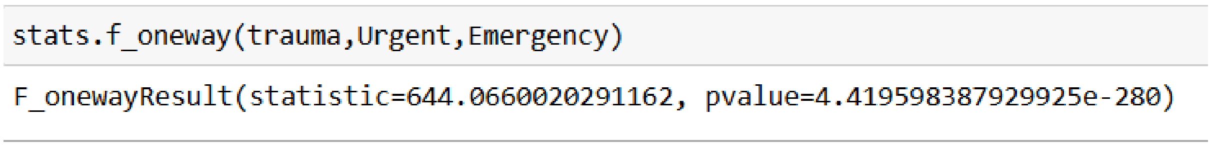
INFERENCE:

- Most features show very low correlation with the target variable "Stay_in_days." The highest correlation with "Stay_in_days" is observed in "Admission_Deposit" (0.0445) and "Visitors_with_Patient" (0.0308), suggesting that these features have only minimal linear relationships with the length of hospital stay.
- "Available_Extra_Rooms_in_Hospital" and "Visitors_with_Patient" have a positive correlation of 0.0693, indicating a weak association. However, this correlation is still minimal, implying limited dependency between the two variables.
- "Admission_Deposit" has a weak negative correlation with "Visitors_with_Patient" (-0.0672) and "Available_Extra_Rooms_in_Hospital" (-0.0495), suggesting that as the deposit amount changes, it is minimally associated with variations in room availability and visitor numbers.
- Overall, the correlation matrix indicates that most features have low correlation values with each other and with the target variable, meaning that these variables may contribute independently in the model without strong multicollinearity issues.

STATISTICAL TEST

1. ANOVA TEST BETWEEN TYPES OF ADMISSION AND STAY IN DAYS

```
trauma = df2[df2['Type of Admission']== 'Trauma']['Stay (in days)']
Urgent= df2[df2['Type of Admission']== 'Urgent']['Stay (in days)']
Emergency= df2[df2['Type of Admission']== 'Emergency']['Stay (in days)']
```

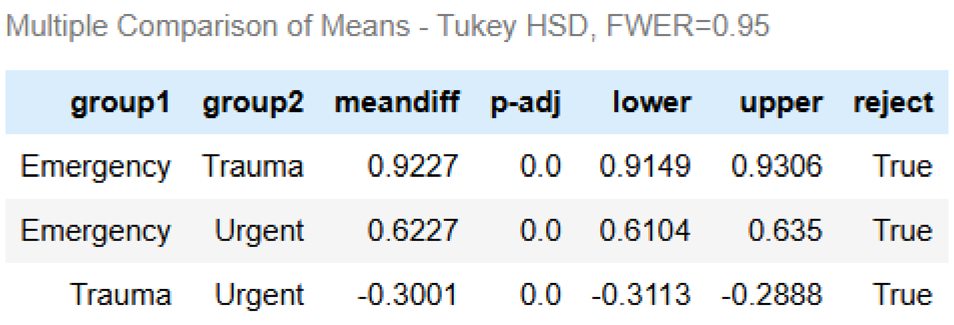


```
stats.f_oneway(trauma,Urgent,Emergency)
```

```
F_onewayResult(statistic=644.0660020291162, pvalue=4.419598387929925e-280)
```

Since the p-value is lower than the significance level, we reject the null hypothesis. This indicates that at least one mean differs, which is expected in this case, as different types of admissions are likely to have varying average lengths of stay.

FINDING THE UNEQUAL MEAN USING TUKEYHSD



Multiple Comparison of Means - Tukey HSD, FWER=0.95

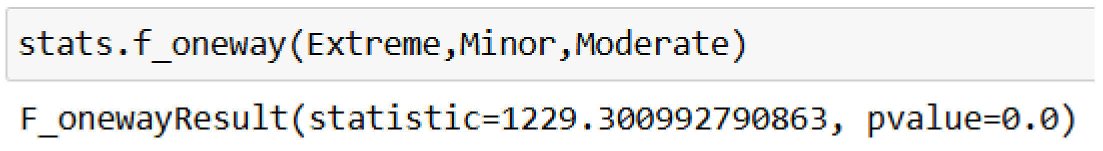
group1	group2	meandiff	p-adj	lower	upper	reject
Emergency	Trauma	0.9227	0.0	0.9149	0.9306	True
Emergency	Urgent	0.6227	0.0	0.6104	0.635	True
Trauma	Urgent	-0.3001	0.0	-0.3113	-0.2888	True

2. ANOVA TEST BETWEEN SEVERITY OF ILLNESS AND STAY IN DAYS

Extreme = df2[df2['Severity of Illness']=='Extreme']['Stay (in days)']

Minor= df2[df2['Severity of Illness']=='Minor']['Stay (in days)']

Moderate= df2[df2['Severity of Illness']=='Moderate']['Stay (in days)']

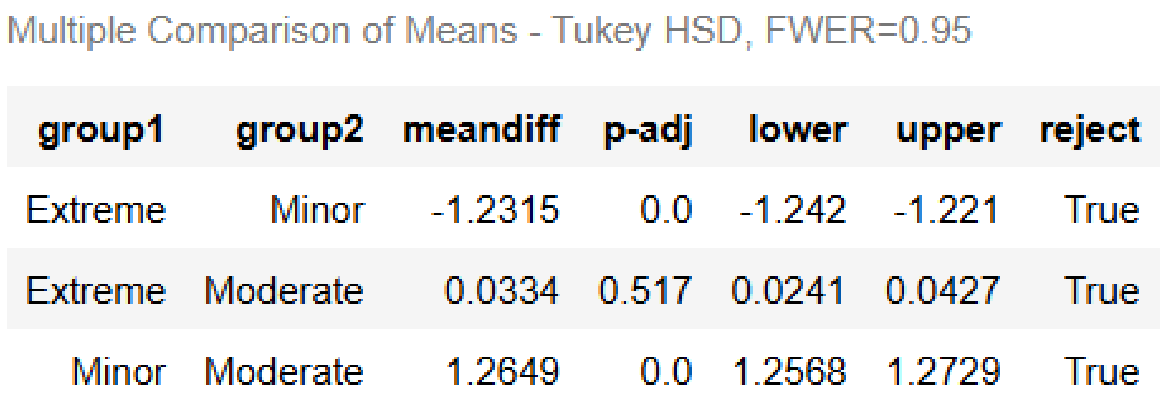


```
stats.f_oneway(Extreme,Minor,Moderate)
```

```
F_onewayResult(statistic=1229.300992790863, pvalue=0.0)
```

As the p-value is lower than the significance level, we reject the null hypothesis. This suggests that at least one mean is different, which makes sense in this context since different admission types are likely to have varying average lengths of stay.

FINDING THE UNEQUAL MEAN USING TUKEYHSD :



Multiple Comparison of Means - Tukey HSD, FWER=0.95

group1	group2	meandiff	p-adj	lower	upper	reject
Extreme	Minor	-1.2315	0.0	-1.242	-1.221	True
Extreme	Moderate	0.0334	0.517	0.0241	0.0427	True
Minor	Moderate	1.2649	0.0	1.2568	1.2729	True

FEATURE ENGINEERING

To convert categorical variables into a numerical format, which is essential for regression modeling, we applied the following encoding methods:

- **Binary Encoding**:
  - **Gender**: Encoded as 0 for Female, 1 for Male, and 2 for Other. This transformation allows the model to interpret gender in a numerical context.
  - **Insurance**: Mapped to 1 for 'Yes' and 0 for 'No', enabling the model to assess the impact of insurance status on hospital stay predictions.
  - **Severity of Illness**: Transformed into ordinal integers representing Minor (0), Moderate (1), and Extreme (2) severity levels.

gender

```
df['gender'].replace({'Female':0,'Male':1,'Other':2},inplace = True)
```

Severity_of_Illness

```
df['Severity_of_Illness'].replace({'Minor':0,'Moderate':1,'Extreme':2},inplace = True)
```

Insurance

```
df['Insurance'].replace({'Yes':1,'No':2},inplace = True)
```

- **Frequency Encoding**:
  - **Type of Admission**: Each type of admission was encoded based on the proportion of cases it represents in the dataset, capturing its relative frequency and importance in the context of patient outcomes.
  - **Department**: Each department was replaced with its relative contribution to the dataset, represented by the proportion of cases managed by that department. This encoding highlights the department's influence on the overall dataset.

- **Health Conditions**: Each health condition was transformed using its proportion of occurrences within the dataset, reflecting its prevalence and impact on patient outcomes.
- **Doctor Name**: Each doctor's name was encoded based on the proportion of cases they handled, illustrating their relative contribution and significance within the dataset.

doctor_name

```
a= df['doctor_name'].value_counts(normalize=True)
```

```
df['doctor_name']= df['doctor_name'].map(a)
```

Department

```
dept = df['Department'].value_counts(normalize=True)
df['Department'] = df['Department'].map(dept)
```

Type of admission

```
adm = df['Type_of_Admission'].value_counts(normalize=True)
df['Type_of_Admission'] = df['Type_of_Admission'].map(adm)
```

Health conditions

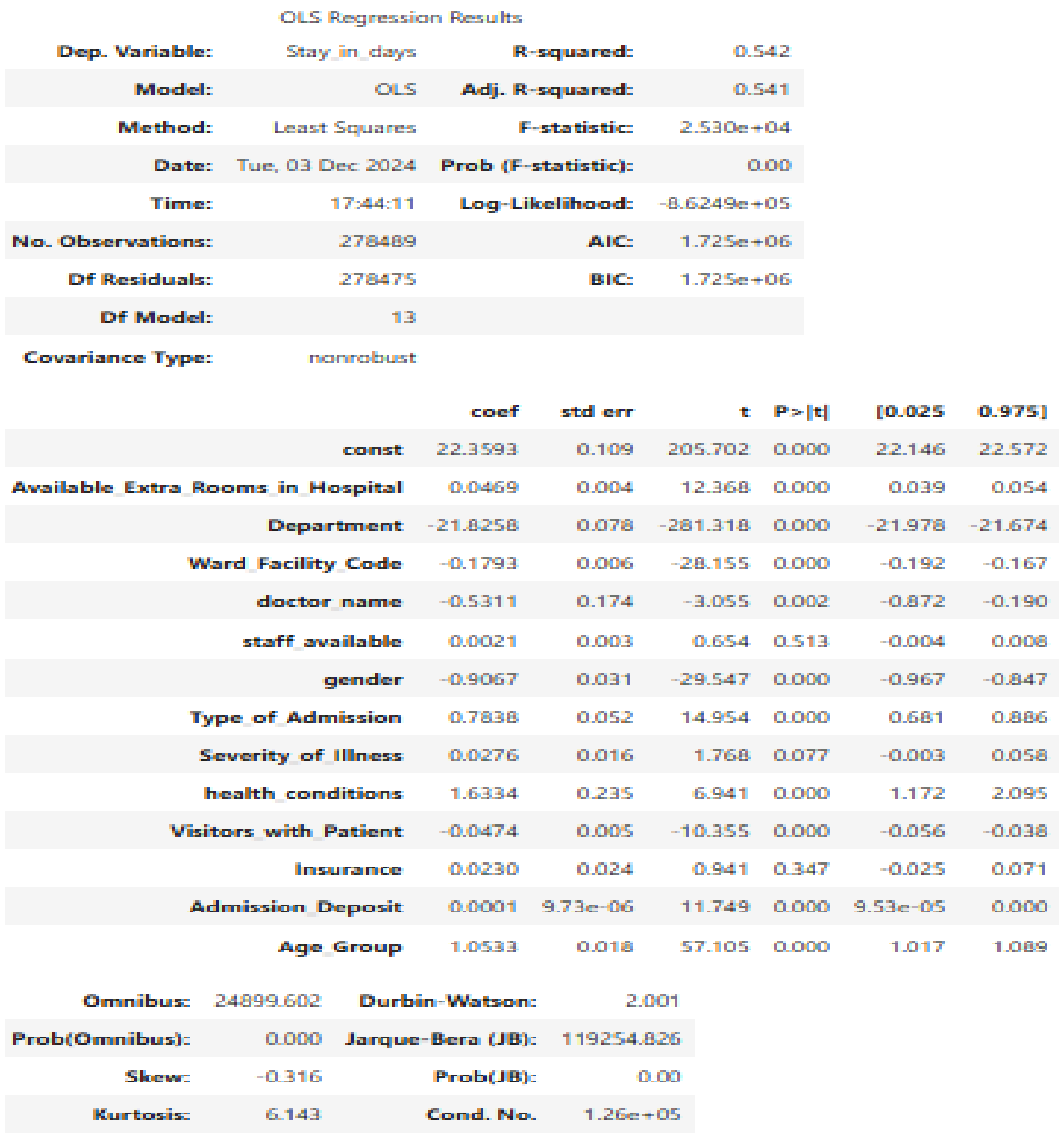
```
helt = df['health_conditions'].value_counts(normalize=True)
df['health_conditions'] = df['health_conditions'].map(helt)
```

BASE MODEL :

Reason for choosing Stats OLS model as base for model building:

The OLS (Ordinary Least Squares) model was chosen because it is simple, interpretable, and effective for modeling linear relationships between dependent and independent variables. It provides clear coefficients to quantify the impact of predictors, and the stats model offers useful diagnostic tools to evaluate model fit and assumptions. OLS is efficient when assumptions like linearity and homoscedasticity hold, and it allows for easy testing of variable significance, making it a reliable choice for regression analysis.

