Hospital Length of Stay (LOS) Prediction

Context:

Hospital management is a vital area that gained a lot of attention during the COVID-19 pandemic. **Inefficient distribution of resources like beds, ventilators might lead to a lot of complications**. However, this can be mitigated by **predicting the length of stay (LOS) of a patient before getting admitted**. Once this is determined, the hospital can plan a suitable treatment, resources, and staff to reduce the LOS and increase the chances of recovery. The rooms and bed can also be planned in accordance with that.

HealthPlus hospital has been incurring a lot of losses in revenue and life due to its inefficient management system. They have been unsuccessful in allocating pieces of equipment, beds, and hospital staff fairly. **A system that could estimate the length of stay (LOS) of a patient can solve this problem to a great extent.**

Objective:

As a Data Scientist, you have been hired by HealthPlus to analyze the data, find out **what factors affect the LOS the most, and come up with a machine learning model which can predict the LOS of a patient** using the data available during admission and after running a few tests. Also, **bring about useful insights and policies from the data, which can help the hospital to improve their health care infrastructure and revenue.**

Data Dictionary:

The data contains various information recorded during the time of admission of the patient. It only contains **records of patients who were admitted to the hospital.** The detailed data dictionary is given below:

- **patientid**: Patient ID
- **Age**: Range of age of the patient
- **gender**: Gender of the patient
- **Type of Admission**: Trauma, emergency or urgent
- **Severity of Illness**: Extreme, moderate, or minor
- **health_condition**s: Any previous health conditions suffered by the patient
- **Visitors with Patient**: The number of patients who accompany the patient
- **Insurance**: Does the patient have health insurance or not?
- **Admission_Deposit**: The deposit paid by the patient during admission

- **Stay (in days)**: The number of days that the patient has stayed in the hospital. This is the **target variable**
- **Available Extra Rooms in Hospital**: The number of rooms available during admission
- **Department**: The department which will be treating the patient
- **Ward_Facility_Code**: The code of the ward facility in which the patient will be admitted
- **doctor_name**: The doctor who will be treating the patient
- **staff_available**: The number of staff who are not occupied at the moment in the ward

Approach to solve the problem:

1. Import the necessary libraries
2. Read the dataset and get an overview
3. Exploratory data analysis - a. Univariate b. Bivariate
4. Data preprocessing if any
5. Define the performance metric and build ML models
6. Checking for assumptions
7. Compare models and determine the best one
8. Observations and business insights

Importing Libraries

In [4]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings("ignore")

# Removes the limit for the number of displayed columns
pd.set_option("display.max_columns", None)

# Sets the limit for the number of displayed rows
pd.set_option("display.max_rows", 200)

# To build models for prediction
from sklearn.model_selection import train_test_split, cross_val_score, KFolc
from sklearn.linear_model import LinearRegression, Ridge, Lasso, ElasticNet
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor,BaggingRegressor

# To encode categorical variables
from sklearn.preprocessing import LabelEncoder
```

```
# For tuning the model
from sklearn.model_selection import GridSearchCV

# To check model performance
from sklearn.metrics import make_scorer,mean_squared_error, r2_score, mean_a
```

In [5]:
```
# Read the healthcare dataset file
data = pd.read_csv("healthcare_data.csv")
```

In [6]:
```
# Copying data to another variable to avoid any changes to original data
same_data = data.copy()
```

Data Overview

In [7]:
```
# View the first 5 rows of the dataset
data.head()
```

Out[7]:

	Available Extra Rooms in Hospital	Department	Ward_Facility_Code	doctor_name	staff_available	patientid
0	4	gynecology	D	Dr Sophia	0	33070
1	4	gynecology	B	Dr Sophia	2	34808
2	2	gynecology	B	Dr Sophia	8	44577
3	4	gynecology	D	Dr Olivia	7	3695
4	2	anesthesia	E	Dr Mark	10	108956

In [8]:
```
# View the last 5 rows of the dataset
data.tail()
```

Out[8]:

	Available Extra Rooms in Hospital	Department	Ward_Facility_Code	doctor_name	staff_available	pa
499995	4	gynecology	F	Dr Sarah	2	
499996	13	gynecology	F	Dr Olivia	8	
499997	2	gynecology	B	Dr Sarah	3	
499998	2	radiotherapy	A	Dr John	1	
499999	3	gynecology	F	Dr Sophia	3	

In [9]:

```
# Understand the shape of the data
data.shape
```

Out[9]:

```
(500000, 15)
```

- The dataset has **5,00,000 rows and 15 columns.**

In [10]:

```
# Checking the info of the data
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500000 entries, 0 to 499999
Data columns (total 15 columns):
 #   Column                             Non-Null Count   Dtype  
---  ------                             --------------   -----  
 0   Available Extra Rooms in Hospital  500000 non-null  int64  
 1   Department                         500000 non-null  object 
 2   Ward_Facility_Code                 500000 non-null  object 
 3   doctor_name                        500000 non-null  object 
 4   staff_available                    500000 non-null  int64  
 5   patientid                          500000 non-null  int64  
 6   Age                                500000 non-null  object 
 7   gender                             500000 non-null  object 
 8   Type of Admission                  500000 non-null  object 
 9   Severity of Illness                500000 non-null  object 
 10  health_conditions                  348112 non-null  object 
 11  Visitors with Patient              500000 non-null  int64  
 12  Insurance                          500000 non-null  object 
 13  Admission_Deposit                  500000 non-null  float64
 14  Stay (in days)                     500000 non-null  int64  
dtypes: float64(1), int64(5), object(9)
memory usage: 57.2+ MB
```

Observations:

- Available Extra Rooms in Hospital, staff_available, patientid, Visitors with Patient, Admission_Deposit, and Stay (in days) are of **numeric data type** and the rest of the columns are of **object data type**.
- The number of non-null values is the same as the total number of entries in the data, i.e., **there are no null values.**
- The column patientid is an identifier for patients in the data. This column will not help with our analysis so we can drop it.

In [11]:
```
# To view patientid and the number of times they have been admitted to the h
data['patientid'].value_counts()
```

Out[11]:
```
patientid
126719    21
125695    21
44572     21
126623    21
125625    19
          ..
37634      1
91436      1
118936     1
52366      1
105506     1
Name: count, Length: 126399, dtype: int64
```

Observation:

- **The maximum number of times the same patient admitted to the hospital is 21 and minimum is 1.**

In [12]:
```
# Dropping patientid from the data as it is an identifier and will not add v
data=data.drop(columns=["patientid"])
```

In [13]:
```
# Checking for duplicate values in the data
data.duplicated().sum()
```

Out[13]:
```
0
```

Observation:

- Data contains unique rows. There is no need to remove any rows.

In [14]:
```
# Checking the descriptive statistics of the columns
data.describe().T
```

Out[14]:

	count	mean	std	min	25%
Available Extra Rooms in Hospital	500000.0	3.638800	2.698124	0.000000	2.000000
staff_available	500000.0	5.020470	3.158103	0.000000	2.000000
Visitors with Patient	500000.0	3.549414	2.241054	0.000000	2.000000
Admission_Deposit	500000.0	4722.315734	1047.324220	1654.005148	4071.714532
Stay (in days)	500000.0	12.381062	7.913174	3.000000	8.000000

Observations:

- There are around **3 rooms available in the hospital on average** and there are times when the hospital is full and there are no rooms available (minimum value is 0). The **maximum number of rooms available in the hospital is 24**.
- **On average, there are around 5 staff personnel available to treat the new patients** but it can also be zero at times. The maximum number of staff available in the hospital is 10.
- **On average, around 3 visitors accompany the patient.** Some patients come on their own (minimum value is zero) and a few cases have 32 visitors. It will be interesting to see if there is any relationship between the number of visitors and the severity of the patient.
- **The average admission deposit lies around 4,722 dollars and a minimum of 1,654 dollars is paid on every admission.**
- **Patient's stay ranges from 3 to 51 days.** There might be outliers in this variable. The median length of stay is 9 days.

In [15]:

```
# List of all important categorical variables
cat_col = ["Department", "Type of Admission", 'Severity of Illness', 'gender

# Printing the number of occurrences of each unique value in each categorica
for column in cat_col:
    print(data[column].value_counts(1))
    print("-" * 50)
```

```
Department
gynecology           0.686956
radiotherapy         0.168630
anesthesia           0.088358
TB & Chest disease   0.045780
surgery              0.010276
Name: proportion, dtype: float64
--------------------------------------------------
Type of Admission
Trauma       0.621072
Emergency    0.271568
Urgent       0.107360
Name: proportion, dtype: float64
--------------------------------------------------
Severity of Illness
Moderate    0.560394
Minor       0.263074
Extreme     0.176532
Name: proportion, dtype: float64
--------------------------------------------------
gender
Female    0.74162
Male      0.20696
Other     0.05142
Name: proportion, dtype: float64
--------------------------------------------------
Insurance
Yes    0.78592
No     0.21408
Name: proportion, dtype: float64
--------------------------------------------------
health_conditions
Other                  0.271209
High Blood Pressure    0.228093
Diabetes               0.211553
Asthama                0.188198
Heart disease          0.100947
Name: proportion, dtype: float64
--------------------------------------------------
doctor_name
Dr Sarah     0.199192
Dr Olivia    0.196704
Dr Sophia    0.149506
Dr Nathan    0.141554
Dr Sam       0.111422
Dr John      0.102526
Dr Mark      0.088820
Dr Isaac     0.006718
Dr Simon     0.003558
Name: proportion, dtype: float64
--------------------------------------------------
Ward_Facility_Code
F    0.241076
D    0.238110
B    0.207770
E    0.190748
```