The below gives the base model output:



OLS Regression Results

Dep. Variable:	Stay_in_days	R-squared:	0.542
Model:	OLS	Adj. R-squared:	0.541
Method:	Least Squares	F-statistic:	2.530e+04
Date:	Tue, 03 Dec 2024	Prob (F-statistic):	0.00
Time:	17:44:11	Log-Likelihood:	-8.6249e+05
No. Observations:	278489	AIC:	1.725e+06
Df Residuals:	278475	BIC:	1.725e+06
Df Model:	13		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	22.3593	0.109	205.702	0.000	22.146	22.572
Available_Extra_Rooms_in_Hospital	0.0469	0.004	12.368	0.000	0.039	0.054
Department	-21.8258	0.078	-281.318	0.000	-21.978	-21.674
Ward_Facility_Code	-0.1793	0.006	-28.155	0.000	-0.192	-0.167
doctor_name	-0.5311	0.174	-3.055	0.002	-0.872	-0.190
staff_available	0.0021	0.003	0.654	0.513	-0.004	0.008
gender	-0.9067	0.031	-29.547	0.000	-0.967	-0.847
Type_of_Admission	0.7838	0.052	14.954	0.000	0.681	0.886
Severity_of_Illness	0.0276	0.016	1.768	0.077	-0.003	0.058
health_conditions	1.6334	0.235	6.941	0.000	1.172	2.095
Visitors_with_Patient	-0.0474	0.005	-10.355	0.000	-0.056	-0.038
Insurance	0.0230	0.024	0.941	0.347	-0.025	0.071
Admission_Deposit	0.0001	9.73e-06	11.749	0.000	9.53e-05	0.000
Age_Group	1.0533	0.018	57.105	0.000	1.017	1.089

Omnibus:	24899.602	Durbin-Watson:	2.001
Prob(Omnibus):	0.000	Jarque-Bera (JB):	119254.826
Skew:	-0.316	Prob(JB):	0.00
Kurtosis:	6.143	Cond. No.	1.26e+05

INFERENCE :

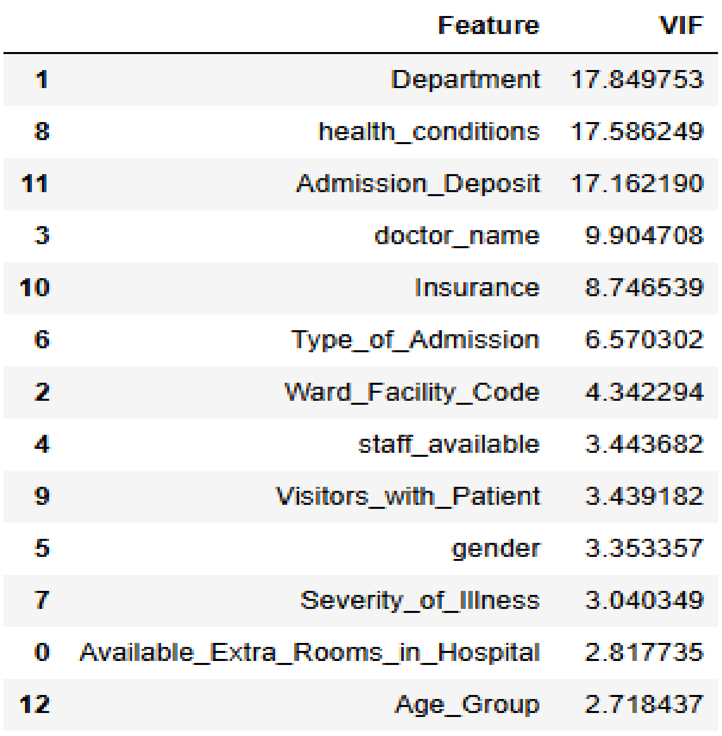
The OLS regression results indicate that the model explains about 54.2% of the variance in the dependent variable, Stay_in_days (R-squared = 0.542).

Several factors significantly influence the length of stay, including Department, Gender, Type_of_Admission, Health_conditions, and Age_Group, all of which have strong p-values (below 0.05). Doctor_name, Visitors_with_Patient, and Admission_Deposit also show significant effects. Some predictors, such as staff_available and Insurance, are not statistically significant at the 5% level.

The model is overall significant ($p < 0.05$) with a high F-statistic, suggesting that the predictors combined have a meaningful impact on the outcome. However, there is evidence of skewness and excess kurtosis, which could suggest potential violations of normality assumptions.

CHECKING MULTICOLLINEARITY USING VARIANCE INFLATION FACTOR:

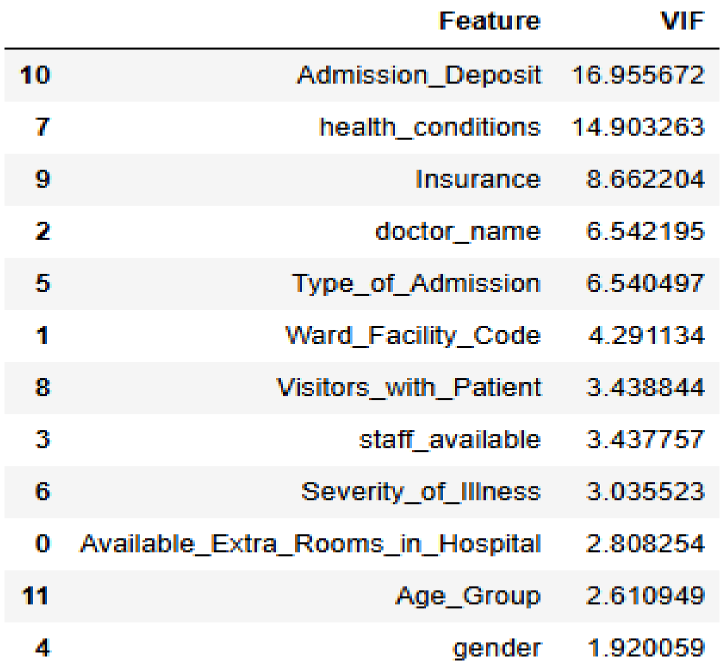
Checking the relationship among the independent variables for possible multicollinearity.



	Feature	VIF
1	Department	17.849753
8	health_conditions	17.586249
11	Admission_Deposit	17.162190
3	doctor_name	9.904708
10	Insurance	8.746539
6	Type_of_Admission	6.570302
2	Ward_Facility_Code	4.342294
4	staff_available	3.443682
9	Visitors_with_Patient	3.439182
5	gender	3.353357
7	Severity_of_Illness	3.040349
0	Available_Extra_Rooms_in_Hospital	2.817735
12	Age_Group	2.718437

KEY POINTS:

- Features like Department, health_conditions, and Admission_Deposit show high multicollinearity (VIF > 10), indicating redundancy.
- Moderate multicollinearity is seen in doctor_name and Insurance, with VIF values near 10.
- Features like Ward_Facility_Code and Severity_of_Illness have manageable VIF values (< 5), posing less concern.
- Minimal multicollinearity is observed in Available_Extra_Rooms_in_Hospital and Age_Group (VIF ~ 2.7).
- High-VIF features may need removal, transformation, or regularization to improve model performance



	Feature	VIF
10	Admission_Deposit	16.955672
7	health_conditions	14.903263
9	Insurance	8.662204
2	doctor_name	6.542195
5	Type_of_Admission	6.540497
1	Ward_Facility_Code	4.291134
8	Visitors_with_Patient	3.438844
3	staff_available	3.437757
6	Severity_of_Illness	3.035523
0	Available_Extra_Rooms_in_Hospital	2.808254
11	Age_Group	2.610949
4	gender	1.920059

We removed the "Department" column due to redundancy with the "Doctor name" column, as both were highly correlated. Redundancy occurs when multiple features provide similar information, leading to unnecessary repetition in the model.

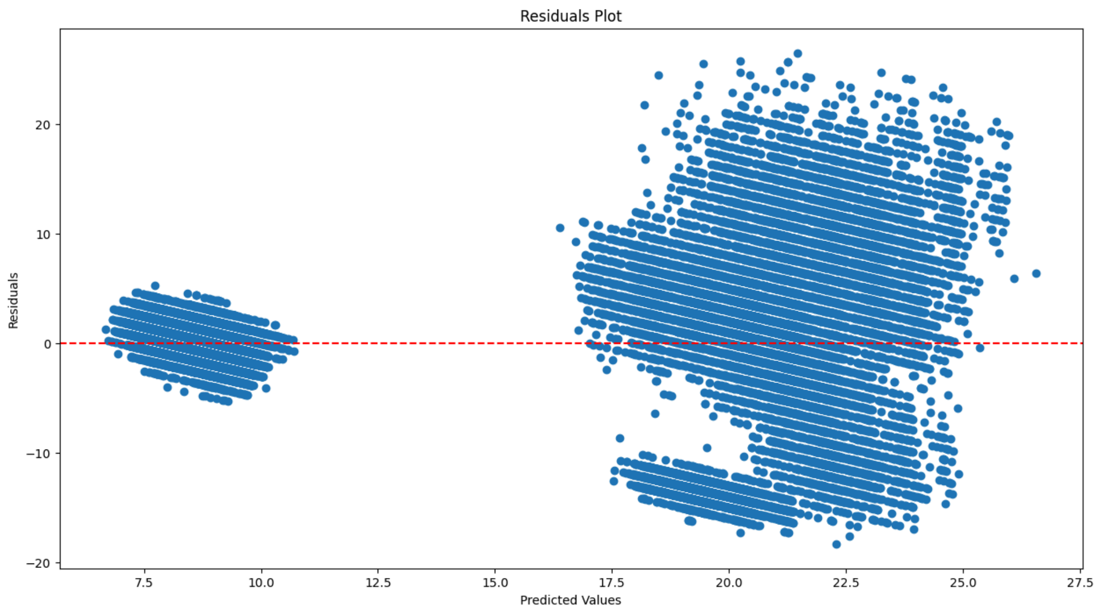
.

LINEAR REGRESSION:

After performing VIF analysis, we proceeded with the linear regression model using the selected features.

Linear regression is a baseline model used to assess the performance of more complex models. It helps to understand the dataset's linear relationships and provides a starting point for comparison.

Model	R-squared Train	R-squared Test	MSE	RMSE	MAE	MAPE
LinearRegression()	0.541501	0.544529	28.459977	5.334789	3.123672	0.303684



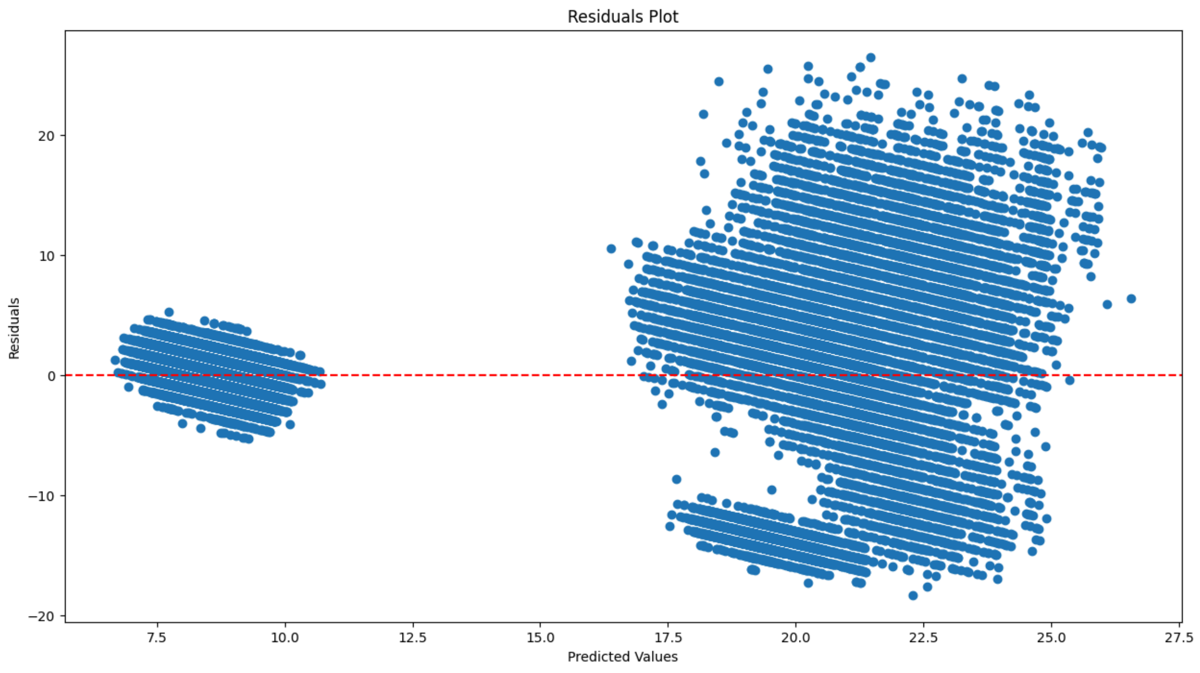
INFERENCE:

Linear regression shows a balanced but modest performance, with R-squared Train and Test around 0.54. While simple, it struggles with non-linear patterns, reflected in higher error metrics like MSE (28.45) and RMSE (5.33).

RIDGE:

Ridge regression adds regularization to linear regression, addressing overfitting by penalizing large coefficients. It is ideal for datasets with multicollinearity.