```
A    0.093102
C    0.029194
Name: proportion, dtype: float64
--------------------------------------------------
Age
21-30     0.319586
31-40     0.266746
41-50     0.160812
11-20     0.093072
61-70     0.053112
51-60     0.043436
71-80     0.037406
81-90     0.016362
0-10      0.006736
91-100    0.002732
Name: proportion, dtype: float64
--------------------------------------------------
```

Observations:

- **The majority of patients (~82%) admit to the hospital with moderate and minor illness**, which is understandable as extreme illness is less frequent than moderate and minor illness.
- **Gynecology department gets the most number of patients (~68%)** in the hospital, whereas patients in Surgery department are very few (~1%).
- **Ward A and C accommodate the least number of patients (~12%).** These might be wards reserved for patient with extreme illness and patients who need surgery. It would be interesting to see if patients from these wards also stay for longer duration.
- **The majority of patients belong to the age group of 21-50 (~75%), and the majority of patients are women (~74%).** The most number of patients in the gynecology department of the hospital can justify this.
- Most of the patients admitted to the hospital are the cases of trauma (~62%).
- After 'Other' category, **High Blood Pressure and Diabetes are the most common health conditions.**

Exploratory Data Analysis (EDA)

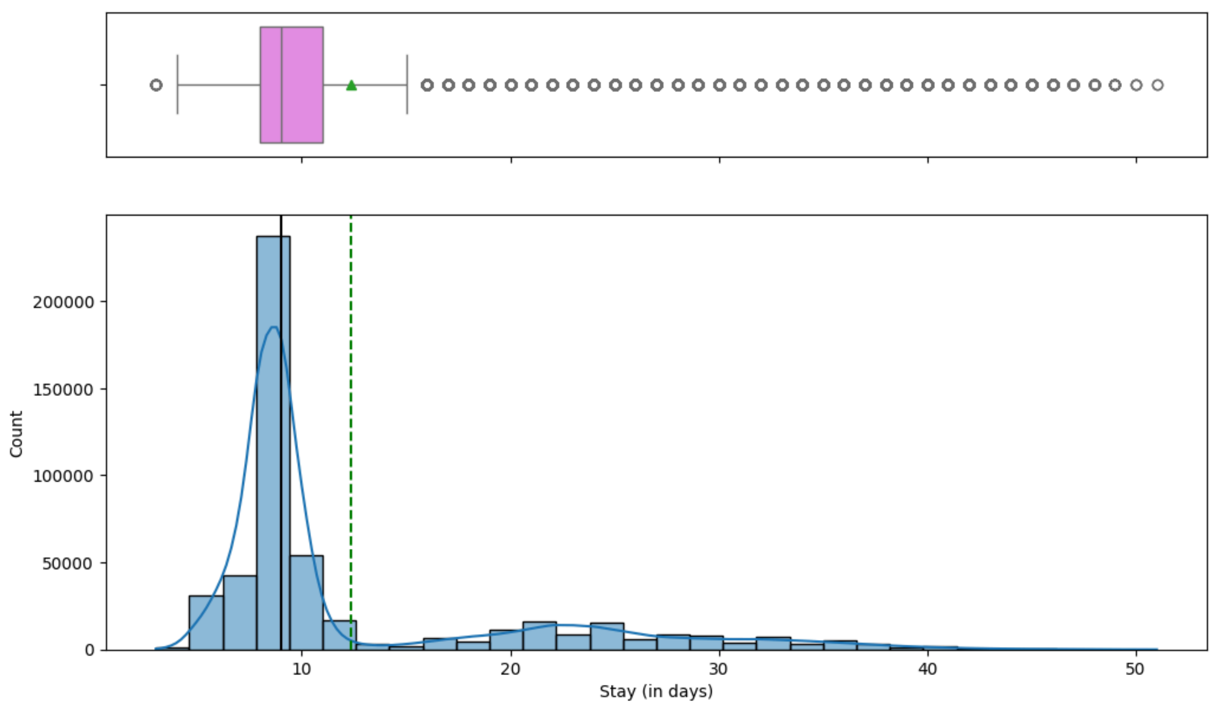
Univariate Analysis

In [16]:
```python
# Function to plot a boxplot and a histogram along the same scale

def histogram_boxplot(data, feature, figsize=(12, 7), kde=False, bins=None):
    """
    Boxplot and histogram combined

    data: dataframe
    feature: dataframe column
    figsize: size of figure (default (12,7))
```

```
    kde: whether to the show density curve (default False)
    bins: number of bins for histogram (default None)
    """
    f2, (ax_box2, ax_hist2) = plt.subplots(
        nrows = 2,      # Number of rows of the subplot grid = 2
        sharex = True,  # x-axis will be shared among all subplots
        gridspec_kw = {"height_ratios": (0.25, 0.75)},
        figsize = figsize,
    )                   # Creating the 2 subplots
    sns.boxplot(data = data, x = feature, ax = ax_box2, showmeans = True, co
    )                   # Boxplot will be created and a star will indicate t
    sns.histplot(
        data = data, x = feature, kde = kde, ax = ax_hist2, bins = bins, pal
    ) if bins else sns.histplot(
        data = data, x = feature, kde = kde, ax = ax_hist2
    )                   # For histogram
    ax_hist2.axvline(
        data[feature].mean(), color = "green", linestyle = "--"
    )                   # Add mean to the histogram
    ax_hist2.axvline(
        data[feature].median(), color = "black", linestyle = "-"
    )                   # Add median to the histogram
```

Length of stay

In [17]:
```
histogram_boxplot(data, "Stay (in days)", kde = True, bins = 30)
```



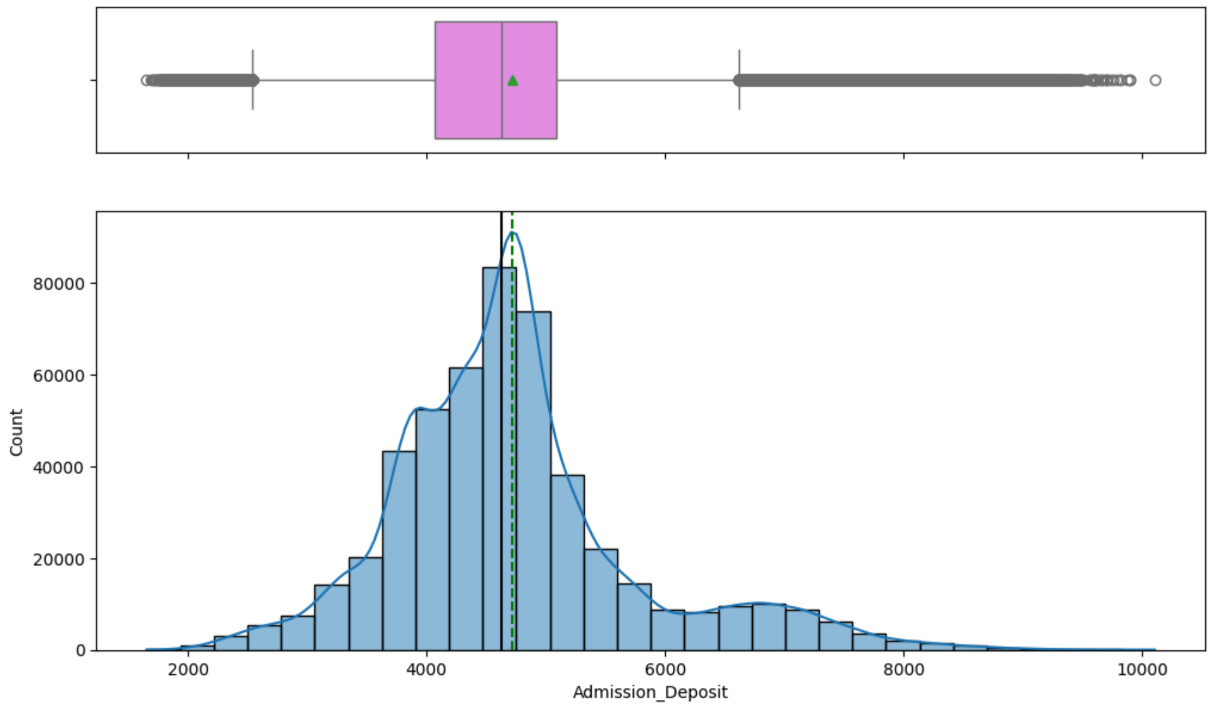
Observations:

- **Fewer patients are staying more than 10 days in the hospital and very few stay for more than 40 days**. This might be because the majority of patients are admitted for moderate or minor illnesses.

- The peak of the distribution shows that **most of the patients stay for 8-9 days in the hospital.**

Admission Deposit

In [18]:
```
histogram_boxplot(data, "Admission_Deposit", kde = True, bins = 30)
```

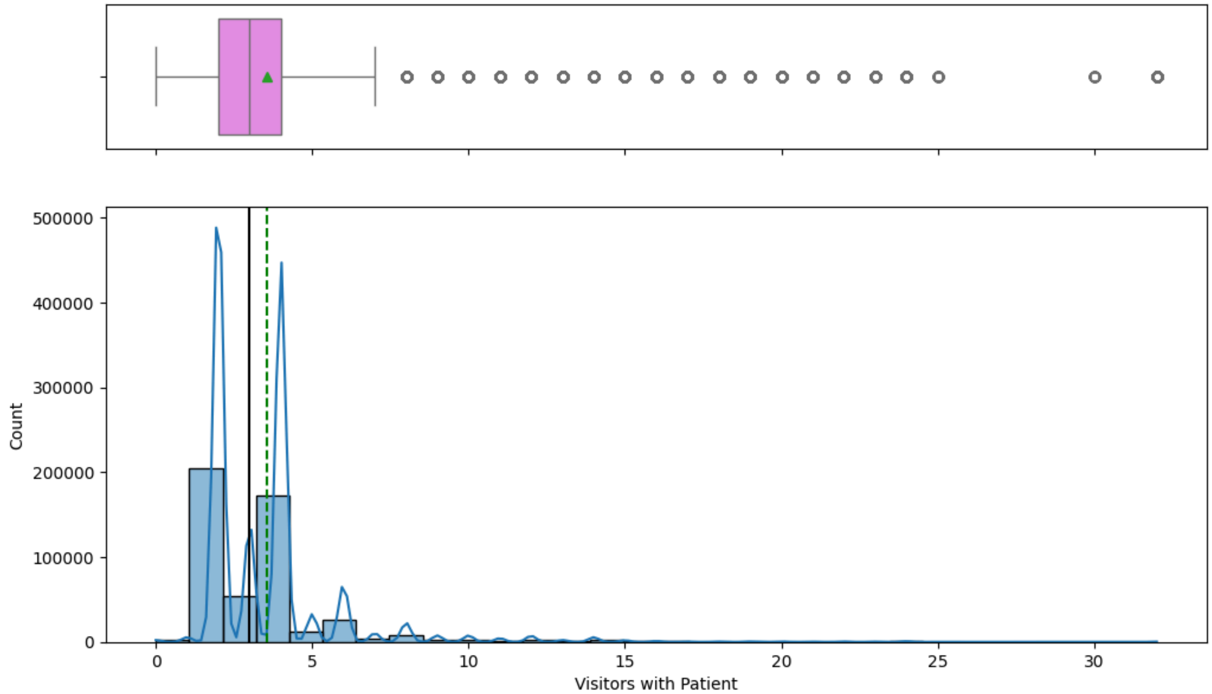


Observation:

- The **distribution of admission fees is close to normal with outliers on both sides**. Few patients are paying a high amount of admission fees and few patients are paying a low amount of admission fees.

Visitors with Patients

In [19]:
```
histogram_boxplot(data, "Visitors with Patient", kde = True, bins = 30)
```



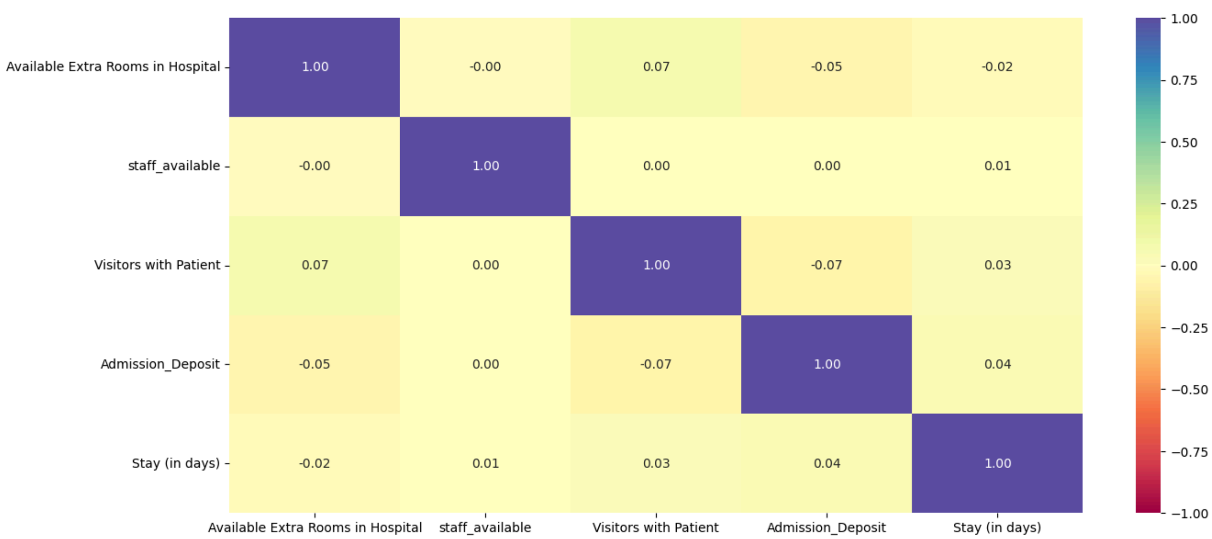
Observations:

- The distribution of the number of visitors with the patient is **highly skewed towards the right**.
- **2 and 4 are the most common number of visitors with patients.**

Bivariate Analysis

In [20]:
```
# Finding the correlation between various columns of the dataset
plt.figure(figsize = (15,7))
sns.heatmap(data.corr(numeric_only = True), annot = True, vmin = -1, vmax =
```

Out[20]: <Axes: >



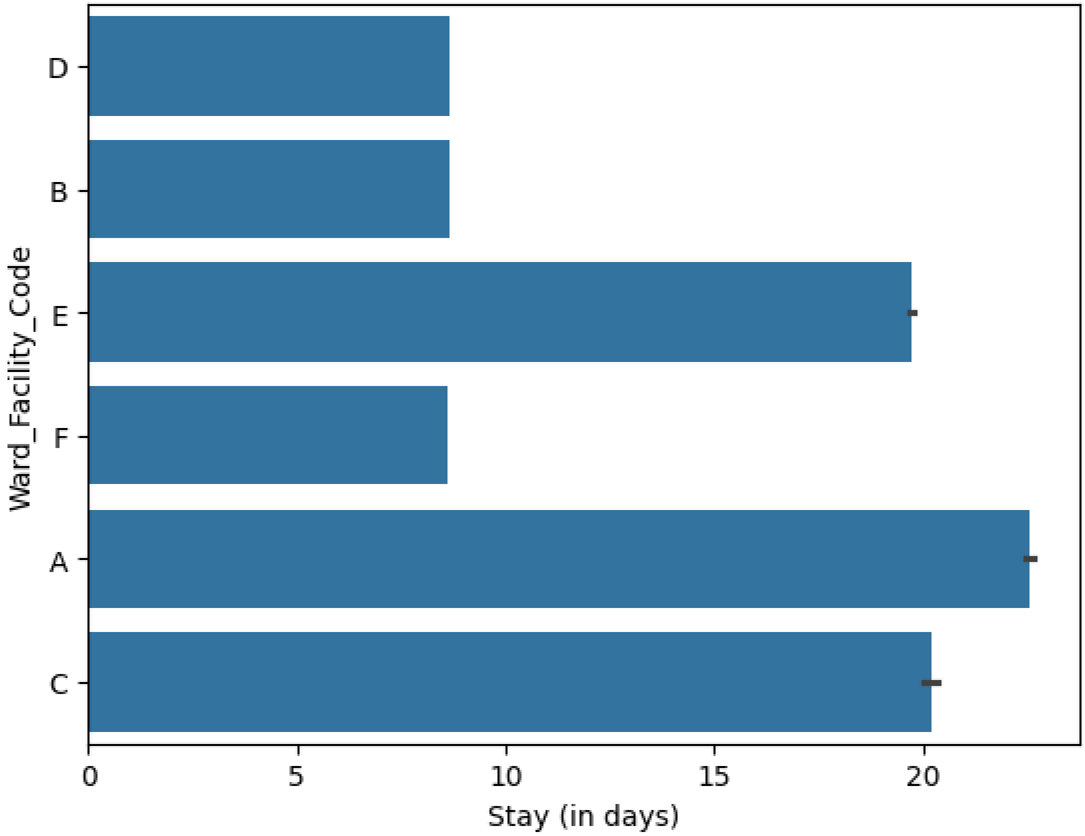
Observations:

- The heatmap shows that there is **no correlation between variables**.
- The continuous variables show no correlation with the target variable (Stay (in days)), which indicates that the **categorical variables might be more important for the prediction.**

In [21]:
```
# Function to plot stacked bar plots