	Model	R-squared Train	R-squared Test	MSE	RMSE	MAE	MAPE
0	LinearRegression()	0.541501	0.544529	28.459977	5.334789	3.123672	0.303684
1	Ridge(alpha=1, max_iter=500)	0.541501	0.544529	28.459992	5.334791	3.123720	0.303686



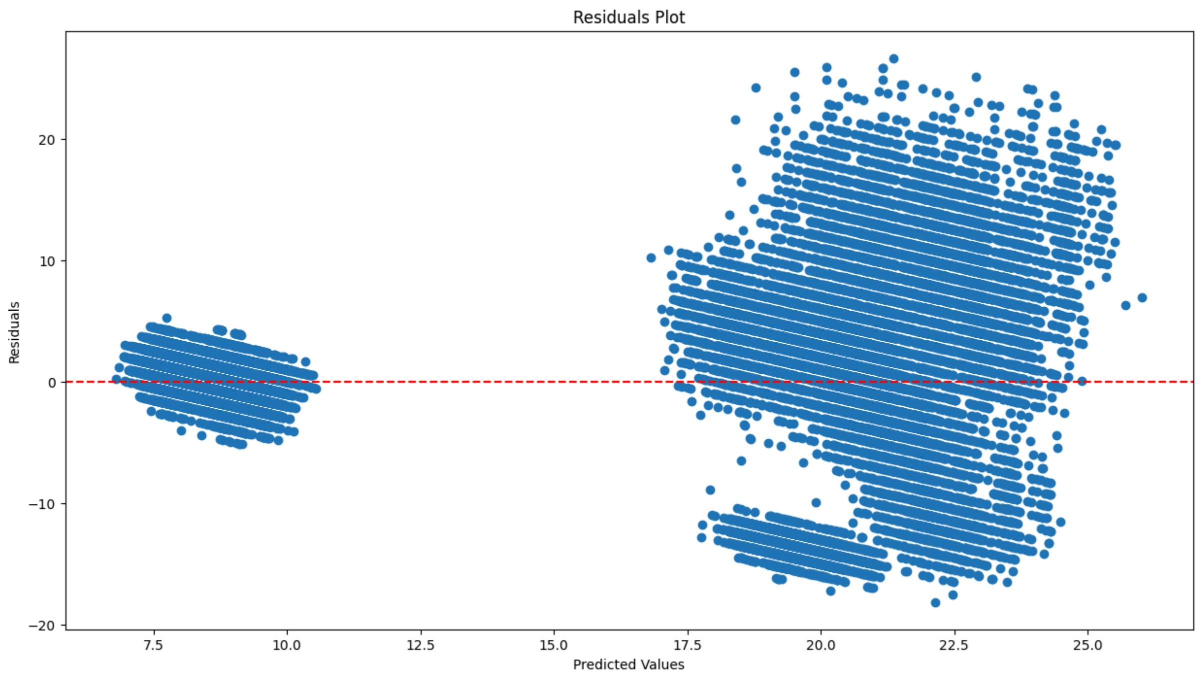
INFERENCE:

Ridge regression provides identical performance to linear regression, with an R-squared Test of 0.5445 and RMSE of 5.33. The regularization doesn't show significant improvement, suggesting minimal multicollinearity in the dataset.

LASSO:

Lasso regression performs both regularization and feature selection by shrinking insignificant coefficients to zero, aiding in simplifying the model.

	Model	R-squared Train	R-squared Test	MSE	RMSE	MAE	MAPE
0	LinearRegression()	0.541501	0.544529	28.459977	5.334789	3.123672	0.303684
1	Ridge(alpha=1, max_iter=500)	0.541501	0.544529	28.459992	5.334791	3.123720	0.303686
2	Lasso(alpha=0.01, max_iter=500)	0.541177	0.544197	28.480752	5.336736	3.122036	0.303056



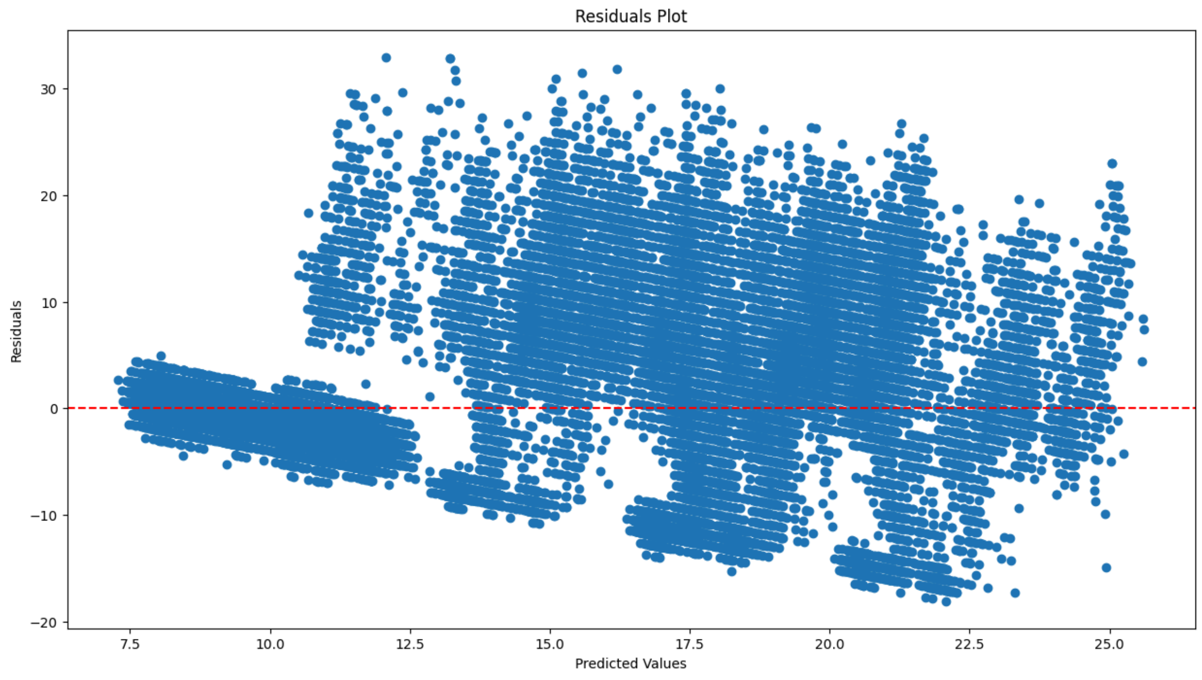
INFERENCE:

Lasso regression offers similar performance to linear and ridge regression, with an R-squared Test of 0.5441 and RMSE of 5.33. This indicates that all features are contributing and no significant simplification occurred.

ELASTIC NET:

ElasticNet combines the benefits of both Lasso and Ridge regression, handling multicollinearity and irrelevant features effectively in datasets.

	Model	R-squared Train	R-squared Test	MSE	RMSE	MAE	MAPE
0	LinearRegression()	0.541501	0.544529	28.459977	5.334789	3.123672	0.303684
1	Ridge(alpha=1, max_iter=500)	0.541501	0.544529	28.459992	5.334791	3.123720	0.303686
2	Lasso(alpha=0.01, max_iter=500)	0.541177	0.544197	28.480752	5.336736	3.122036	0.303056
3	ElasticNet(alpha=0.1, l1_ratio=0.01, max_iter=...	0.439379	0.441277	34.911691	5.908612	3.778581	0.342459



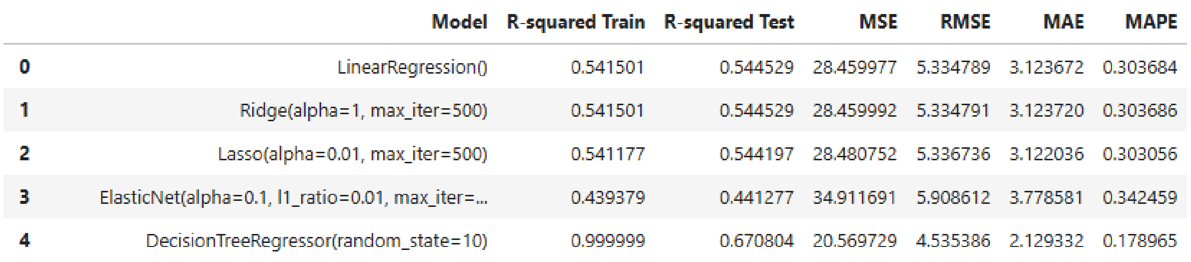
INFERENCE:

ElasticNet shows worse performance (R-squared Test: 0.4412, RMSE: 5.90) than simpler linear models, suggesting that its combined penalties may not suit this dataset's structure.

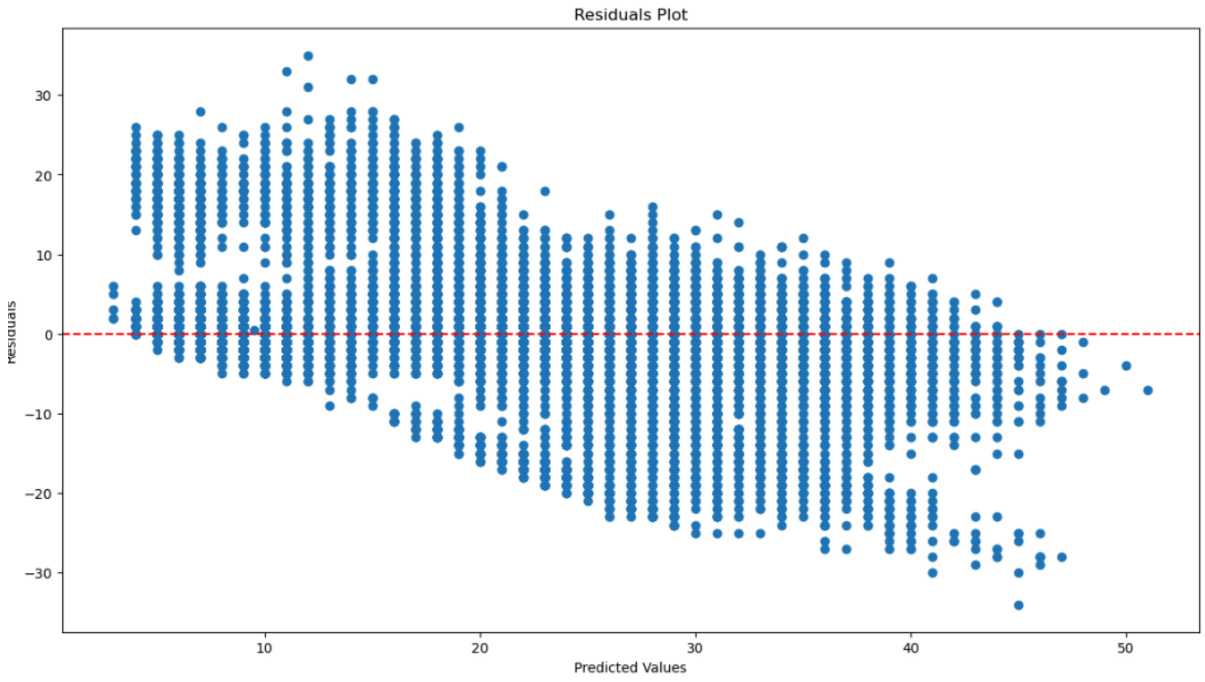
DECISION TREE REGRESSOR:

We chose the Decision Tree Regressor because it can model non-linear relationships between features and the target variable, allowing for more flexible predictions. It also handles both numerical and categorical data, is easy to interpret, and does not require feature scaling.

Decision trees can capture complex, non-linear relationships and provide an intuitive structure for data interpretation.



	Model	R-squared Train	R-squared Test	MSE	RMSE	MAE	MAPE
0	LinearRegression()	0.541501	0.544529	28.459977	5.334789	3.123672	0.303684
1	Ridge(alpha=1, max_iter=500)	0.541501	0.544529	28.459992	5.334791	3.123720	0.303686
2	Lasso(alpha=0.01, max_iter=500)	0.541177	0.544197	28.480752	5.336736	3.122036	0.303056
3	ElasticNet(alpha=0.1, l1_ratio=0.01, max_iter=...	0.439379	0.441277	34.911691	5.908612	3.778581	0.342459
4	DecisionTreeRegressor(random_state=10)	0.999999	0.670804	20.569729	4.535386	2.129332	0.178965

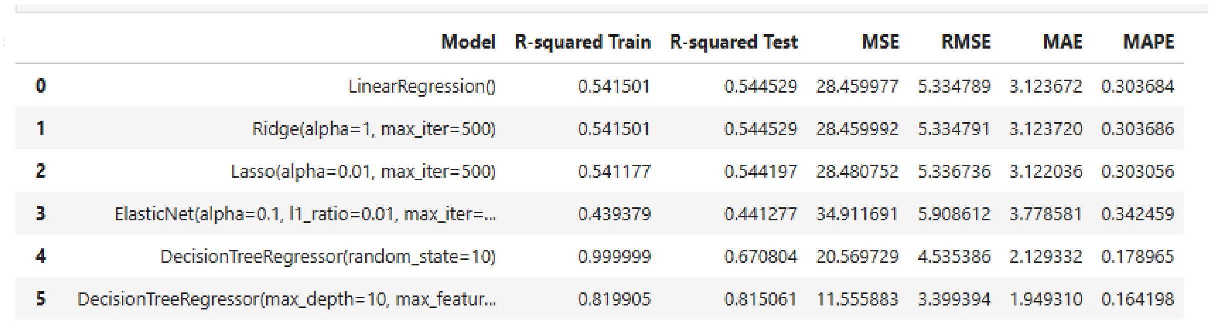


INFERENCE:

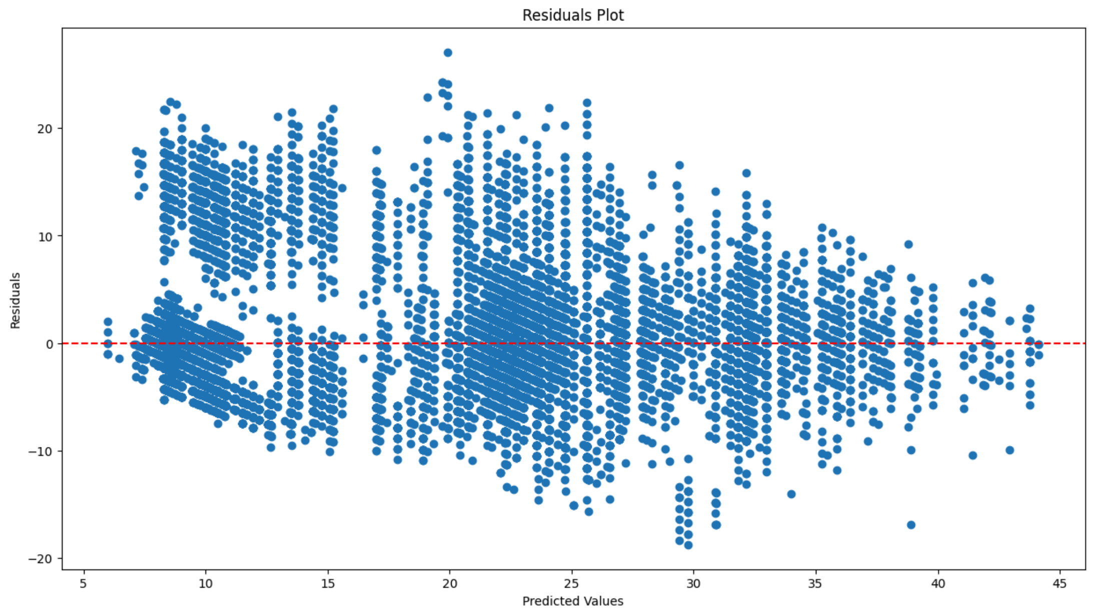
The model overfits the training data (**R-squared Train**: 1.0) but performs poorly on the test set (**R-squared Test**: 0.6708). The high variance and moderate errors highlight the need for tuning.

HYPER PARAMETER TUNING - DECISION TREE:

Tuning the decision tree's depth and features reduces overfitting, improving its ability to generalize to unseen data.



	Model	R-squared Train	R-squared Test	MSE	RMSE	MAE	MAPE
0	LinearRegression()	0.541501	0.544529	28.459977	5.334789	3.123672	0.303684
1	Ridge(alpha=1, max_iter=500)	0.541501	0.544529	28.459992	5.334791	3.123720	0.303686
2	Lasso(alpha=0.01, max_iter=500)	0.541177	0.544197	28.480752	5.336736	3.122036	0.303056
3	ElasticNet(alpha=0.1, l1_ratio=0.01, max_iter=...	0.439379	0.441277	34.911691	5.908612	3.778581	0.342459
4	DecisionTreeRegressor(random_state=10)	0.999999	0.670804	20.569729	4.535386	2.129332	0.178965
5	DecisionTreeRegressor(max_depth=10, max_featur...	0.819905	0.815061	11.555883	3.399394	1.949310	0.164198



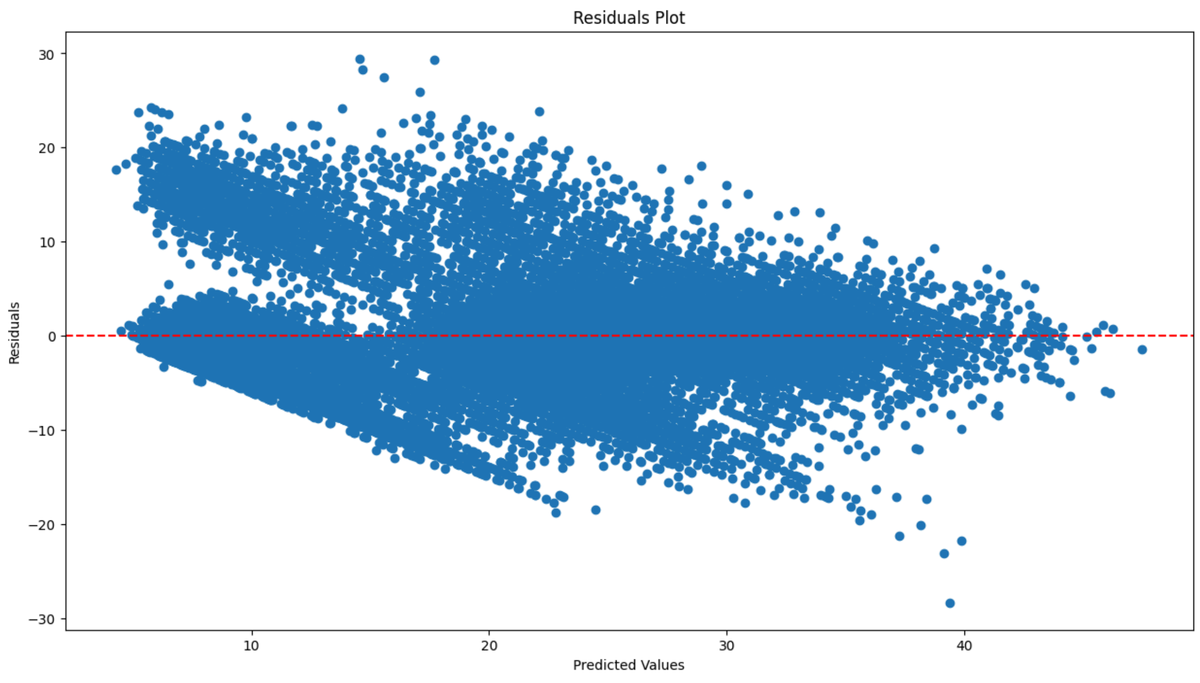
INFERENCE:

- With limited depth and no constraint on features, this DecisionTreeRegressor strikes a better balance between fitting and generalization. It outperforms the above model in all test metrics and appears to be more reliable for predictions.
- This configuration shows better generalization (R-squared Test: 0.8151), with lower errors (MSE: 11.55, RMSE: 3.39) than the untuned version. It is more reliable for predictions.

RANDOMFOREST REGRESSOR:

Random forests average multiple decision trees, reducing overfitting and capturing complex patterns in the data.

	Model	R-squared Train	R-squared Test	MSE	RMSE	MAE	MAPE
0	LinearRegression()	0.541501	0.544529	28.459977	5.334789	3.123672	0.303684
1	Ridge(alpha=1, max_iter=500)	0.541501	0.544529	28.459992	5.334791	3.123720	0.303686
2	Lasso(alpha=0.01, max_iter=500)	0.541177	0.544197	28.480752	5.336736	3.122036	0.303056
3	ElasticNet(alpha=0.1, l1_ratio=0.01, max_iter=...	0.439379	0.441277	34.911691	5.908612	3.778581	0.342459
4	DecisionTreeRegressor(random_state=10)	0.999999	0.670804	20.569729	4.535386	2.129332	0.178965
5	DecisionTreeRegressor(max_depth=10, max_featur...	0.819905	0.815061	11.555883	3.399394	1.949310	0.164198
6	RandomForestRegressor(random_state=10)	0.973769	0.812438	11.719755	3.423413	1.789791	0.152407



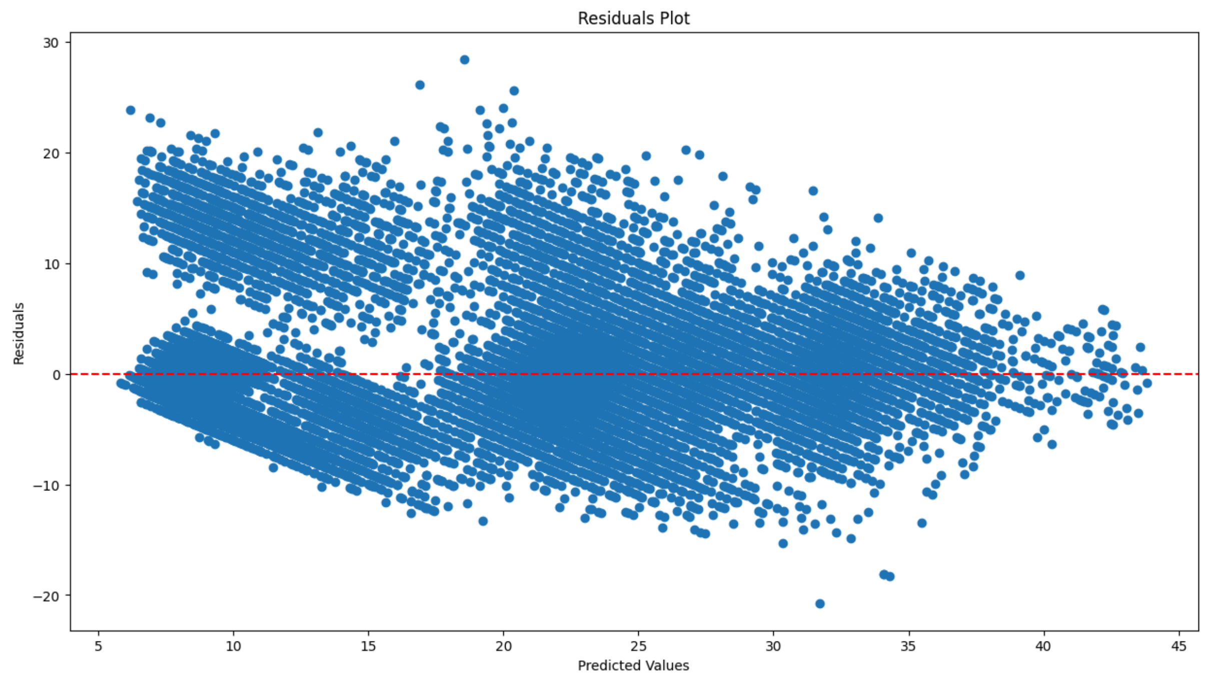
INFERENCE:

The model performs exceptionally well (R-squared Test: 0.8124, RMSE: 3.42), balancing bias and variance effectively while delivering robust predictions.

HYPER PARAMETER TUNING- RANDOMFOREST REGRESSOR

Hyperparameter tuning enhances the model's performance by optimizing feature selection and tree configuration.

	Model	R-squared Train	R-squared Test	MSE	RMSE	MAE	MAPE
0	LinearRegression()	0.541501	0.544529	28.459977	5.334789	3.123672	0.303684
1	Ridge(alpha=1, max_iter=500)	0.541501	0.544529	28.459992	5.334791	3.123720	0.303686
2	Lasso(alpha=0.01, max_iter=500)	0.541177	0.544197	28.480752	5.336736	3.122036	0.303056
3	ElasticNet(alpha=0.1, l1_ratio=0.01, max_iter=...	0.439379	0.441277	34.911691	5.908612	3.778581	0.342459
4	DecisionTreeRegressor(random_state=10)	0.999999	0.670804	20.569729	4.535386	2.129332	0.178965
5	DecisionTreeRegressor(max_depth=10, max_featur...	0.819905	0.815061	11.555883	3.399394	1.949310	0.164198
6	RandomForestRegressor(random_state=10)	0.973769	0.812438	11.719755	3.423413	1.789791	0.152407
7	RandomForestRegressor(max_features='sqrt', min...	0.876472	0.824635	10.957657	3.310235	1.817514	0.153362



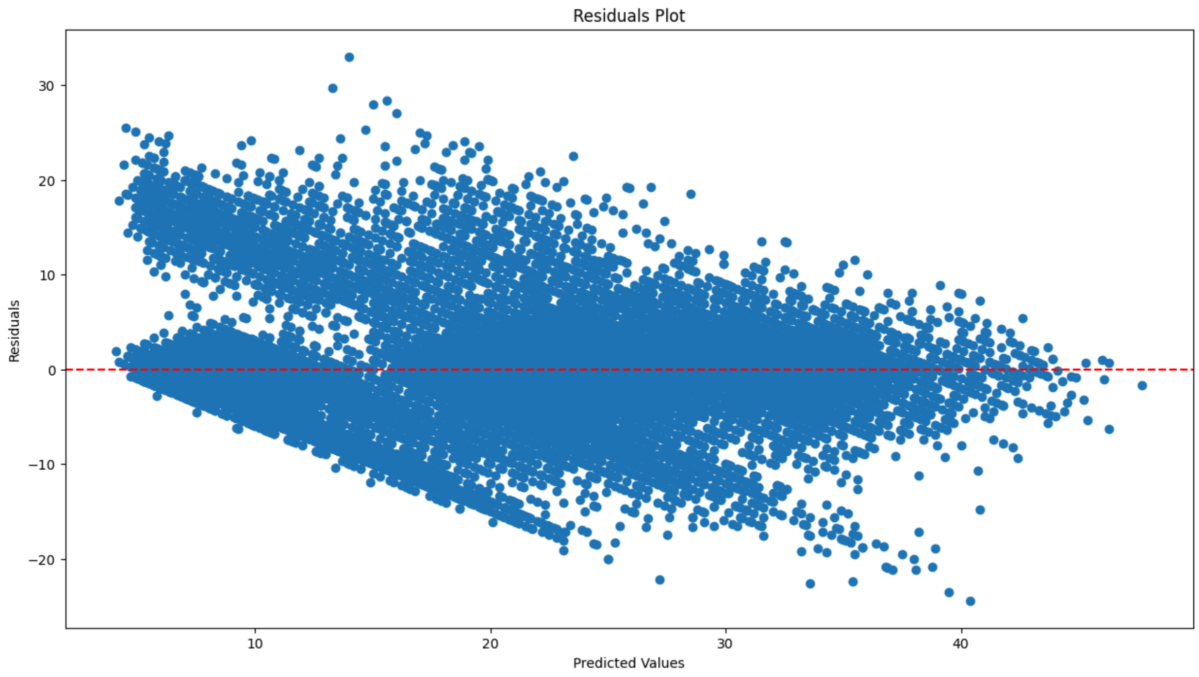
INFERENCE:

The tuned random forest improves slightly (R-squared Test: 0.8264), with reduced error metrics (RMSE: 3.30). It demonstrates excellent generalization and predictive accuracy.

BAGGING REGRESSOR:

Bagging reduces variance by combining multiple models, making it effective for handling overfitting and noisy data.