def stacked_barplot(data, predictor, target):
    """
    Print the category counts and plot a stacked bar chart

    data: dataframe
    predictor: independent variable
    target: target variable
    """
    count = data[predictor].nunique()
    sorter = data[target].value_counts().index[-1]
    tab1 = pd.crosstab(data[predictor], data[target], margins = True).sort_v
        by = sorter, ascending = False
    )
    print(tab1)
    print("-" * 120)
    tab = pd.crosstab(data[predictor], data[target], normalize = "index").sc
        by = sorter, ascending = False
    )
    tab.plot(kind = "bar", stacked = True, figsize = (count + 1, 5))
    plt.legend(
        loc = "lower left",
        frameon = False,
    )
    plt.legend(loc = "upper left", bbox_to_anchor = (1, 1))
    plt.show()
```

Let's start by checking the distribution of the LOS for the various wards

In [22]:
```
sns.barplot(y = 'Ward_Facility_Code', x = 'Stay (in days)', data = data)
plt.show()
```

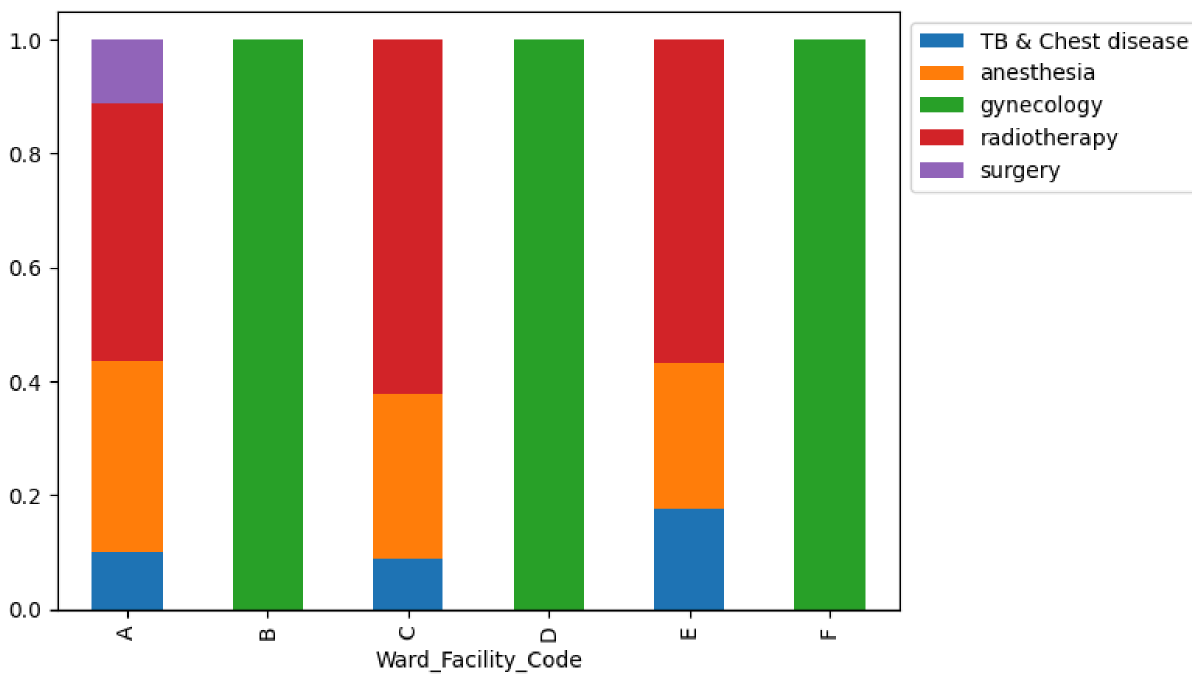


Observation:

- The hypothesis we made earlier is correct, i.e., **wards A and C has the patients staying for the longest duration, which implies these wards might be for patients with serious illnesses.**

In [23]:
```
stacked_barplot(data, "Ward_Facility_Code", "Department")
```

```
Department          TB & Chest disease  anesthesia  gynecology  radiotherapy  \
Ward_Facility_Code
A                                 4709       15611           0         21093
All                              22890       44179      343478         84315
B                                    0           0      103885             0
C                                 1319        4199           0          9079
D                                    0           0      119055             0
E                                16862       24369           0         54143
F                                    0           0      120538             0

Department          surgery     All
Ward_Facility_Code
A                      5138   46551
All                    5138  500000
B                         0  103885
C                         0   14597
D                         0  119055
E                         0   95374
F                         0  120538
------------------------------------------------------------------------------------------------------------------------
```



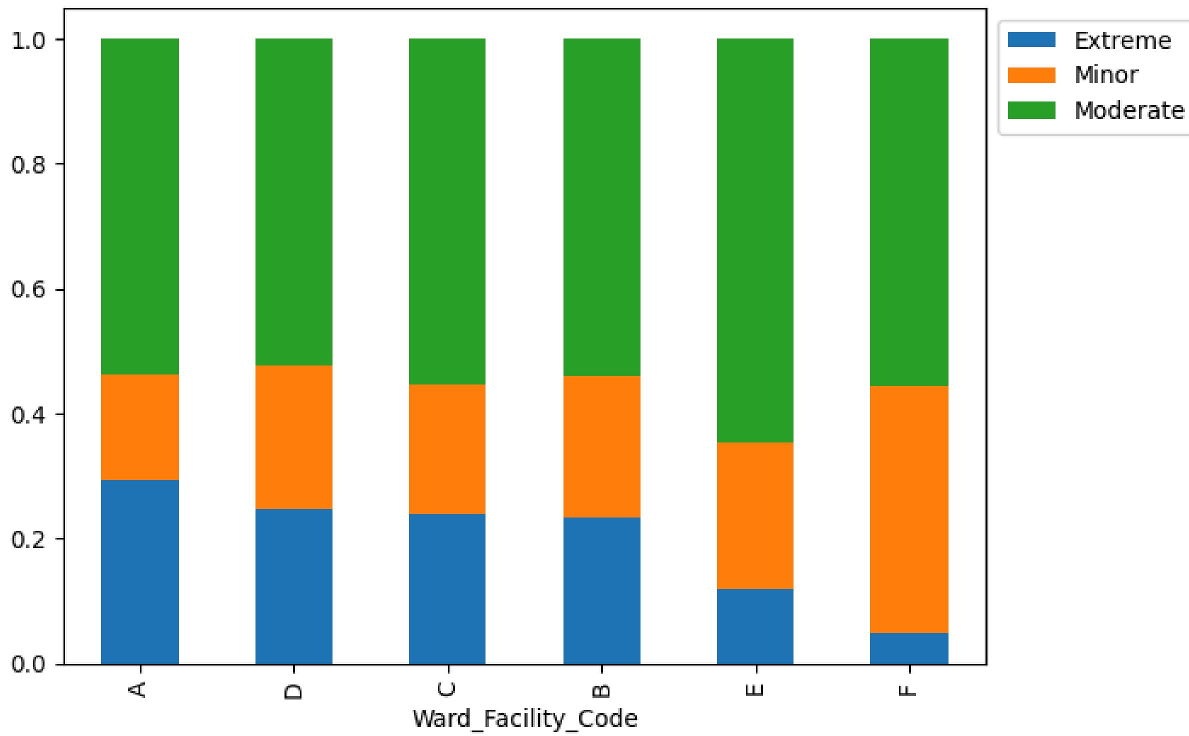
Observations:

- **Ward Facility B, D, and F are dedicated only to the gynecology department.**
- Wards A, C, and E have patients with all other diseases, and **patients undergoing surgery are admitted to ward A only.**

Usually, the more severe the illness, the more the LOS, let's check the distribution of severe patients in various wards.

In [24]:
```
stacked_barplot(data, "Ward_Facility_Code", "Severity of Illness")
```