	Model	R-squared Train	R-squared Test	MSE	RMSE	MAE	MAPE
0	LinearRegression()	0.541501	0.544529	28.459977	5.334789	3.123672	0.303684
1	Ridge(alpha=1, max_iter=500)	0.541501	0.544529	28.459992	5.334791	3.123720	0.303686
2	Lasso(alpha=0.01, max_iter=500)	0.541177	0.544197	28.480752	5.336736	3.122036	0.303056
3	ElasticNet(alpha=0.1, l1_ratio=0.01, max_iter=...	0.439379	0.441277	34.911691	5.908612	3.778581	0.342459
4	DecisionTreeRegressor(random_state=10)	0.999999	0.670804	20.569729	4.535386	2.129332	0.178965
5	DecisionTreeRegressor(max_depth=10, max_featur...	0.819905	0.815061	11.555883	3.399394	1.949310	0.164198
6	RandomForestRegressor(random_state=10)	0.973769	0.812438	11.719755	3.423413	1.789791	0.152407
7	RandomForestRegressor(max_features='sqrt', min...	0.876472	0.824635	10.957657	3.310235	1.817514	0.153362
8	BaggingRegressor(random_state=10)	0.964796	0.800046	12.494108	3.534700	1.831221	0.155978



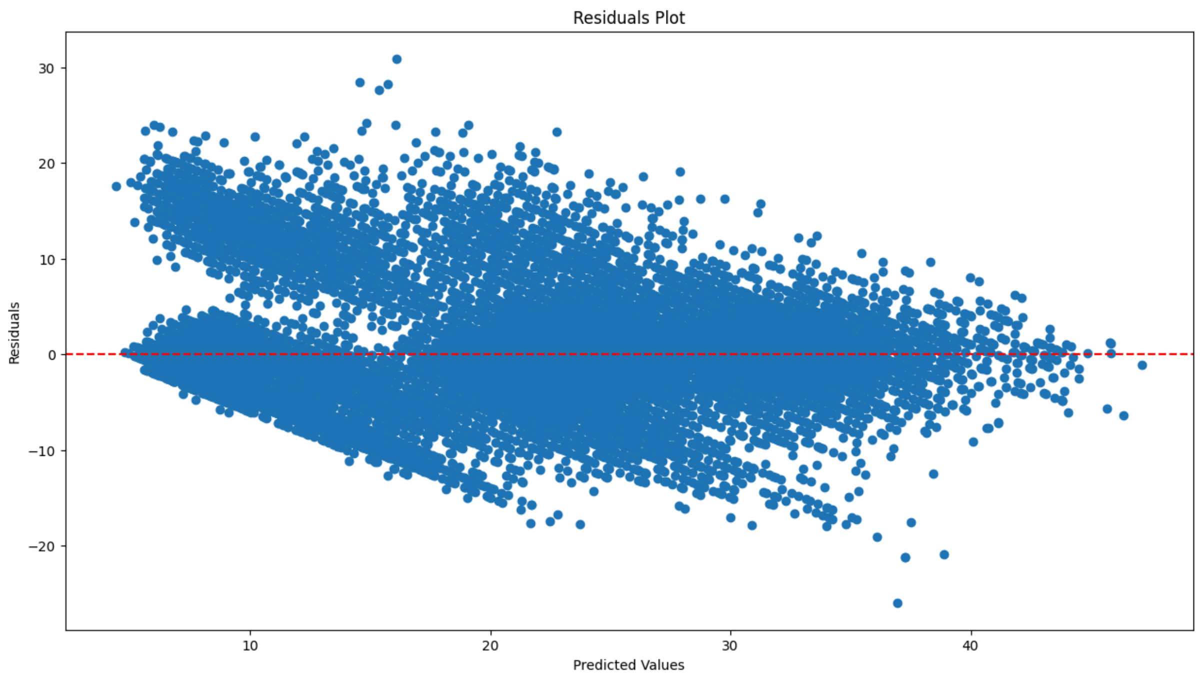
INFERENCE:

The model performs well (R-squared Test: 0.8000), with low errors (RMSE: 3.53). However, it slightly underperforms compared to the tuned random forest.

HYPER PARAMETER TUNING- BAGGING REGRESSOR:

Tuning bagging hyperparameters improves model stability and prediction accuracy by controlling sampling strategies.

	Model	R-squared Train	R-squared Test	MSE	RMSE	MAE	MAPE
0	LinearRegression()	0.541501	0.544529	28.459977	5.334789	3.123672	0.303684
1	Ridge(alpha=1, max_iter=500)	0.541501	0.544529	28.459992	5.334791	3.123720	0.303686
2	Lasso(alpha=0.01, max_iter=500)	0.541177	0.544197	28.480752	5.336736	3.122036	0.303056
3	ElasticNet(alpha=0.1, l1_ratio=0.01, max_iter=...	0.439379	0.441277	34.911691	5.908612	3.778581	0.342459
4	DecisionTreeRegressor(random_state=10)	0.999999	0.670804	20.569729	4.535386	2.129332	0.178965
5	DecisionTreeRegressor(max_depth=10, max_featur...	0.819905	0.815061	11.555883	3.399394	1.949310	0.164198
6	RandomForestRegressor(random_state=10)	0.973769	0.812438	11.719755	3.423413	1.789791	0.152407
7	RandomForestRegressor(max_features='sqrt', min...	0.876472	0.824635	10.957657	3.310235	1.817514	0.153362
8	BaggingRegressor(random_state=10)	0.964796	0.800046	12.494108	3.534700	1.831221	0.155978
9	BaggingRegressor(bootstrap=False, max_samples=...	0.953381	0.817469	11.405384	3.377186	1.781295	0.151793



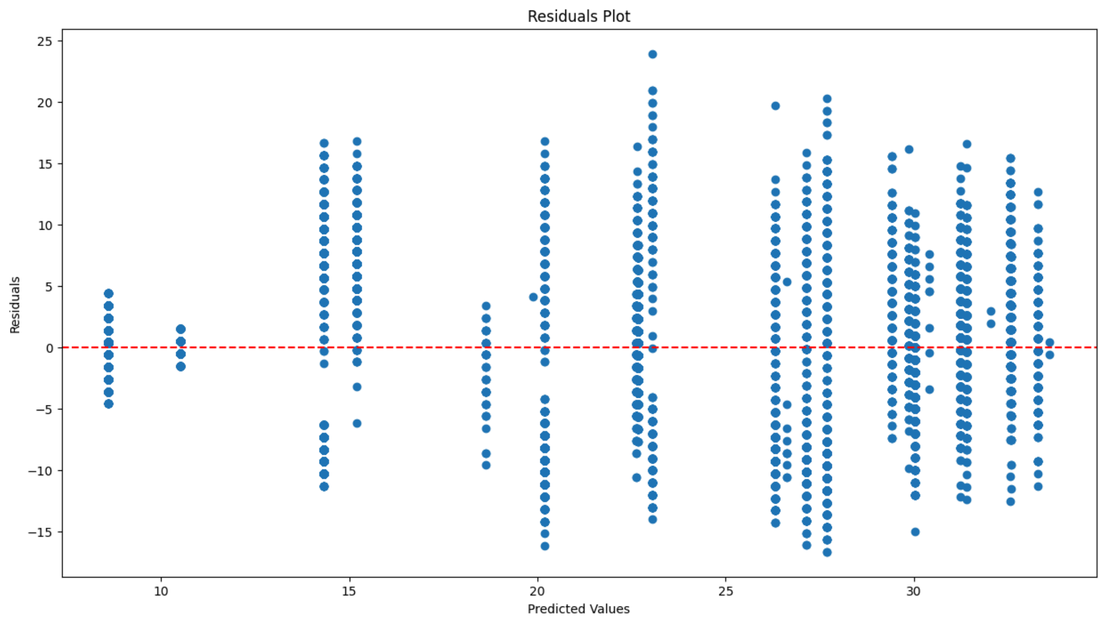
INFERENCE:

The tuned bagging regressor shows better performance (R-squared Test: 0.8175, RMSE: 3.37), making it competitive with random forests.

ADABOOST REGRESSOR:

AdaBoost focuses on minimizing errors iteratively, improving weak learners' accuracy through boosting techniques.

	Model	R-squared Train	R-squared Test	MSE	RMSE	MAE	MAPE
0	LinearRegression()	0.541501	0.544529	28.459977	5.334789	3.123672	0.303684
1	Ridge(alpha=1, max_iter=500)	0.541501	0.544529	28.459992	5.334791	3.123720	0.303686
2	Lasso(alpha=0.01, max_iter=500)	0.541177	0.544197	28.480752	5.336736	3.122036	0.303056
3	ElasticNet(alpha=0.1, l1_ratio=0.01, max_iter=...	0.439379	0.441277	34.911691	5.908612	3.778581	0.342459
4	DecisionTreeRegressor(random_state=10)	0.999999	0.670804	20.569729	4.535386	2.129332	0.178965
5	DecisionTreeRegressor(max_depth=10, max_featur...	0.819905	0.815061	11.555883	3.399394	1.949310	0.164198
6	RandomForestRegressor(random_state=10)	0.973769	0.812438	11.719755	3.423413	1.789791	0.152407
7	RandomForestRegressor(max_features='sqrt', min...	0.876472	0.824635	10.957657	3.310235	1.817514	0.153362
8	BaggingRegressor(random_state=10)	0.964796	0.800046	12.494108	3.534700	1.831221	0.155978
9	BaggingRegressor(bootstrap=False, max_samples=...	0.953381	0.817469	11.405384	3.377186	1.781295	0.151793
10	AdaBoostRegressor()	0.724672	0.724803	17.195612	4.146759	2.535948	0.235850

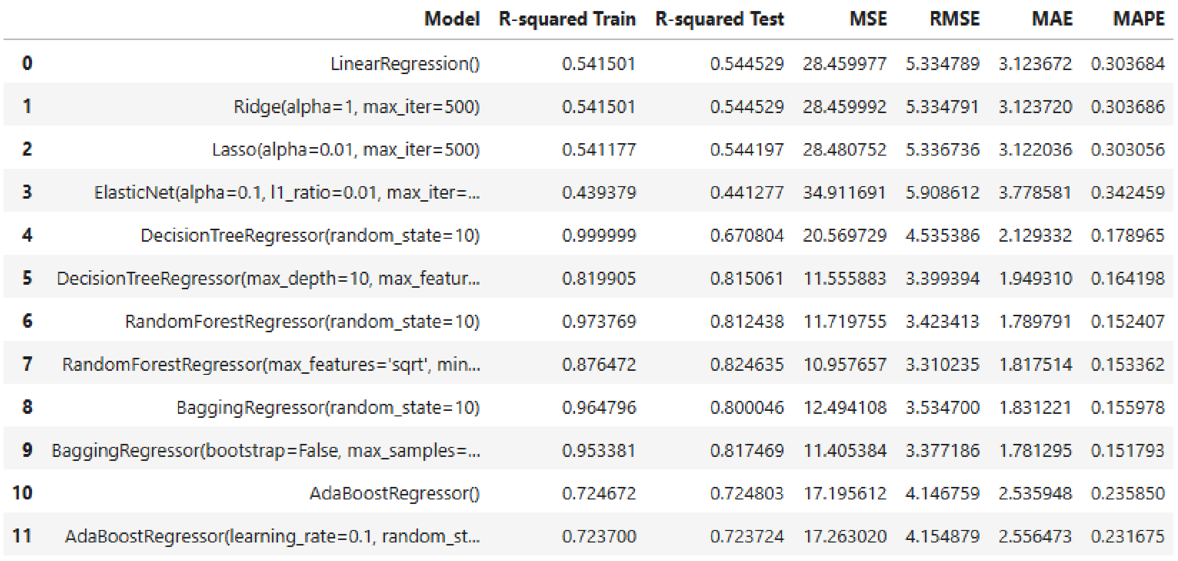


INFERENCE:

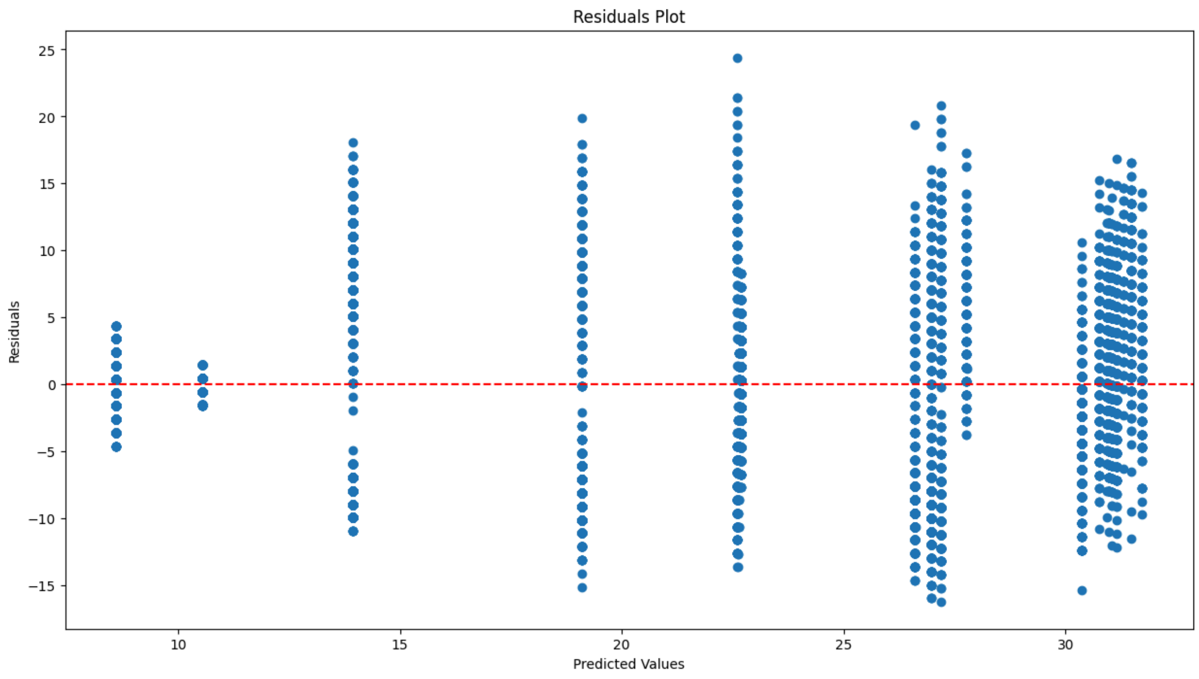
AdaBoost shows moderate performance (R-squared Test: 0.7248, RMSE: 4.14). It performs worse than random forests and bagging models, with slightly higher errors.

HYPER PARAMETER TUNING- ADABOOSTREGRESSOR:

Tuning AdaBoost improves its learning rate and handling of complex patterns, potentially reducing errors.



	Model	R-squared Train	R-squared Test	MSE	RMSE	MAE	MAPE
0	LinearRegression()	0.541501	0.544529	28.459977	5.334789	3.123672	0.303684
1	Ridge(alpha=1, max_iter=500)	0.541501	0.544529	28.459992	5.334791	3.123720	0.303686
2	Lasso(alpha=0.01, max_iter=500)	0.541177	0.544197	28.480752	5.336736	3.122036	0.303056
3	ElasticNet(alpha=0.1, l1_ratio=0.01, max_iter=...	0.439379	0.441277	34.911691	5.908612	3.778581	0.342459
4	DecisionTreeRegressor(random_state=10)	0.999999	0.670804	20.569729	4.535386	2.129332	0.178965
5	DecisionTreeRegressor(max_depth=10, max_featur...	0.819905	0.815061	11.555883	3.399394	1.949310	0.164198
6	RandomForestRegressor(random_state=10)	0.973769	0.812438	11.719755	3.423413	1.789791	0.152407
7	RandomForestRegressor(max_features='sqrt', min...	0.876472	0.824635	10.957657	3.310235	1.817514	0.153362
8	BaggingRegressor(random_state=10)	0.964796	0.800046	12.494108	3.534700	1.831221	0.155978
9	BaggingRegressor(bootstrap=False, max_samples=...	0.953381	0.817469	11.405384	3.377186	1.781295	0.151793
10	AdaBoostRegressor()	0.724672	0.724803	17.195612	4.146759	2.535948	0.235850
11	AdaBoostRegressor(learning_rate=0.1, random_st...	0.723700	0.723724	17.263020	4.154879	2.556473	0.231675

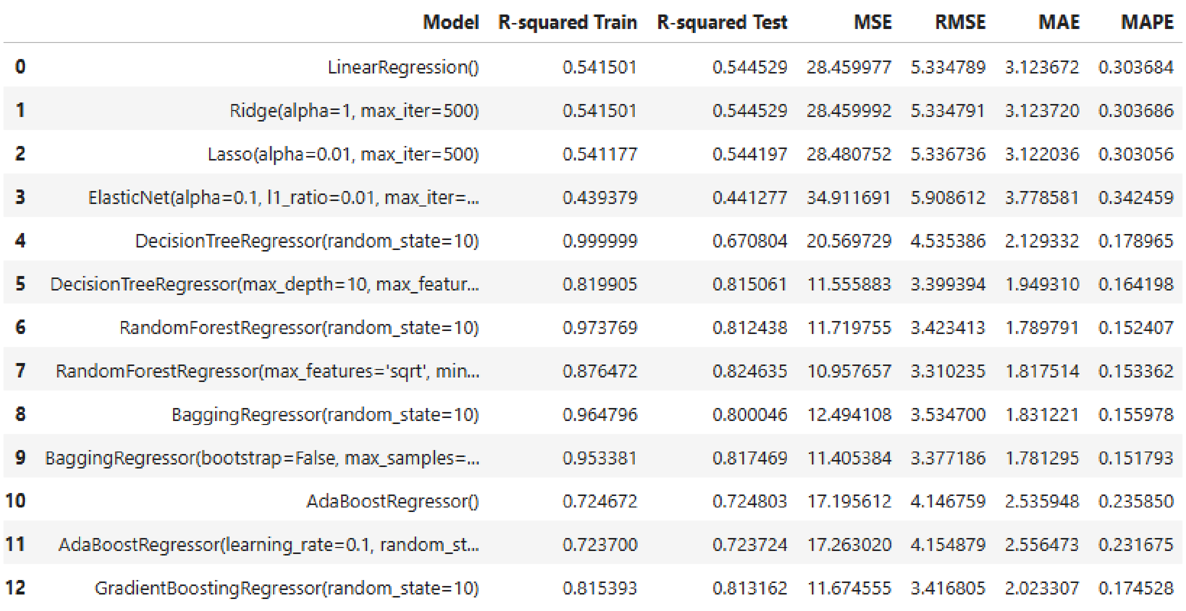


INFERENCE:

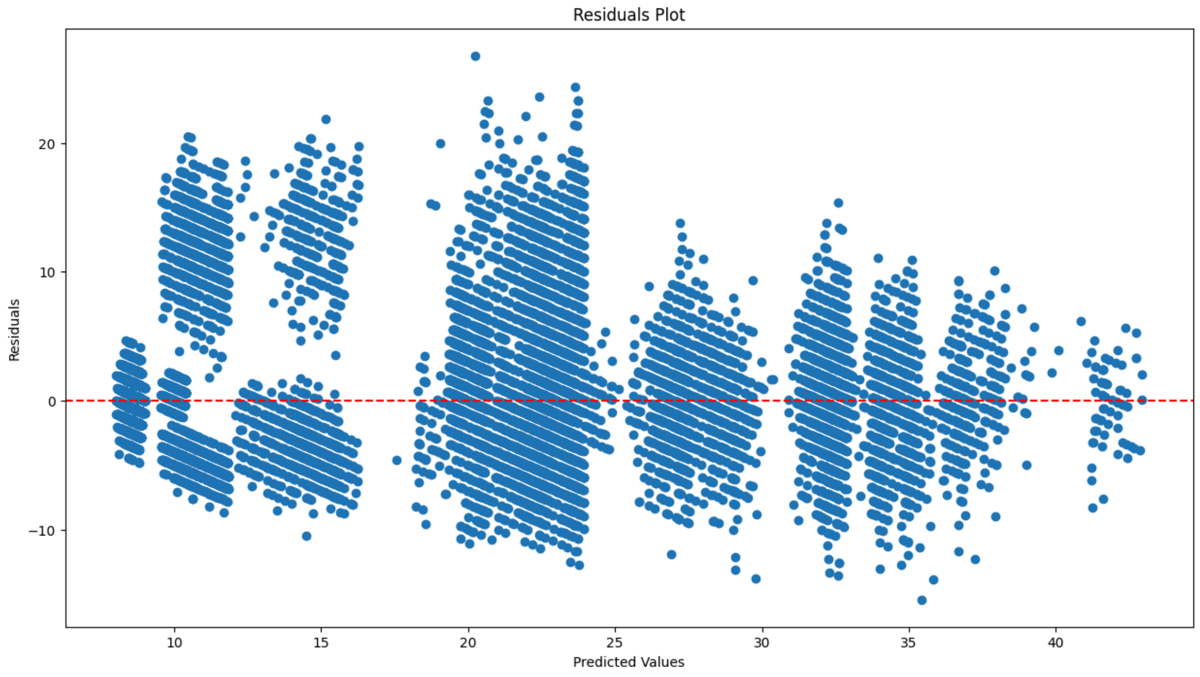
The tuned version achieves similar results (R-squared Test: 0.7237), with slightly higher RMSE (4.15). It doesn't outperform simpler ensemble methods.

GRADIENTBOOST REGRESSOR:

Gradient boosting sequentially improves weak learners, making it effective for capturing non-linear relationships in data.



	Model	R-squared Train	R-squared Test	MSE	RMSE	MAE	MAPE
0	LinearRegression()	0.541501	0.544529	28.459977	5.334789	3.123672	0.303684
1	Ridge(alpha=1, max_iter=500)	0.541501	0.544529	28.459992	5.334791	3.123720	0.303686
2	Lasso(alpha=0.01, max_iter=500)	0.541177	0.544197	28.480752	5.336736	3.122036	0.303056
3	ElasticNet(alpha=0.1, l1_ratio=0.01, max_iter=...	0.439379	0.441277	34.911691	5.908612	3.778581	0.342459
4	DecisionTreeRegressor(random_state=10)	0.999999	0.670804	20.569729	4.535386	2.129332	0.178965
5	DecisionTreeRegressor(max_depth=10, max_featur...	0.819905	0.815061	11.555883	3.399394	1.949310	0.164198
6	RandomForestRegressor(random_state=10)	0.973769	0.812438	11.719755	3.423413	1.789791	0.152407
7	RandomForestRegressor(max_features='sqrt', min...	0.876472	0.824635	10.957657	3.310235	1.817514	0.153362
8	BaggingRegressor(random_state=10)	0.964796	0.800046	12.494108	3.534700	1.831221	0.155978
9	BaggingRegressor(bootstrap=False, max_samples=...	0.953381	0.817469	11.405384	3.377186	1.781295	0.151793
10	AdaBoostRegressor()	0.724672	0.724803	17.195612	4.146759	2.535948	0.235850
11	AdaBoostRegressor(learning_rate=0.1, random_st...	0.723700	0.723724	17.263020	4.154879	2.556473	0.231675
12	GradientBoostingRegressor(random_state=10)	0.815393	0.813162	11.674555	3.416805	2.023307	0.174528

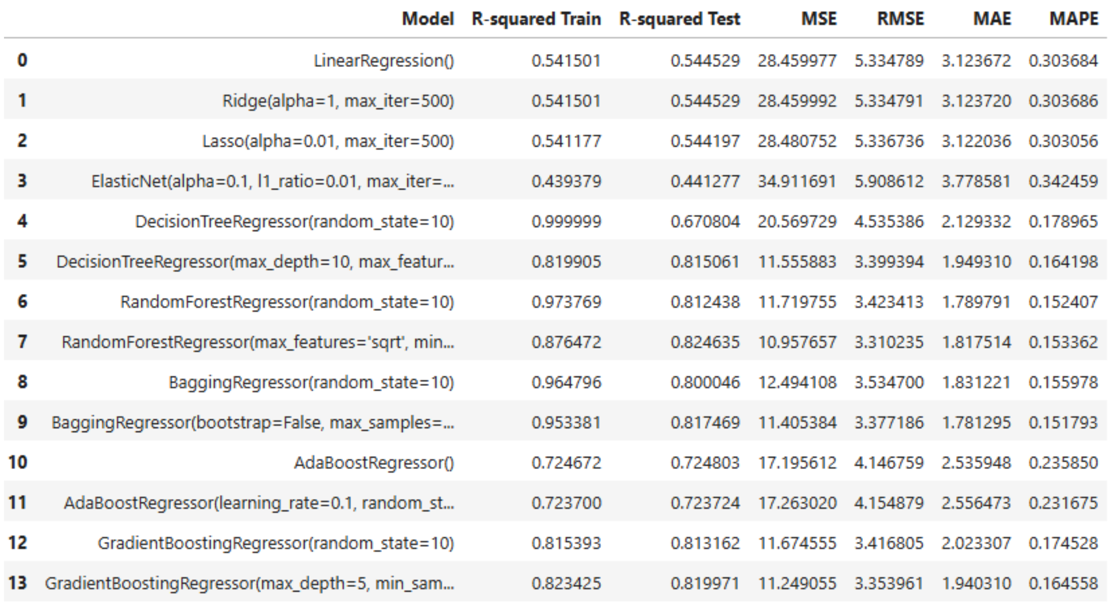


INFERENCE:

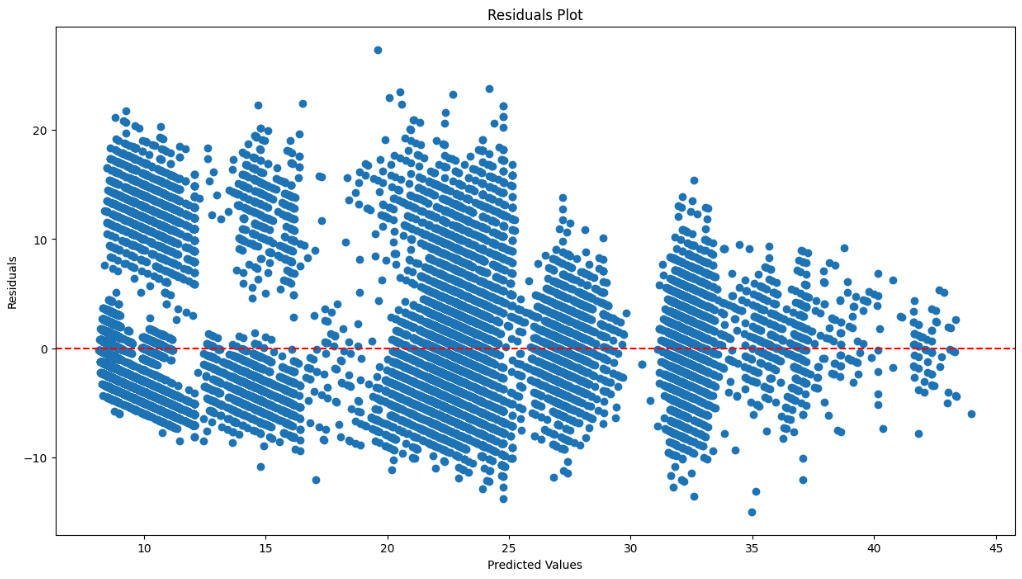
Gradient boosting performs well (R-squared Test: 0.8131, RMSE: 3.41), with competitive errors and good generalization, slightly underperforming compared to tuned random forests.

HYPER PARAMETER TUNING- GRADIENTBOOST REGRESSOR:

Hyperparameter tuning enhances gradient boosting efficiency and accuracy by optimizing tree depth and learning rate.