```
Severity of Illness  Extreme   Minor  Moderate     All
Ward_Facility_Code
All                    88266  131537    280197  500000
D                      29549   27220     62286  119055
B                      24222   23579     56084  103885
A                      13662    7877     25012   46551
E                      11488   22254     61632   95374
F                       5842   47594     67102  120538
C                       3503    3013      8081   14597
------------------------------------------------------------------------------
---------------------------------------------
```

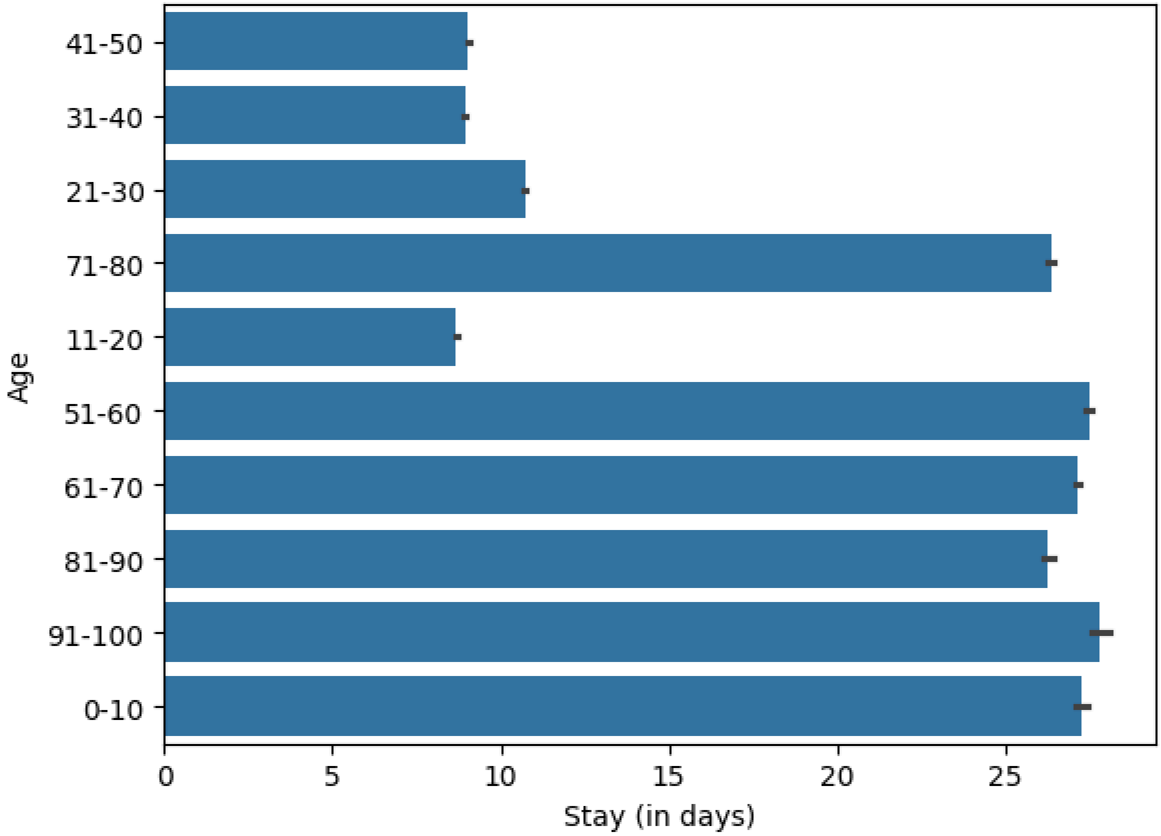


Observations:

- **Ward A has the highest number of extreme cases.** We observed earlier that ward A has the longest length of stay in the hospital as well. It might require more staff and resources as compared to other wards.
- **Ward F has the highest number of minor cases and Ward E has the highest number of moderate cases.**

Age can also be an important factor to find the length of stay. Let's check the same.

In [25]:
```
sns.barplot(y = 'Age', x = 'Stay (in days)', data = data)
plt.show()
```



Observation:

- **Patients aged between 1-10 and 51-100 tend to stay the most number of days in the hospital.** This might be because the majority of the patients between the 21-50 age group get admitted to the gynecology department and patients in age groups 1-10 and 5-100 might get admitted due to some serious illness.

Let's look at the doctors, their department names, and the total number of patients they have treated.

In [26]:
```
data.groupby(['doctor_name'])['Department'].agg(Department_Name='unique',Pat
```

Out[26]:

doctor_name	Department_Name	Patients_Treated
Dr Isaac	[surgery]	3359
Dr John	[TB & Chest disease, anesthesia, radiotherapy]	51263
Dr Mark	[anesthesia, TB & Chest disease]	44410
Dr Nathan	[gynecology]	70777
Dr Olivia	[gynecology]	98352
Dr Sam	[radiotherapy]	55711
Dr Sarah	[gynecology]	99596
Dr Simon	[surgery]	1779
Dr Sophia	[gynecology]	74753

Observations:

- The hospital employs a total of 9 doctors. Four of the doctors work in the department of gynecology, which sees the most patients.
- The majority of patients that attended the hospital were treated by Dr. Sarah and Olivia.
- Two doctors are working in the surgical department (Dr. Isaac and Dr. Simon), while Dr. Sam works in the radiotherapy department.
- The only two doctors who work in several departments are Dr. John and Dr. Mark.

Data Preparation for Model Building

- Before we proceed to build a model, we'll have to encode categorical features.
- Separate the independent variables and dependent Variables.
- We'll split the data into train and test to be able to evaluate the model that we train on the training data.

In [27]:

```
# Creating dummy variables for the categorical columns
# drop_first=True is used to avoid redundant variables
data = pd.get_dummies(
    data,
    columns = data.select_dtypes(include = ["object", "category"]).columns.t
    drop_first = True,
)
```

In [28]:

```
# Check the data after handling categorical data
data
```

Out[28]:

	Available Extra Rooms in Hospital	staff_available	Visitors with Patient	Admission_Deposit	Stay (in days)	Department_ar
0	4	0	4	2966.408696	8	
1	4	2	2	3554.835677	9	
2	2	8	2	5624.733654	7	
3	4	7	4	4814.149231	8	
4	2	10	2	5169.269637	34	
...	...	...	...	...	...	
499995	4	2	3	4105.795901	10	
499996	13	8	2	4631.550257	11	
499997	2	3	2	5456.930075	8	
499998	2	1	2	4694.127772	23	
499999	3	3	4	4713.868519	10	

500000 rows × 42 columns

In [29]:
```
# Separating independent variables and the target variable
x = data.drop('Stay (in days)',axis=1)

y = data['Stay (in days)']
```

In [30]:
```
# Splitting the dataset into train and test datasets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, s
```

In [31]:
```
# Checking the shape of the train and test data
print("Shape of Training set : ", x_train.shape)
print("Shape of test set : ", x_test.shape)
```

```
Shape of Training set :  (400000, 41)
Shape of test set :  (100000, 41)
```

Model Building

- We will be using different metrics functions defined in sklearn like RMSE, MAE, $R2$, Adjusted $R2$, and MAPE for regression models evaluation. We will define a function to calculate these metric.
- The mean absolute percentage error (MAPE) measures the accuracy of predictions as a percentage, and can be calculated as the average of absolute percentage error for all data points. The absolute percentage error is defined as predicted value

minus actual values divided by actual values. It works best if there are no extreme values in the data and none of the actual values are 0.

In [32]:
```
# Function to compute adjusted R-squared
def adj_r2_score(predictors, targets, predictions):
    r2 = r2_score(targets, predictions)
    n = predictors.shape[0]
    k = predictors.shape[1]
    return 1 - ((1 - r2) * (n - 1) / (n - k - 1))


# Function to compute MAPE
def mape_score(targets, predictions):
    return np.mean(np.abs(targets - predictions) / targets) * 100


# Function to compute different metrics to check performance of a regression
def model_performance_regression(model, predictors, target):
    """
    Function to compute different metrics to check regression model performa

    model: regressor
    predictors: independent variables
    target: dependent variable
    """

    pred = model.predict(predictors)                  # Predict using the in
    r2 = r2_score(target, pred)                       # To compute R-squared
    adjr2 = adj_r2_score(predictors, target, pred)    # To compute adjusted 
    rmse = np.sqrt(mean_squared_error(target, pred))  # To compute RMSE
    mae = mean_absolute_error(target, pred)           # To compute MAE
    mape = mape_score(target, pred)                   # To compute MAPE

    # Creating a dataframe of metrics
    df_perf = pd.DataFrame(
        {
            "RMSE": rmse,
            "MAE": mae,
            "R-squared": r2,
            "Adj. R-squared": adjr2,
            "MAPE": mape,
        },
        index=[0],
    )

    return df_perf
```

Linear Regression

In [33]:
```
from sklearn.linear_model import LinearRegression

# Initialize the model
model = LinearRegression()
```

```python
# Fit the model on the training data
model.fit(x_train, y_train)
```

Out[33]:

```
▾ LinearRegression

LinearRegression()
```

In [34]:
```python
# Checking performance on the training data
linear_reg = model_performance_regression(model, x_train, y_train)
linear_reg
```

Out[34]:

	RMSE	MAE	R-squared	Adj. R-squared	MAPE
0	3.135093	2.146244	0.842813	0.842796	19.591833

In [35]:
```python
# Checking performance on the testing data
linear_reg_test = model_performance_regression(model, x_test, y_test)
linear_reg_test
```

Out[35]:

	RMSE	MAE	R-squared	Adj. R-squared	MAPE
0	3.144055	2.155765	0.843028	0.842964	19.676966

Observations:

- **The Root Mean Squared Error** and the **adjusted** R^2 of train and test data are **very close**, indicating that **our model is not overfitting** to the training data.
- The adjusted R^2 of ~0.84 implies that the independent variables are able to explain ~84% variance in the target variable.
- Mean Absolute Error (MAE) indicates that the current model can predict LOS of patients within **mean error of 2.15 days** on the test data.
- The units of both RMSE and MAE are the same, i.e., days in this case. But RMSE is greater than MAE because it penalizes the outliers more.
- **Mean Absolute Percentage Error is ~19%** on the test data, indicating that the average difference between the predicted value and the actual value is ~19%.

Regularization

Regularization is a fundamental concept in machine learning. It is a method of preventing the model from **overfitting** by adding additional information to it.

The machine learning model may perform well with training data but not with test data. It means that when dealing with unseen data, the model cannot anticipate the result since it introduces noise into the output, and so the model is termed **overfit**. A **regularization** technique can be used to solve this problem.

By lowering the magnitude of the variables, this technique allows for the preservation of all variables or features in the model. As a result, it maintains accuracy as well as model generalization.

Its primary function is to regularize or lower the coefficient of features towards zero. In other words, "the regularization strategy reduces the magnitude of the features while maintaining the same number of features."

Regularization is accomplished by introducing a penalty or complexity term into the complex model.

Regularization procedures are classified into two types, which are listed below:

- **Ridge Regression**
- **Lasso Regression**

Ridge Regression

Ridge regression is a sort of linear regression in which a **small amount of bias** is introduced to improve long-term predictions.

- Ridge regression is a regularization technique that is **used to reduce model complexity**. It's also known as L_2 **regularization**.
- The penalty term is added to the cost function in this technique. The amount of bias introduced into the model is referred to as the **Ridge Regression penalty**.
- We may compute it by multiplying the squared weight of each individual feature by the alpha.
- In general, Ridge Regression calculates the equation's parameters:

$$\hat{y} \ = slope \times X + y\, intercept$$

By minimizing the:

$$the\ sum\ of\ squared\ residuals + \alpha \times slope^2$$

- As we can see from the above equation, if the values of α tend to **zero**, the equation becomes the linear regression model's cost function. As a result, for the **minimum value of** α, the model will be similar to the linear regression model.
- Because a general linear or polynomial regression will fail if the independent variables are highly collinear, Ridge regression can be utilized to tackle such

situations.
- When we have more parameters than samples, it is easier to solve problems using Ridge Regression.

Ridge Regression with default parameters

In [36]:
```
ridge_model = Ridge() #creating Ridge Regression model
ridge_model.fit(x_train, y_train) # Fitting the data into the model
```

Out[36]:
```
▾ Ridge
Ridge()
```

In [37]:
```
ridge_reg = model_performance_regression(ridge_model, x_test, y_test) #getti
ridge_reg
```

Out[37]:

	RMSE	MAE	R-squared	Adj. R-squared	MAPE
0	3.144057	2.155826	0.843028	0.842963	19.677968

Observations:

- The performance metrics are showing almost similar results as compared to the Least Squares method.

Ridge Regression with optimized α

In [38]:
```
folds = KFold(n_splits=10, shuffle=True, random_state=1)
params = {'alpha':[0.001, 0.01, 0.1, 0.2, 0.5, 0.9, 1, 5,10,20]}
model = Ridge()
model_cv = GridSearchCV(estimator=model, param_grid=params, scoring='r2', cv
model_cv.fit(x_train,y_train)
```

Out[38]:
```
▸ GridSearchCV
▸ best_estimator_: Ridge
  ▸ Ridge
```

In [39]:
```
model_cv.best_params_ #getting optimised parameters for alpha
```

Out[39]:
```
{'alpha': 0.1}
```

In [40]:
```
ridge_model_tuned = Ridge(alpha=0.1) ##creating Tuned Ridge Regression model
```

```
ridge_model_tuned.fit(x_train, y_train) # Fitting the data into the tuned mo
```

Out[40]:

```
▾ Ridge
Ridge(alpha=0.1)
```

In [41]:

```
ridge_reg_tuned = model_performance_regression(ridge_model_tuned, x_test, y_
ridge_reg_tuned
```

Out[41]:

	RMSE	MAE	R-squared	Adj. R-squared	MAPE
0	3.144055	2.155771	0.843028	0.842964	19.677066

Observations:

- After applying the Grid SearchCV, the optimized value of alpha results out to be 0.1.
- It can be observed that after tuning the parameters of Ridge Regression, the performance parameters does not change implying that Ridge Regression does not help in improving the model.

Lasso Regression

Lasso regression is another regularisation technique for reducing model complexity. It is an abbreviation for **Least Absolute and Selection Operator.**

- It is identical to Ridge Regression except that the **penalty term only contains absolute weights** rather than a square of weights.

- Because it uses absolute data, it **can decrease the slope to zero**, whereas **Ridge Regression can only get close to zero**.

- It is also known as L_1 **regularisation**.

Fundamentally, Lasso Regression calculates the equation's parameters:

$$\hat{y} \ = slope \times X + y\ intercept$$

By minimizing the:

$$the\ sum\ of\ squared\ residuals + \alpha \times |slope|$$

Lasso Regression with default parameters

In [42]:

```
lasso_model = Lasso()
lasso_model.fit(x_train, y_train)
```

Out[42]:

```
▾ Lasso
Lasso()
```

In [43]:

```python
lasso_reg = model_performance_regression(lasso_model, x_test, y_test)
lasso_reg
```

Out[43]:

	RMSE	MAE	R-squared	Adj. R-squared	MAPE
0	6.064339	3.873332	0.416006	0.415766	34.652716

Observations:

- After fitting the data into Lasso Regression Model with default value of alpha (=1), the performance metrics are showing poor results as compared to Least Squares method and Ridge Regression.
- We can tune the alpha to get the optimized value similar to Ridge Regression using Grid SearchCV.

Lasso Regression with optimized α

In [44]:

```python
folds = KFold(n_splits=10, shuffle=True, random_state=1)
params = {'alpha':[0.001, 0.01, 0.1, 0.2, 0.5, 0.9, 1, 5,10,20]}
model = Lasso()
model_cv = GridSearchCV(estimator=model, param_grid=params, scoring='r2', cv
model_cv.fit(x_train,y_train)
```

Out[44]:

```
▸ GridSearchCV
▸ best_estimator_: Lasso
  ▸ Lasso
```

In [45]:

```python
model_cv.best_params_
```

Out[45]: {'alpha': 0.001}

In [46]:

```python
lasso_model_tuned = Lasso(alpha=0.001)
lasso_model_tuned.fit(x_train, y_train)
```

Out[46]:

```
▾ Lasso
Lasso(alpha=0.001)
```

In [47]:
```
lasso_reg_tuned = model_performance_regression(lasso_model_tuned, x_test, y_
lasso_reg_tuned
```

Out[47]:

	RMSE	MAE	R-squared	Adj. R-squared	MAPE
0	3.144315	2.157198	0.843002	0.842938	19.702959

Observation:

- After applying the Grid SearchCV, the optimized value of alpha results out to be 0.001.
- The performance metrics are showing similar results as compared to Least Squares method and Ridge Regression, implying that after adding the penalty, the model does not improve.

Elastic Net Regression

Elastic Net is a regularized regression model that combines L_1 and L_2 penalties, i.e., **lasso** and **ridge** regression. As a result, it performs a more efficient smoothing process.

- The elastic net includes the **penalty of lasso regression**, and **when used in isolation, it becomes the ridge regression**.
- In the procedure of regularization with an elastic net, **first, the coefficient of ridge regression is determined**.
- After this, a **lasso algorithm is performed on the ridge regression coefficient to shrink the coefficient**.
- It has two parameters to be set, α_1 and α_2 where α_1 controls the L_1 penalty and α_2 controls the L_2 penalty.

Instead of utilising two α-parameters, we can use simply one α and one L_1-ratio-parameter, which sets the proportion of our L_1 penalty in relation to α. If $\alpha = 1$ and L_1-ratio = 0.3, our L_1 penalty is multiplied by 0.3, and our L_2 penalty is multiplied by $1 - L_1 - ratio = 0.7$.

$$ElasticNetMSE = MSE(y, y_{pred}) + \alpha \cdot (1 - L_1 Ratio) \sum_{i=1}^{m} |\theta_i| + \alpha \cdot L$$

Elastic Net Regression with default parameters

In [48]:
```
elasticnet_model = ElasticNet()
elasticnet_model.fit(x_train, y_train)
```

Out [48]:

```
▾ ElasticNet
ElasticNet()
```

In [49]:

```
elasticnet_reg = model_performance_regression(elasticnet_model, x_test, y_te
elasticnet_reg
```

Out [49]:

	RMSE	MAE	R-squared	Adj. R-squared	MAPE
0	6.556087	4.678504	0.317455	0.317175	40.121657

Observations:

- After fitting the data into Elastic Net Model with default value of alpha (=1) and l1_ratio, the performance metrics are showing poor results as compared to Least Squares method and Ridge Regression.
- We can tune the alpha to get the optimized value similar to Ridge Regression using Grid SearchCV.

Elastic Net Regression with optimized α and $L_1 - ratio$

In [50]:

```
folds = KFold(n_splits=10, shuffle=True, random_state=1)
params = {'alpha':[0.001, 0.01, 0.1, 0.2, 0.5, 0.9],
         'l1_ratio': [0.001, 0.01, 0.02, 0.03, 0.04, 0.05]}
model = ElasticNet()
model_cv = GridSearchCV(estimator=model, param_grid=params, scoring='r2', cv
model_cv.fit(x_train,y_train)
```

Out [50]:

```
▸ GridSearchCV
▸ best_estimator_:
  ElasticNet
    ▸ ElasticNet
```

In [51]:

```
model_cv.best_params_
```

Out [51]:

```
{'alpha': 0.001, 'l1_ratio': 0.05}
```

In [52]:

```
elasticnet_model_tuned = ElasticNet(alpha=0.001, l1_ratio=0.05)
elasticnet_model_tuned.fit(x_train, y_train)
```

Out [52]:

```
▾ ElasticNet
ElasticNet(alpha=0.001, l1_ratio=0.05)
```

In [53]:
```
elasticnet_reg_tuned = model_performance_regression(elasticnet_model_tuned,
elasticnet_reg_tuned
```

Out[53]:

	RMSE	MAE	R-squared	Adj. R-squared	MAPE
0	3.157478	2.178911	0.841685	0.84162	19.981572

Observation

- After applying the Grid SearchCV, the optimized value of alpha results out to be 0.001, and l1_ratio = 0.05.
- The performance metrics are showing almost similar results as compared to Least Squares method, Ridge Regression and Lasso Regression, implying that after tuning the Elastic Net, the model does not improve.

In [54]:
```
models= pd.concat([linear_reg_test,ridge_reg,ridge_reg_tuned,lasso_reg,lassc
                   elasticnet_reg_tuned], axis=0) #combining all models intc
models['Models'] = ['Least Squares', 'Ridge Regression', 'Ridge Regression T
                                  'Lasso Regression Tuned', 'Elastic Net
                  'Elastic Net Regression Tuned'] #adding names of the moc
models = models.iloc[:,[5, 0,1,2,3,4]] #ordering names of the models as the
models
```

Out[54]:

	Models	RMSE	MAE	R-squared	Adj. R-squared	MAPE
0	Least Squares	3.144055	2.155765	0.843028	0.842964	19.676966
0	Ridge Regression	3.144057	2.155826	0.843028	0.842963	19.677968
0	Ridge Regression Tuned	3.144055	2.155771	0.843028	0.842964	19.677066
0	Lasso Regression	6.064339	3.873332	0.416006	0.415766	34.652716
0	Lasso Regression Tuned	3.144315	2.157198	0.843002	0.842938	19.702959
0	Elastic Net Regression	6.556087	4.678504	0.317455	0.317175	40.121657
0	Elastic Net Regression Tuned	3.157478	2.178911	0.841685	0.841620	19.981572

Observations:

- As per the above result, the **Least Squares Method** is giving the best results as compared to other models.
- Regularization technique does not offer any significant improvement to the performance metrics.
- So, we will apply some **Non Linear models** to check if the model performance improves or not.

Forward Feature Selection using SequentialFeatureSelector

We will see how to use **SequentialFeatureSelector** to select a subset of key features using forward feature selection. It is a greedy search algorithm that is used to **reduce an initial d-dimensional feature space to a k-dimensional feature subspace where k < d**. It is useful to automatically select a subset of the most relevant featuresthat are most relevant to the problem.

Why should we do feature selection?

- Reduces dimensionality
- Discards deceptive features; Deceptive features appear to aid learning on the training set but impair generalization
- Speeds training/testing

How does forward feature selection work?

- It starts with an empty model and adds variables one by one.
- In each forward step, you add the one variable that gives the highest improvement to your model.

We will use forward feature selection on all the variables.

SFS.png

In [55]:
```
# Installing mlxtend library. You need to run the below code only once if ml