	Model	R-squared Train	R-squared Test	MSE	RMSE	MAE	MAPE
0	LinearRegression()	0.541501	0.544529	28.459977	5.334789	3.123672	0.303684
1	Ridge(alpha=1, max_iter=500)	0.541501	0.544529	28.459992	5.334791	3.123720	0.303686
2	Lasso(alpha=0.01, max_iter=500)	0.541177	0.544197	28.480752	5.336736	3.122036	0.303056
3	ElasticNet(alpha=0.1, l1_ratio=0.01, max_iter=...	0.439379	0.441277	34.911691	5.908612	3.778581	0.342459
4	DecisionTreeRegressor(random_state=10)	0.999999	0.670804	20.569729	4.535386	2.129332	0.178965
5	DecisionTreeRegressor(max_depth=10, max_featur...	0.819905	0.815061	11.555883	3.399394	1.949310	0.164198
6	RandomForestRegressor(random_state=10)	0.973769	0.812438	11.719755	3.423413	1.789791	0.152407
7	RandomForestRegressor(max_features='sqrt', min...	0.876472	0.824635	10.957657	3.310235	1.817514	0.153362
8	BaggingRegressor(random_state=10)	0.964796	0.800046	12.494108	3.534700	1.831221	0.155978
9	BaggingRegressor(bootstrap=False, max_samples=...	0.953381	0.817469	11.405384	3.377186	1.781295	0.151793
10	AdaBoostRegressor()	0.724672	0.724803	17.195612	4.146759	2.535948	0.235850
11	AdaBoostRegressor(learning_rate=0.1, random_st...	0.723700	0.723724	17.263020	4.154879	2.556473	0.231675
12	GradientBoostingRegressor(random_state=10)	0.815393	0.813162	11.674555	3.416805	2.023307	0.174528
13	GradientBoostingRegressor(max_depth=5, min_sam...	0.823425	0.819971	11.249055	3.353961	1.940310	0.164558

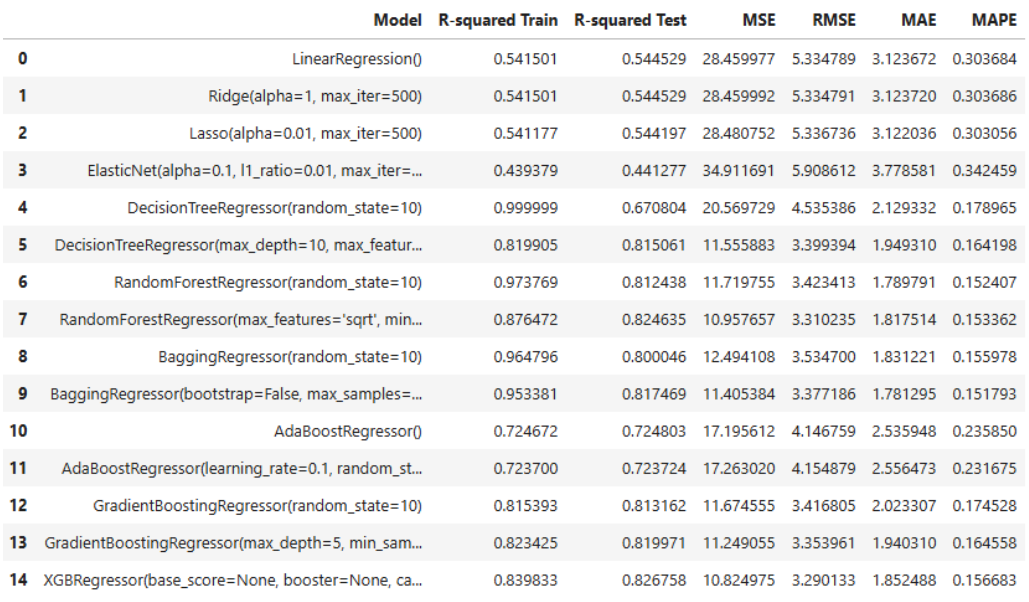


INFERENCE:

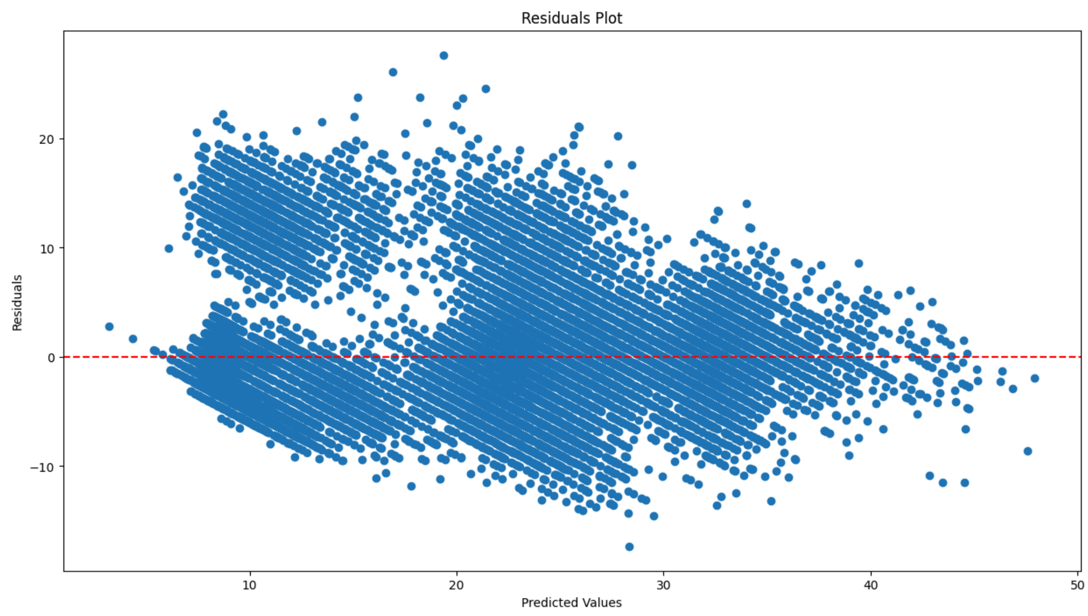
The tuned model performs better (R-squared Test: 0.8197, RMSE: 3.39), showing improved generalization and comparable accuracy to random forests.

XG BOOST:

XGBoost is a powerful, efficient gradient boosting method known for its scalability and high predictive performance.



	Model	R-squared Train	R-squared Test	MSE	RMSE	MAE	MAPE
0	LinearRegression()	0.541501	0.544529	28.459977	5.334789	3.123672	0.303684
1	Ridge(alpha=1, max_iter=500)	0.541501	0.544529	28.459992	5.334791	3.123720	0.303686
2	Lasso(alpha=0.01, max_iter=500)	0.541177	0.544197	28.480752	5.336736	3.122036	0.303056
3	ElasticNet(alpha=0.1, l1_ratio=0.01, max_iter=...	0.439379	0.441277	34.911691	5.908612	3.778581	0.342459
4	DecisionTreeRegressor(random_state=10)	0.999999	0.670804	20.569729	4.535386	2.129332	0.178965
5	DecisionTreeRegressor(max_depth=10, max_featur...	0.819905	0.815061	11.555883	3.399394	1.949310	0.164198
6	RandomForestRegressor(random_state=10)	0.973769	0.812438	11.719755	3.423413	1.789791	0.152407
7	RandomForestRegressor(max_features='sqrt', min...	0.876472	0.824635	10.957657	3.310235	1.817514	0.153362
8	BaggingRegressor(random_state=10)	0.964796	0.800046	12.494108	3.534700	1.831221	0.155978
9	BaggingRegressor(bootstrap=False, max_samples=...	0.953381	0.817469	11.405384	3.377186	1.781295	0.151793
10	AdaBoostRegressor()	0.724672	0.724803	17.195612	4.146759	2.535948	0.235850
11	AdaBoostRegressor(learning_rate=0.1, random_st...	0.723700	0.723724	17.263020	4.154879	2.556473	0.231675
12	GradientBoostingRegressor(random_state=10)	0.815393	0.813162	11.674555	3.416805	2.023307	0.174528
13	GradientBoostingRegressor(max_depth=5, min_sam...	0.823425	0.819971	11.249055	3.353961	1.940310	0.164558
14	XGBRegressor(base_score=None, booster=None, ca...	0.839833	0.826758	10.824975	3.290133	1.852488	0.156683

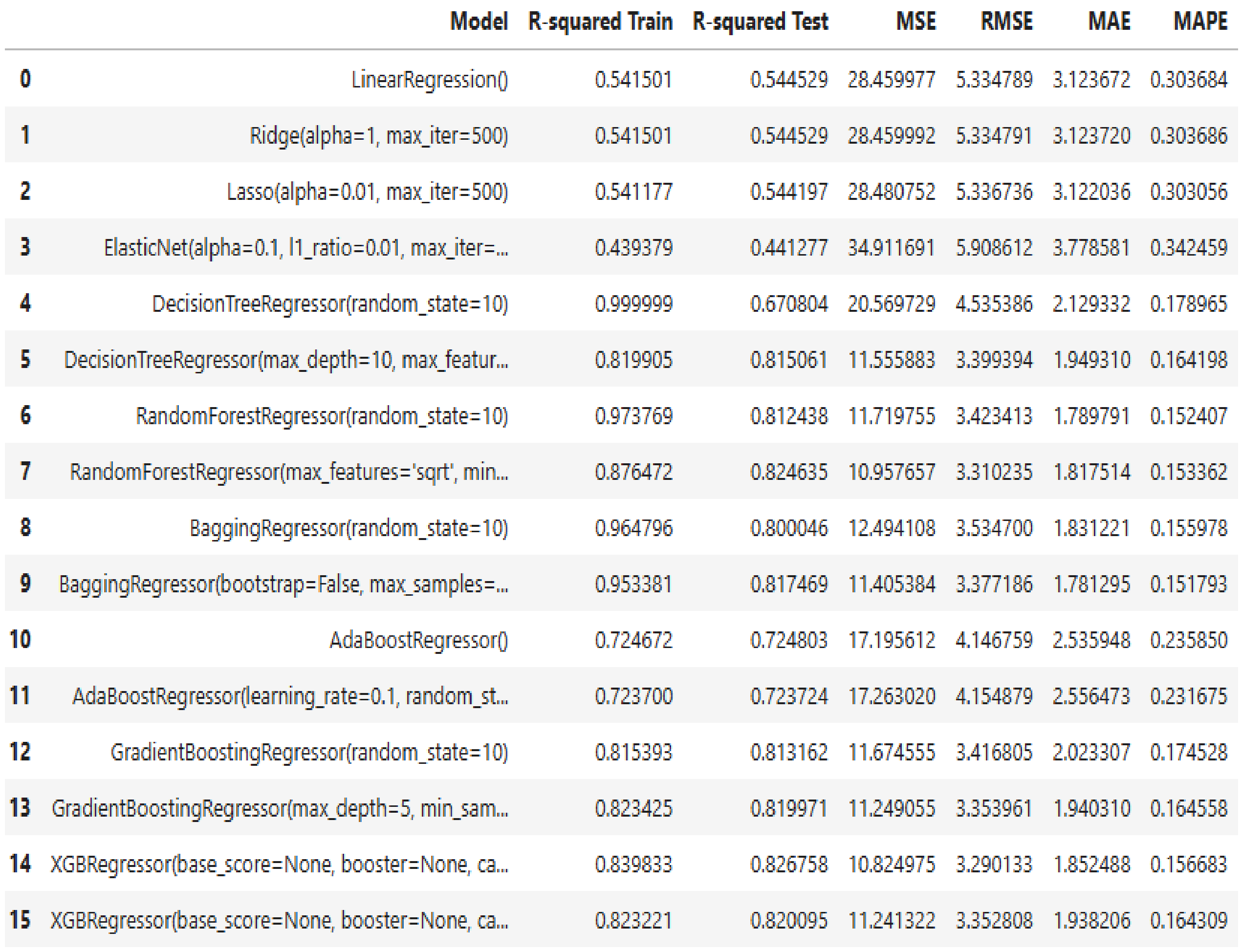


INFERENCE:

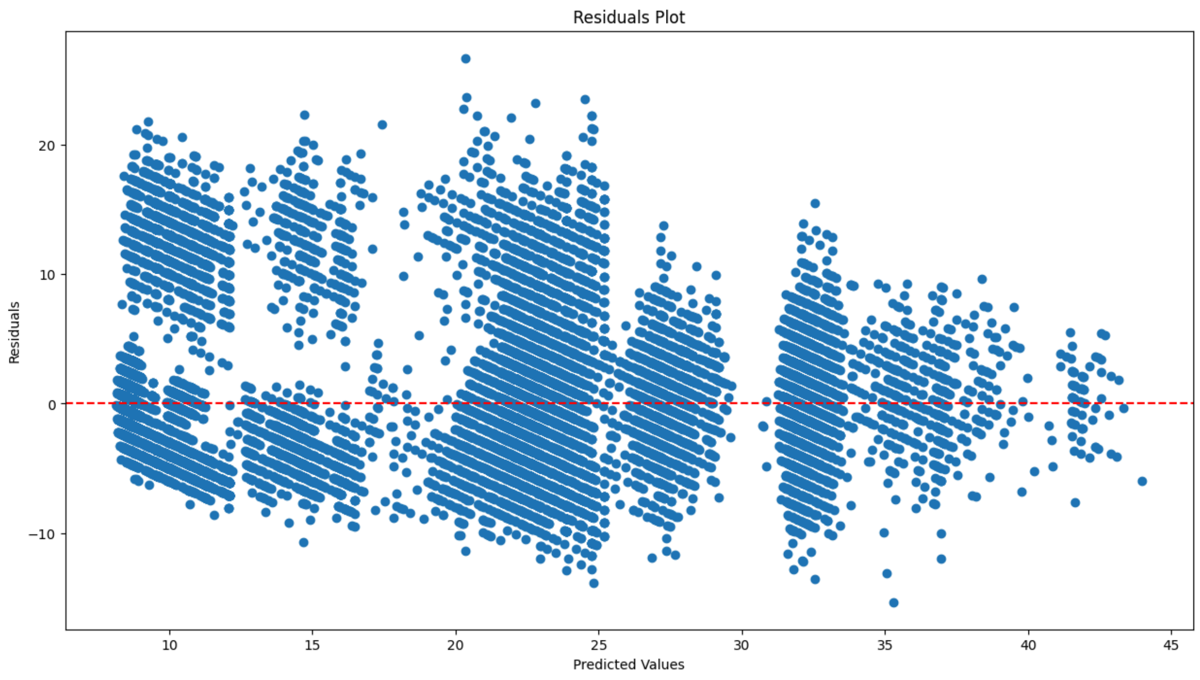
This XGBRegressor shows strong performance, balancing training fit and test generalization effectively. It has low error metrics and demonstrates good predictive accuracy. This model achieves excellent results (R-squared Test: 0.8268, RMSE: 3.30), outperforming many models with low error metrics, indicating strong generalization.

HYPER PARAMETER TUNING DECISION TREE -XG BOOST:

Tuning XGBoost optimizes its performance by adjusting learning rate, tree depth, and boosting parameters for better predictions.