!pip install mlxtend
```

```
Requirement already satisfied: mlxtend in /Users/obaozai/miniconda3/envs/jup
yter/lib/python3.11/site-packages (0.23.3)
Requirement already satisfied: scipy>=1.2.1 in /Users/obaozai/miniconda3/env
s/jupyter/lib/python3.11/site-packages (from mlxtend) (1.14.1)
Requirement already satisfied: numpy>=1.16.2 in /Users/obaozai/miniconda3/en
vs/jupyter/lib/python3.11/site-packages (from mlxtend) (1.23.5)
Requirement already satisfied: pandas>=0.24.2 in /Users/obaozai/miniconda3/e
nvs/jupyter/lib/python3.11/site-packages (from mlxtend) (2.2.3)
Requirement already satisfied: scikit-learn>=1.3.1 in /Users/obaozai/minicon
da3/envs/jupyter/lib/python3.11/site-packages (from mlxtend) (1.6.1)
Requirement already satisfied: matplotlib>=3.0.0 in /Users/obaozai/miniconda
3/envs/jupyter/lib/python3.11/site-packages (from mlxtend) (3.10.0)
Requirement already satisfied: joblib>=0.13.2 in /Users/obaozai/miniconda3/e
nvs/jupyter/lib/python3.11/site-packages (from mlxtend) (1.4.2)
Requirement already satisfied: contourpy>=1.0.1 in /Users/obaozai/miniconda
3/envs/jupyter/lib/python3.11/site-packages (from matplotlib>=3.0.0->mlxten
d) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /Users/obaozai/miniconda3/env
s/jupyter/lib/python3.11/site-packages (from matplotlib>=3.0.0->mlxtend) (0.
11.0)
Requirement already satisfied: fonttools>=4.22.0 in /Users/obaozai/miniconda
3/envs/jupyter/lib/python3.11/site-packages (from matplotlib>=3.0.0->mlxten
d) (4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /Users/obaozai/miniconda
3/envs/jupyter/lib/python3.11/site-packages (from matplotlib>=3.0.0->mlxten
d) (1.4.4)
Requirement already satisfied: packaging>=20.0 in /Users/obaozai/miniconda3/
envs/jupyter/lib/python3.11/site-packages (from matplotlib>=3.0.0->mlxtend)
(24.1)
Requirement already satisfied: pillow>=8 in /Users/obaozai/miniconda3/envs/j
upyter/lib/python3.11/site-packages (from matplotlib>=3.0.0->mlxtend) (11.0.
0)
Requirement already satisfied: pyparsing>=2.3.1 in /Users/obaozai/miniconda
3/envs/jupyter/lib/python3.11/site-packages (from matplotlib>=3.0.0->mlxten
d) (3.2.0)
Requirement already satisfied: python-dateutil>=2.7 in /Users/obaozai/minico
nda3/envs/jupyter/lib/python3.11/site-packages (from matplotlib>=3.0.0->mlxt
end) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /Users/obaozai/miniconda3/env
s/jupyter/lib/python3.11/site-packages (from pandas>=0.24.2->mlxtend) (2024.
1)
Requirement already satisfied: tzdata>=2022.7 in /Users/obaozai/miniconda3/e
nvs/jupyter/lib/python3.11/site-packages (from pandas>=0.24.2->mlxtend) (202
3.3)
Requirement already satisfied: threadpoolctl>=3.1.0 in /Users/obaozai/minico
nda3/envs/jupyter/lib/python3.11/site-packages (from scikit-learn>=1.3.1->ml
xtend) (3.5.0)
Requirement already satisfied: six>=1.5 in /Users/obaozai/miniconda3/envs/ju
pyter/lib/python3.11/site-packages (from python-dateutil>=2.7->matplotlib>=
3.0.0->mlxtend) (1.16.0)
```

In [56]:
```
# Importing Sequential Feature Selector
from mlxtend.feature_selection import SequentialFeatureSelector as SFS
```

Parameters to pass in SequentialFeatureSelector:

- **estimator:** scikit-learn classifier or regressor.
- **k_features:** int or tuple or str (default: 1).
  - The number of features to choose, where k features equals the entire feature collection, can be specified as an integer.
  - The SFS will consider returning any feature combination between min and max that scored highest in cross-validation if a tuple containing a min and max value is provided. For example, instead of a set amount of characteristics k, the tuple (1, 4) will return any combination of 1 to 4 features.
  - A string argument such as "best" or "parsimonious". If you choose "best," the feature selector will provide the feature subset with the best cross-validation performance. If the input "parsimonious" is provided, the smallest feature subset that is within one standard error of the cross-validation performance will be chosen.
- **forward:** bool (default: True). Forward selection if True, backward selection otherwise.
- **floating:** bool (default: False). Adds a conditional exclusion/inclusion if True:
  - Sequential floating forward selection (SFFS) starts from the empty set.
  - After each forward step, it performs backward steps as long as the objective function increases.
  - Once it stops increasing, the forward selection is continued.
- **verbose:** int (default: 0), level of verbosity to use in logging. If 0 then no output, if 1then the number of features in the current set, and if 2 then detailed logging including timestamp and cv scores at each step.
- **scoring:** str, callable, or None (default: None). If None (default), uses 'accuracy' for sklearn classifiers and 'r2' for sklearn regressors.
- **cv:** int (default: 5). Integer or iterable yielding train, test splits. If cv is an integer and estimator is a classifier (or y consists of integer class labels) stratified k-fold. Otherwise, regular k-fold cross-validation is performed. No cross-validation if cv is None, False, or 0.
- **n_jobs:** int (default: 1). The number of CPUs to use for evaluating different feature subsets in parallel. -1 means 'all CPUs'.

In [57]:
```
# Initializing the model to pass to SFS
reg = LinearRegression()

# Forward Feature Selection
sfs = SFS(
    reg,
    k_features=x_train.shape[1],
    forward=True,
```

```
    floating=False,
    scoring="r2",
    n_jobs=-1,
    verbose=2,
    cv=5,
)

# Perform SFS
sfs = sfs.fit(x_train, y_train)
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done  13 tasks      | elapsed:    3.8s
[Parallel(n_jobs=-1)]: Done  35 out of  41 | elapsed:    8.7s remaining:
1.5s
[Parallel(n_jobs=-1)]: Done  41 out of  41 | elapsed:   10.1s finished

[2025-01-24 22:21:45] Features: 1/41 -- score: 0.4918898861031266[Parallel(n
_jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done  13 tasks      | elapsed:    3.6s
[Parallel(n_jobs=-1)]: Done  34 out of  40 | elapsed:    8.4s remaining:
1.5s
[Parallel(n_jobs=-1)]: Done  40 out of  40 | elapsed:    9.6s finished

[2025-01-24 22:21:55] Features: 2/41 -- score: 0.6046160397618205[Parallel(n
_jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done  32 out of  39 | elapsed:    8.2s remaining:
1.8s
[Parallel(n_jobs=-1)]: Done  39 out of  39 | elapsed:    9.7s finished

[2025-01-24 22:22:04] Features: 3/41 -- score: 0.646190914266793[Parallel(n_
jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done  31 out of  38 | elapsed:    7.9s remaining:
1.8s
[Parallel(n_jobs=-1)]: Done  38 out of  38 | elapsed:    9.5s finished

[2025-01-24 22:22:14] Features: 4/41 -- score: 0.7013054914237962[Parallel(n
_jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done  29 out of  37 | elapsed:    7.8s remaining:
2.1s
[Parallel(n_jobs=-1)]: Done  37 out of  37 | elapsed:    9.5s finished

[2025-01-24 22:22:24] Features: 5/41 -- score: 0.7323069421611099[Parallel(n
_jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 out of  36 | elapsed:    7.5s remaining:
2.1s
[Parallel(n_jobs=-1)]: Done  36 out of  36 | elapsed:    9.2s finished

[2025-01-24 22:22:33] Features: 6/41 -- score: 0.8191351388509375[Parallel(n
_jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done  26 out of  35 | elapsed:    7.1s remaining:
2.4s
[Parallel(n_jobs=-1)]: Done  35 out of  35 | elapsed:    9.2s finished

[2025-01-24 22:22:42] Features: 7/41 -- score: 0.830328286206495[Parallel(n_
jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done  25 out of  34 | elapsed:    6.8s remaining:
2.5s
[Parallel(n_jobs=-1)]: Done  34 out of  34 | elapsed:    8.9s finished

[2025-01-24 22:22:51] Features: 8/41 -- score: 0.8395075505200218[Parallel(n
_jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done  23 out of  33 | elapsed:    6.5s remaining:
2.8s
[Parallel(n_jobs=-1)]: Done  33 out of  33 | elapsed:    8.7s finished

[2025-01-24 22:23:00] Features: 9/41 -- score: 0.8406253593745706[Parallel(n
```

```
_jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done  22 out of  32 | elapsed:    6.3s remaining:
2.9s
[Parallel(n_jobs=-1)]: Done  32 out of  32 | elapsed:    8.5s finished

[2025-01-24 22:23:09] Features: 10/41 -- score: 0.841460043255142[Parallel(n
_jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done  20 out of  31 | elapsed:    6.1s remaining:
3.4s
[Parallel(n_jobs=-1)]: Done  31 out of  31 | elapsed:    8.5s finished

[2025-01-24 22:23:17] Features: 11/41 -- score: 0.842217272243895[Parallel(n
_jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done  19 out of  30 | elapsed:    5.9s remaining:
3.4s
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed:    8.4s finished

[2025-01-24 22:23:26] Features: 12/41 -- score: 0.8423261376358362[Parallel
(n_jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 out of  29 | elapsed:    5.8s remaining:
4.1s
[Parallel(n_jobs=-1)]: Done  29 out of  29 | elapsed:    8.6s finished

[2025-01-24 22:23:35] Features: 13/41 -- score: 0.8423952895273406[Parallel
(n_jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done  16 out of  28 | elapsed:    5.7s remaining:
4.3s
[Parallel(n_jobs=-1)]: Done  28 out of  28 | elapsed:    8.5s finished

[2025-01-24 22:23:43] Features: 14/41 -- score: 0.8424554518867546[Parallel
(n_jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done  14 out of  27 | elapsed:    5.3s remaining:
4.9s
[Parallel(n_jobs=-1)]: Done  27 out of  27 | elapsed:    8.5s finished

[2025-01-24 22:23:52] Features: 15/41 -- score: 0.8425213613275838[Parallel
(n_jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done  13 out of  26 | elapsed:    5.4s remaining:
5.4s
[Parallel(n_jobs=-1)]: Done  26 out of  26 | elapsed:    8.5s finished

[2025-01-24 22:24:00] Features: 16/41 -- score: 0.8425660490400713[Parallel
(n_jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done  11 out of  25 | elapsed:    5.0s remaining:
6.4s
[Parallel(n_jobs=-1)]: Done  25 out of  25 | elapsed:    8.5s finished

[2025-01-24 22:24:09] Features: 17/41 -- score: 0.8426615241241718[Parallel
(n_jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done  10 out of  24 | elapsed:    4.9s remaining:
6.8s
[Parallel(n_jobs=-1)]: Done  24 out of  24 | elapsed:    8.5s finished

[2025-01-24 22:24:18] Features: 18/41 -- score: 0.8426866078058491[Parallel
(n_jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done   8 out of  23 | elapsed:    4.6s remaining:
```

```
8.7s
[Parallel(n_jobs=-1)]: Done  20 out of  23 | elapsed:    7.9s remaining:    
1.2s
[Parallel(n_jobs=-1)]: Done  23 out of  23 | elapsed:    8.6s finished

[2025-01-24 22:24:26] Features: 19/41 -- score: 0.8427094423867043[Parallel
(n_jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done   7 out of  22 | elapsed:    4.4s remaining:    
9.5s
[Parallel(n_jobs=-1)]: Done  19 out of  22 | elapsed:    7.8s remaining:    
1.2s
[Parallel(n_jobs=-1)]: Done  22 out of  22 | elapsed:    8.7s finished

[2025-01-24 22:24:35] Features: 20/41 -- score: 0.8427304449567397[Parallel
(n_jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done   5 out of  21 | elapsed:    4.5s remaining:   1
4.4s
[Parallel(n_jobs=-1)]: Done  16 out of  21 | elapsed:    7.4s remaining:    
2.3s
[Parallel(n_jobs=-1)]: Done  21 out of  21 | elapsed:    8.7s finished

[2025-01-24 22:24:44] Features: 21/41 -- score: 0.8427404093833673[Parallel
(n_jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done   4 out of  20 | elapsed:    4.2s remaining:   1
6.9s
[Parallel(n_jobs=-1)]: Done  15 out of  20 | elapsed:    7.3s remaining:    
2.4s
[Parallel(n_jobs=-1)]: Done  20 out of  20 | elapsed:    8.5s finished

[2025-01-24 22:24:52] Features: 22/41 -- score: 0.8427505409320879[Parallel
(n_jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done   2 out of  19 | elapsed:    3.5s remaining:   2
9.5s
[Parallel(n_jobs=-1)]: Done  12 out of  19 | elapsed:    6.6s remaining:    
3.8s
[Parallel(n_jobs=-1)]: Done  19 out of  19 | elapsed:    8.3s finished

[2025-01-24 22:25:01] Features: 23/41 -- score: 0.8427572335567[Parallel(n_j
obs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done  11 out of  18 | elapsed:    6.3s remaining:    
4.0s
[Parallel(n_jobs=-1)]: Done  18 out of  18 | elapsed:    8.9s finished

[2025-01-24 22:25:10] Features: 24/41 -- score: 0.8427691076570655[Parallel
(n_jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done   8 out of  17 | elapsed:    6.3s remaining:    
7.0s
[Parallel(n_jobs=-1)]: Done  17 out of  17 | elapsed:    8.1s finished

[2025-01-24 22:25:18] Features: 25/41 -- score: 0.8427740700470843[Parallel
(n_jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done   7 out of  16 | elapsed:    5.8s remaining:    
7.5s
[Parallel(n_jobs=-1)]: Done  16 out of  16 | elapsed:    8.1s finished

[2025-01-24 22:25:26] Features: 26/41 -- score: 0.8427758697137149[Parallel
```

```
(n_jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done   4 out of  15 | elapsed:    5.2s remaining:   1
4.2s
[Parallel(n_jobs=-1)]: Done  12 out of  15 | elapsed:    7.1s remaining:
1.8s
[Parallel(n_jobs=-1)]: Done  15 out of  15 | elapsed:    8.0s finished

[2025-01-24 22:25:34] Features: 27/41 -- score: 0.8427778696285013[Parallel
(n_jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done   3 out of  14 | elapsed:    4.9s remaining:   1
8.1s
[Parallel(n_jobs=-1)]: Done  11 out of  14 | elapsed:    7.1s remaining:
1.9s
[Parallel(n_jobs=-1)]: Done  14 out of  14 | elapsed:    7.8s finished

[2025-01-24 22:25:42] Features: 28/41 -- score: 0.8427791238823727[Parallel
(n_jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done   7 out of  13 | elapsed:    6.3s remaining:
5.4s
[Parallel(n_jobs=-1)]: Done  13 out of  13 | elapsed:    7.6s finished

[2025-01-24 22:25:50] Features: 29/41 -- score: 0.8427807461672205[Parallel
(n_jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done   6 out of  12 | elapsed:    5.9s remaining:
5.9s
[Parallel(n_jobs=-1)]: Done  12 out of  12 | elapsed:    7.3s finished

[2025-01-24 22:25:57] Features: 30/41 -- score: 0.8427809322153685[Parallel
(n_jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done   2 out of  11 | elapsed:    4.6s remaining:   2
0.6s
[Parallel(n_jobs=-1)]: Done   8 out of  11 | elapsed:    6.2s remaining:
2.3s
[Parallel(n_jobs=-1)]: Done  11 out of  11 | elapsed:    6.9s finished

[2025-01-24 22:26:04] Features: 31/41 -- score: 0.8427809322153685[Parallel
(n_jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done   7 out of  10 | elapsed:    6.0s remaining:
2.6s
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:    6.7s finished

[2025-01-24 22:26:11] Features: 32/41 -- score: 0.8427809322153685[Parallel
(n_jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done   2 out of   9 | elapsed:    4.7s remaining:   1
6.5s
[Parallel(n_jobs=-1)]: Done   7 out of   9 | elapsed:    6.0s remaining:
1.7s
[Parallel(n_jobs=-1)]: Done   9 out of   9 | elapsed:    6.4s finished

[2025-01-24 22:26:18] Features: 33/41 -- score: 0.8427803380446244[Parallel
(n_jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done   6 out of   8 | elapsed:    5.7s remaining:
1.9s
[Parallel(n_jobs=-1)]: Done   8 out of   8 | elapsed:    6.2s finished

[2025-01-24 22:26:24] Features: 34/41 -- score: 0.8427796949690372[Parallel
```

```
(n_jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done   4 out of   7 | elapsed:    5.4s remaining:
4.1s
[Parallel(n_jobs=-1)]: Done   7 out of   7 | elapsed:    6.0s finished

[2025-01-24 22:26:30] Features: 35/41 -- score: 0.8427789615641776[Parallel
(n_jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done   3 out of   6 | elapsed:    4.5s remaining:
4.5s
[Parallel(n_jobs=-1)]: Done   6 out of   6 | elapsed:    6.0s finished

[2025-01-24 22:26:36] Features: 36/41 -- score: 0.8427779197388698[Parallel
(n_jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done   2 out of   5 | elapsed:    4.6s remaining:
6.9s
[Parallel(n_jobs=-1)]: Done   5 out of   5 | elapsed:    5.5s finished

[2025-01-24 22:26:42] Features: 37/41 -- score: 0.8427759963463547[Parallel
(n_jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done   4 out of   4 | elapsed:    5.1s finished

[2025-01-24 22:26:47] Features: 38/41 -- score: 0.8427733960444744[Parallel
(n_jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done   3 out of   3 | elapsed:    5.9s finished

[2025-01-24 22:26:53] Features: 39/41 -- score: 0.8427687463538682[Parallel
(n_jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done   2 out of   2 | elapsed:    4.9s finished

[2025-01-24 22:26:58] Features: 40/41 -- score: 0.8427687463538682[Parallel
(n_jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.

[2025-01-24 22:27:03] Features: 41/41 -- score: 0.8427687463538682
```

Now, let's plot the the model performance with addition of each feature. We will use the **plot_sequential_feature_selection** function for this. It has the following parameters:

- **metric_dict:** mlxtend.SequentialFeatureSelector.get_metric_dict() object, which is a dictionary with items where each dictionary value is a list with the number of iterations (number of feature subsets) as its length. The **dictionary keys** corresponding to these lists are as follows:
  - 'feature_idx': tuple of the indices of the feature subset
  - 'cv_scores': list with individual CV scores
  - 'avg_score': of CV average scores
  - 'feature_names': Name of features in the subset
  - 'ci_bound': confidence interval bound of the CV score average
  - 'std_dev': standard deviation of the CV score average
  - 'std_err': standard error of the CV score average
- **figsize:** tuple (default: None). Height and width of the figure.

- **kind:** str (default: "std_dev"). The kind of error bar or confidence interval in {'std_dev', 'std_err', 'ci', None}.
- **color:** str (default: "blue"). Color of the lineplot (accepts any matplotlib color name).
- **bcolor:** str (default: "steelblue"). Color of the error bars / confidence intervals (accepts any matplotlib color name).
- **marker:** str (default: "o"). Marker of the line plot (accepts any matplotlib marker name).
- **alpha:** float in [0, 1] (default: 0.2). Transparency of the error bars / confidence intervals.
- **ylabel:** str (default: "Performance"). Y-axis label.
- **confidence_interval:** float (default: 0.95). Confidence level if kind='ci'.

In [58]:
```
sfs.get_metric_dict()
```

Out[58]:
```
{1: {'feature_idx': (5,),
  'cv_scores': array([0.48930483, 0.48939957, 0.49302582, 0.48922331, 0.498
4959 ]),
  'avg_score': 0.4918898861031266,
  'feature_names': ('Department_gynecology',),
  'ci_bound': 0.004631493099881328,
  'std_dev': 0.003603458980122188,
  'std_err': 0.001801729490061094},
 2: {'feature_idx': (5, 6),
  'cv_scores': array([0.60393142, 0.60360277, 0.60160635, 0.60347042, 0.610
46924]),
  'avg_score': 0.6046160397618205,
  'feature_names': ('Department_gynecology', 'Department_radiotherapy'),
  'ci_bound': 0.003903818903767414,
  'std_dev': 0.0030373037338459795,
  'std_err': 0.0015186518669229895},
 3: {'feature_idx': (5, 6, 23),
  'cv_scores': array([0.64519204, 0.6466788 , 0.64365579, 0.64406087, 0.651
36707]),
  'avg_score': 0.646190914266793,
  'feature_names': ('Department_gynecology',
   'Department_radiotherapy',
   'Age_31-40'),
  'ci_bound': 0.0035892679591376106,
  'std_dev': 0.0027925724125014164,
  'std_err': 0.0013962862062507082},
 4: {'feature_idx': (5, 6, 23, 24),
  'cv_scores': array([0.70130514, 0.70174535, 0.69909054, 0.70011695, 0.704
26948]),
  'avg_score': 0.7013054914237962,
  'feature_names': ('Department_gynecology',
   'Department_radiotherapy',
   'Age_31-40',
   'Age_41-50'),
  'ci_bound': 0.0022481465709767914,
  'std_dev': 0.0017491344098137162,
  'std_err': 0.000874567204906858},
 5: {'feature_idx': (5, 6, 22, 23, 24),
  'cv_scores': array([0.73232311, 0.73311883, 0.73025991, 0.73047301, 0.735
35986]),
  'avg_score': 0.7323069421611099,
  'feature_names': ('Department_gynecology',
   'Department_radiotherapy',
   'Age_21-30',
   'Age_31-40',
   'Age_41-50'),
  'ci_bound': 0.002406852284551526,
  'std_dev': 0.0018726128467765675,
  'std_err': 0.0009363064233882837},
 6: {'feature_idx': (5, 6, 21, 22, 23, 24),
  'cv_scores': array([0.81759505, 0.82082637, 0.81776443, 0.81792308, 0.821
56676]),
  'avg_score': 0.8191351388509375,
  'feature_names': ('Department_gynecology',
   'Department_radiotherapy',
   'Age_11-20',
```