	Model	R-squared Train	R-squared Test	MSE	RMSE	MAE	MAPE
0	LinearRegression()	0.541501	0.544529	28.459977	5.334789	3.123672	0.303684
1	Ridge(alpha=1, max_iter=500)	0.541501	0.544529	28.459992	5.334791	3.123720	0.303686
2	Lasso(alpha=0.01, max_iter=500)	0.541177	0.544197	28.480752	5.336736	3.122036	0.303056
3	ElasticNet(alpha=0.1, l1_ratio=0.01, max_iter=...	0.439379	0.441277	34.911691	5.908612	3.778581	0.342459
4	DecisionTreeRegressor(random_state=10)	0.999999	0.670804	20.569729	4.535386	2.129332	0.178965
5	DecisionTreeRegressor(max_depth=10, max_featur...	0.819905	0.815061	11.555883	3.399394	1.949310	0.164198
6	RandomForestRegressor(random_state=10)	0.973769	0.812438	11.719755	3.423413	1.789791	0.152407
7	RandomForestRegressor(max_features='sqrt', min...	0.876472	0.824635	10.957657	3.310235	1.817514	0.153362
8	BaggingRegressor(random_state=10)	0.964796	0.800046	12.494108	3.534700	1.831221	0.155978
9	BaggingRegressor(bootstrap=False, max_samples=...	0.953381	0.817469	11.405384	3.377186	1.781295	0.151793
10	AdaBoostRegressor()	0.724672	0.724803	17.195612	4.146759	2.535948	0.235850
11	AdaBoostRegressor(learning_rate=0.1, random_st...	0.723700	0.723724	17.263020	4.154879	2.556473	0.231675
12	GradientBoostingRegressor(random_state=10)	0.815393	0.813162	11.674555	3.416805	2.023307	0.174528
13	GradientBoostingRegressor(max_depth=5, min_sam...	0.823425	0.819971	11.249055	3.353961	1.940310	0.164558
14	XGBRegressor(base_score=None, booster=None, ca...	0.839833	0.826758	10.824975	3.290133	1.852488	0.156683
15	XGBRegressor(base_score=None, booster=None, ca...	0.823221	0.820095	11.241322	3.352808	1.938206	0.164309

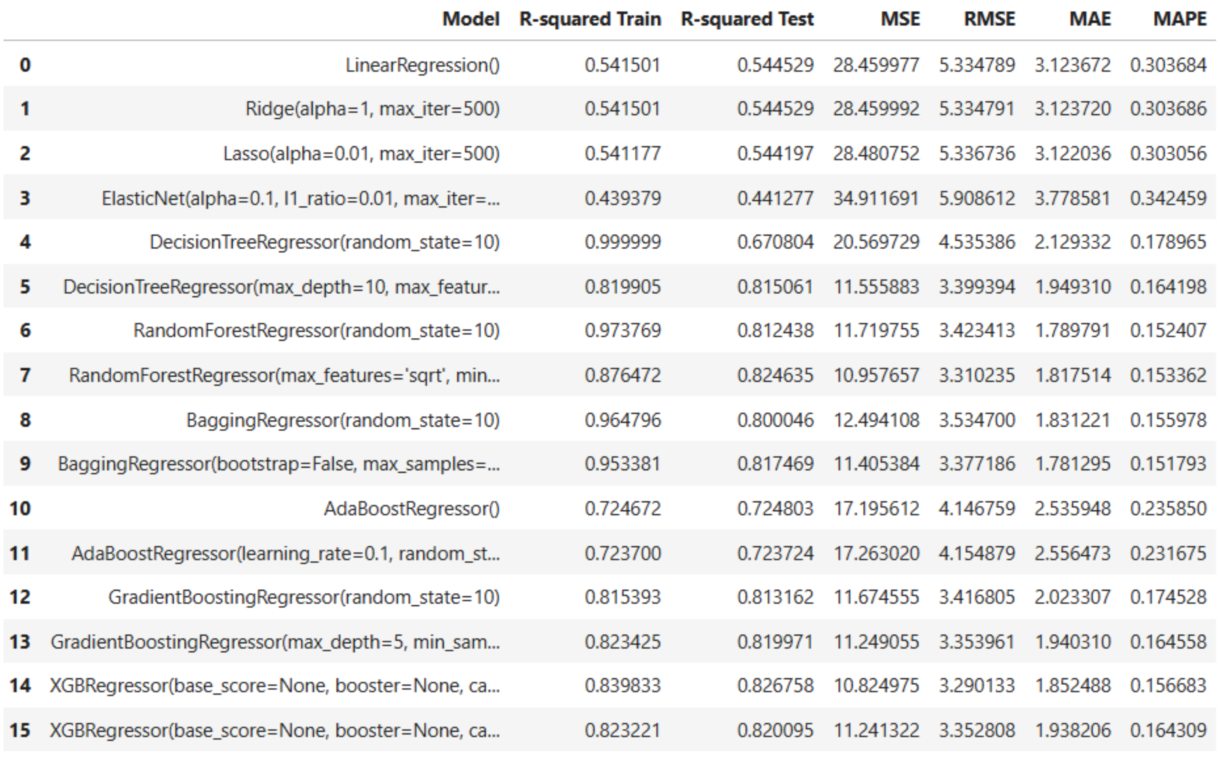


INFERENCE:

This XGBRegressor configuration is slightly less effective than , with marginally worse metrics across the board. It still performs well, but the first configuration edges it out in accuracy and generalization.

The tuned model performs well (R-squared Test: 0.8201, RMSE: 3.35), slightly underperforming compared to its default version but still competitive.

SCORE CARD AND COMPARISON OF DIFFERENT MODELS



	Model	R-squared Train	R-squared Test	MSE	RMSE	MAE	MAPE
0	LinearRegression()	0.541501	0.544529	28.459977	5.334789	3.123672	0.303684
1	Ridge(alpha=1, max_iter=500)	0.541501	0.544529	28.459992	5.334791	3.123720	0.303686
2	Lasso(alpha=0.01, max_iter=500)	0.541177	0.544197	28.480752	5.336736	3.122036	0.303056
3	ElasticNet(alpha=0.1, l1_ratio=0.01, max_iter=...	0.439379	0.441277	34.911691	5.908612	3.778581	0.342459
4	DecisionTreeRegressor(random_state=10)	0.999999	0.670804	20.569729	4.535386	2.129332	0.178965
5	DecisionTreeRegressor(max_depth=10, max_featur...	0.819905	0.815061	11.555883	3.399394	1.949310	0.164198
6	RandomForestRegressor(random_state=10)	0.973769	0.812438	11.719755	3.423413	1.789791	0.152407
7	RandomForestRegressor(max_features='sqrt', min...	0.876472	0.824635	10.957657	3.310235	1.817514	0.153362
8	BaggingRegressor(random_state=10)	0.964796	0.800046	12.494108	3.534700	1.831221	0.155978
9	BaggingRegressor(bootstrap=False, max_samples=...	0.953381	0.817469	11.405384	3.377186	1.781295	0.151793
10	AdaBoostRegressor()	0.724672	0.724803	17.195612	4.146759	2.535948	0.235850
11	AdaBoostRegressor(learning_rate=0.1, random_st...	0.723700	0.723724	17.263020	4.154879	2.556473	0.231675
12	GradientBoostingRegressor(random_state=10)	0.815393	0.813162	11.674555	3.416805	2.023307	0.174528
13	GradientBoostingRegressor(max_depth=5, min_sam...	0.823425	0.819971	11.249055	3.353961	1.940310	0.164558
14	XGBRegressor(base_score=None, booster=None, ca...	0.839833	0.826758	10.824975	3.290133	1.852488	0.156683
15	XGBRegressor(base_score=None, booster=None, ca...	0.823221	0.820095	11.241322	3.352808	1.938206	0.164309

Analysis of the Dataset Using Regression Models for Predicting Hospital Stay Duration

Model Performance Overview:

The evaluation of various regression models for predicting hospital stay duration highlights that the tuned versions of Random Forest Regressor, Gradient Boosting Regressor, and XGBoost Regressor are the top performers. Notably, the tuned Random Forest Regressor achieved a R² score of 82.46% on the test set, with an RMSE of 3.31 and a MAPE of 15.34%. These metrics indicate the model's strong ability to capture the underlying patterns of the target variable while maintaining high prediction accuracy.

Error Metrics:

The Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE) for each model provide insights into prediction accuracy. The tuned XGBoost Regressor recorded the lowest RMSE of 3.29, closely followed by the tuned Random Forest Regressor and Gradient Boosting Regressor, with

RMSE values of 3.31 and 3.35, respectively. These low RMSE and MAPE values highlight the effectiveness of ensemble models in minimizing prediction errors.

Overfitting Insights:

The Decision Tree Regressor displayed an R^2 score of 100% on the training set but dropped to 67.08% on the test set, indicating severe overfitting. This significant disparity underscores the model's inability to generalize to unseen data. Ensemble models like Random Forest, Bagging Regressor, and Boosting Regressors proved to be more robust, effectively balancing bias and variance while improving generalization.

Model Selection:

Among all evaluated models, the tuned versions of Random Forest, XGBoost, and Gradient Boosting Regressors exhibited the strongest generalization capabilities, with the following performance metrics:

Tuned Random Forest Regressor:

R^2: 82.46% | RMSE: 3.31 | MAPE: 15.34%

Tuned XGBoost Regressor:

R^2: 82.68% | RMSE: 3.29 | MAPE: 15.67%

Tuned Gradient Boosting Regressor:

R^2: 81.99% | RMSE: 3.35 | MAPE: 16.46%

These models deliver competitive performance, making them suitable for further analysis and optimization.

Hyperparameter Tuning:

While the current tuning has yielded promising results, further exploration of hyperparameters—such as maximum depth, learning rate, and tree-specific parameters—could enhance predictive performance. Techniques like Randomized Search or Bayesian Optimization could uncover optimal configurations that maximize model accuracy and minimize errors.

PROJECT JUSTIFICATION:

COMPLEXITY INVOLVED

The complexity of this project stems from several interrelated factors:

Data Diversity: The project requires the integration of various data types, including numerical, categorical, and temporal data. This complexity necessitates advanced preprocessing techniques and feature engineering to derive meaningful insights.

Non-Linear Relationships: The relationship between patient characteristics and hospital stay duration is often non-linear and influenced by multiple factors. Developing models that accurately capture these complex interactions requires sophisticated machine learning algorithms and fine-tuning of hyperparameters.

Dynamic Healthcare Environment: The healthcare landscape is continually evolving, with changes in medical protocols, treatment strategies, and patient demographics. The model must remain adaptable to these changes to ensure long-term effectiveness.

Data Quality Challenges: Issues such as missing values, inaccuracies in patient records, and biases in the dataset can hinder the model's performance. Implementing robust data cleaning and validation processes is essential to mitigate these challenges.

PROJECT OUTCOME:

Commercial Value:

Resource Optimization: Hospitals can significantly reduce costs by optimizing bed utilization and staffing based on accurate predictions of patient stay durations. This efficiency can lead to increased revenue and improved operational workflows.

Enhanced Patient Satisfaction: By minimizing wait times for beds and improving discharge planning, hospitals can provide a better patient experience, potentially leading to increased patient loyalty and positive referrals.

Academic Value:

Contributions to Healthcare Analytics: The project contributes to the academic field of healthcare analytics by providing a case study on the application of machine learning to predict hospital stay durations. It can serve as a reference for future research in predictive modeling and resource management in healthcare.

Knowledge Dissemination: Findings from the project can be published in academic journals, shared at conferences, or used for educational purposes in healthcare management and data science courses.

Social Value:

Improved Patient Outcomes: By ensuring that patients receive the appropriate care for the necessary duration, the project can enhance overall health outcomes and reduce readmission rates, which benefits public health.

Informed Policy-Making: Insights gained from the predictive model can inform healthcare policies related to patient flow management, resource allocation, and funding for hospital infrastructure improvements.

BUSINESS INSIGHTS & RECOMMENDATIONS:

Business Insights:

1. The use of hospital resources is significantly influenced by the departments of surgery and gynecology. Their huge caseloads have a direct impact on staff workload, operating room availability, and bed occupancy rates. These departments frequently run close to capacity, which emphasizes how crucial targeted resource planning is to preserving efficiency.
2. The length of hospital stays is greatly influenced by the age of the patient; older patients frequently need more extensive care and longer recovery times. Age-specific resource forecasting is essential since this demographic trend can put a burden on hospital resources, particularly in hospitals serving an elderly population.
3. Hospital infrastructure is further strained by the need for more specialized treatment and resources for certain medical diseases, such as heart disease. Furthermore, some physicians who focus on complicated cases might attract patients who require more resources, which could have an impact on hospital planning as a whole.
4. Unexpected increases in resource demands are caused by severe and urgent cases, such as crises or patients in need of critical care. Because of this unpredictability, hospitals must continue to allocate resources flexibly and take proactive steps to guarantee prompt treatment delivery.
5. Because departmental coordination, patient demographics, and staff workload are all intricately linked and have a cumulative impact on resource use, managing hospital efficiency necessitates a comprehensive approach.

Recommendations:

1. Use the predictive model to improve patient care, bed management, and resource allocation; pay special attention to high-demand departments like surgery and gynecology.
2. To improve the model's accuracy and suitability for a range of situations, include more patient data, such as socioeconomic characteristics and comorbidities.
3. Implement the predictive system in all hospitals to manage resources in real time and make smarter decisions during times of high demand or unforeseen spikes in patient admissions.
4. To establish a single system for hospital resource planning and management, gradually extend the concept to additional divisions and facilities.
5. To keep the model flexible and sensitive to shifts in patient requirements and healthcare trends, create a feedback loop that allows it to be updated and improved over time using actual hospital data.
6. Hospital employees can improve patient outcomes and efficiency while lessening the burden on hospital resources by learning how to incorporate predictive insights into daily operations.
7. Through the implementation of these measures, hospitals can improve patient care, make better use of their resources, and guarantee more efficient operations in a changing healthcare environment.