```
   'Age_21-30',
   'Age_31-40',
   'Age_41-50'),
  'ci_bound': 0.0021882349498266998,
  'std_dev': 0.00170252113314652,
  'std_err': 0.00085126056657326},
 7: {'feature_idx': (4, 5, 6, 21, 22, 23, 24),
  'cv_scores': array([0.82910903, 0.83188966, 0.82885568, 0.82915996, 0.832
6271 ]),
  'avg_score': 0.830328286206495,
  'feature_names': ('Department_anesthesia',
   'Department_gynecology',
   'Department_radiotherapy',
   'Age_11-20',
   'Age_21-30',
   'Age_31-40',
   'Age_41-50'),
  'ci_bound': 0.0020518485785454826,
  'std_dev': 0.00159640790275605,
  'std_err': 0.0007982039513780249},
 8: {'feature_idx': (4, 5, 6, 7, 21, 22, 23, 24),
  'cv_scores': array([0.83841525, 0.84068   , 0.83778767, 0.83893676, 0.841
71807]),
  'avg_score': 0.8395075505200218,
  'feature_names': ('Department_anesthesia',
   'Department_gynecology',
   'Department_radiotherapy',
   'Department_surgery',
   'Age_11-20',
   'Age_21-30',
   'Age_31-40',
   'Age_41-50'),
  'ci_bound': 0.0018835514583498865,
  'std_dev': 0.001465467025587721,
  'std_err': 0.0007327335127938605},
 9: {'feature_idx': (4, 5, 6, 7, 18, 21, 22, 23, 24),
  'cv_scores': array([0.8395841 , 0.84177861, 0.83891449, 0.84007992, 0.842
76968]),
  'avg_score': 0.8406253593745706,
  'feature_names': ('Department_anesthesia',
   'Department_gynecology',
   'Department_radiotherapy',
   'Department_surgery',
   'doctor_name_Dr Sarah',
   'Age_11-20',
   'Age_21-30',
   'Age_31-40',
   'Age_41-50'),
  'ci_bound': 0.001839080555394871,
  'std_dev': 0.0014308671522528134,
  'std_err': 0.0007154335761264066},
 10: {'feature_idx': (4, 5, 6, 7, 18, 21, 22, 23, 24, 37),
  'cv_scores': array([0.84042973, 0.84272043, 0.8397547 , 0.84087775, 0.843
51761]),
  'avg_score': 0.841460043255142,
  'feature_names': ('Department_anesthesia',
```

```
   'Department_gynecology',
   'Department_radiotherapy',
   'Department_surgery',
   'doctor_name_Dr Sarah',
   'Age_11-20',
   'Age_21-30',
   'Age_31-40',
   'Age_41-50',
   'health_conditions_Heart disease'),
  'ci_bound': 0.0018295399644201768,
  'std_dev': 0.0014234442483464435,
  'std_err': 0.0007117221241732216},
 11: {'feature_idx': (0, 4, 5, 6, 7, 18, 21, 22, 23, 24, 37),
  'cv_scores': array([0.84122914, 0.84335217, 0.84047911, 0.84174549, 0.844
28045]),
  'avg_score': 0.842217272243895,
  'feature_names': ('Available Extra Rooms in Hospital',
   'Department_anesthesia',
   'Department_gynecology',
   'Department_radiotherapy',
   'Department_surgery',
   'doctor_name_Dr Sarah',
   'Age_11-20',
   'Age_21-30',
   'Age_31-40',
   'Age_41-50',
   'health_conditions_Heart disease'),
  'ci_bound': 0.0017961805749327953,
  'std_dev': 0.0013974895099872781,
  'std_err': 0.0006987447549936391},
 12: {'feature_idx': (0, 4, 5, 6, 7, 12, 18, 21, 22, 23, 24, 37),
  'cv_scores': array([0.84133146, 0.84346648, 0.84059185, 0.84183307, 0.844
40783]),
  'avg_score': 0.8423261376358362,
  'feature_names': ('Available Extra Rooms in Hospital',
   'Department_anesthesia',
   'Department_gynecology',
   'Department_radiotherapy',
   'Department_surgery',
   'Ward_Facility_Code_F',
   'doctor_name_Dr Sarah',
   'Age_11-20',
   'Age_21-30',
   'Age_31-40',
   'Age_41-50',
   'health_conditions_Heart disease'),
  'ci_bound': 0.0018061965534049654,
  'std_dev': 0.0014052822815173011,
  'std_err': 0.0007026411407586504},
 13: {'feature_idx': (0, 4, 5, 6, 7, 12, 18, 21, 22, 23, 24, 32, 37),
  'cv_scores': array([0.84143434, 0.84354144, 0.84068201, 0.84186707, 0.844
45158]),
  'avg_score': 0.8423952895273406,
  'feature_names': ('Available Extra Rooms in Hospital',
   'Department_anesthesia',
   'Department_gynecology',
```

```
   'Department_radiotherapy',
   'Department_surgery',
   'Ward_Facility_Code_F',
   'doctor_name_Dr Sarah',
   'Age_11-20',
   'Age_21-30',
   'Age_31-40',
   'Age_41-50',
   'Type of Admission_Trauma',
   'health_conditions_Heart disease'),
  'ci_bound': 0.0017883348227490802,
  'std_dev': 0.0013913852482400361,
  'std_err': 0.000695692624120018},
 14: {'feature_idx': (0, 4, 5, 6, 7, 11, 12, 18, 21, 22, 23, 24, 32, 37),
  'cv_scores': array([0.84149253, 0.84359962, 0.84074705, 0.84189538, 0.844
54268]),
  'avg_score': 0.8424554518867546,
  'feature_names': ('Available Extra Rooms in Hospital',
   'Department_anesthesia',
   'Department_gynecology',
   'Department_radiotherapy',
   'Department_surgery',
   'Ward_Facility_Code_E',
   'Ward_Facility_Code_F',
   'doctor_name_Dr Sarah',
   'Age_11-20',
   'Age_21-30',
   'Age_31-40',
   'Age_41-50',
   'Type of Admission_Trauma',
   'health_conditions_Heart disease'),
  'ci_bound': 0.001801719163516881,
  'std_dev': 0.0014017987200713952,
  'std_err': 0.0007008993600356975},
 15: {'feature_idx': (0, 4, 5, 6, 7, 9, 11, 12, 18, 21, 22, 23, 24, 32, 3
7),
  'cv_scores': array([0.84151756, 0.84365472, 0.84083538, 0.84196472, 0.844
63442]),
  'avg_score': 0.8425213613275838,
  'feature_names': ('Available Extra Rooms in Hospital',
   'Department_anesthesia',
   'Department_gynecology',
   'Department_radiotherapy',
   'Department_surgery',
   'Ward_Facility_Code_C',
   'Ward_Facility_Code_E',
   'Ward_Facility_Code_F',
   'doctor_name_Dr Sarah',
   'Age_11-20',
   'Age_21-30',
   'Age_31-40',
   'Age_41-50',
   'Type of Admission_Trauma',
   'health_conditions_Heart disease'),
  'ci_bound': 0.001809433627154751,
  'std_dev': 0.0014078008348696273,
```

```
  'std_err': 0.0007039004174348136},
 16: {'feature_idx': (0,
   4,
   5,
   6,
   7,
   9,
   11,
   12,
   18,
   20,
   21,
   22,
   23,
   24,
   32,
   37),
  'cv_scores': array([0.84157076, 0.84368731, 0.84088641, 0.84200681, 0.844
67896]),
  'avg_score': 0.8425660490400713,
  'feature_names': ('Available Extra Rooms in Hospital',
   'Department_anesthesia',
   'Department_gynecology',
   'Department_radiotherapy',
   'Department_surgery',
   'Ward_Facility_Code_C',
   'Ward_Facility_Code_E',
   'Ward_Facility_Code_F',
   'doctor_name_Dr Sarah',
   'doctor_name_Dr Sophia',
   'Age_11-20',
   'Age_21-30',
   'Age_31-40',
   'Age_41-50',
   'Type of Admission_Trauma',
   'health_conditions_Heart disease'),
  'ci_bound': 0.0018036427267596614,
  'std_dev': 0.0014032953176246132,
  'std_err': 0.0007016476588123066},
 17: {'feature_idx': (0,
   4,
   5,
   6,
   7,
   9,
   11,
   12,
   18,
   20,
   21,
   22,
   23,
   24,
   32,
   36,
   37),
```

```
  'cv_scores': array([0.84163027, 0.84381662, 0.8409956 , 0.84209143, 0.844
77371]),
  'avg_score': 0.8426615241241718,
  'feature_names': ('Available Extra Rooms in Hospital',
   'Department_anesthesia',
   'Department_gynecology',
   'Department_radiotherapy',
   'Department_surgery',
   'Ward_Facility_Code_C',
   'Ward_Facility_Code_E',
   'Ward_Facility_Code_F',
   'doctor_name_Dr Sarah',
   'doctor_name_Dr Sophia',
   'Age_11-20',
   'Age_21-30',
   'Age_31-40',
   'Age_41-50',
   'Type of Admission_Trauma',
   'health_conditions_Diabetes',
   'health_conditions_Heart disease'),
  'ci_bound': 0.0018139772096602376,
  'std_dev': 0.0014113358964206724,
  'std_err': 0.0007056679482103362},
 18: {'feature_idx': (0,
   4,
   5,
   6,
   7,
   9,
   11,
   12,
   18,
   20,
   21,
   22,
   23,
   24,
   32,
   33,
   36,
   37),
  'cv_scores': array([0.84165314, 0.84383133, 0.84104803, 0.84211931, 0.844
78123]),
  'avg_score': 0.8426866078058491,
  'feature_names': ('Available Extra Rooms in Hospital',
   'Department_anesthesia',
   'Department_gynecology',
   'Department_radiotherapy',
   'Department_surgery',
   'Ward_Facility_Code_C',
   'Ward_Facility_Code_E',
   'Ward_Facility_Code_F',
   'doctor_name_Dr Sarah',
   'doctor_name_Dr Sophia',
   'Age_11-20',
   'Age_21-30',
```

```
   'Age_31-40',
   'Age_41-50',
   'Type of Admission_Trauma',
   'Type of Admission_Urgent',
   'health_conditions_Diabetes',
   'health_conditions_Heart disease'),
  'ci_bound': 0.0017968945250524768,
  'std_dev': 0.001398044987435834,
  'std_err': 0.0006990224937179169},
 19: {'feature_idx': (0,
   3,
   4,
   5,
   6,
   7,
   9,
   11,
   12,
   18,
   20,
   21,
   22,
   23,
   24,
   32,
   33,
   36,
   37),
  'cv_scores': array([0.84165556, 0.84387059, 0.8410869 , 0.84214295, 0.844
79122]),
  'avg_score': 0.8427094423867043,
  'feature_names': ('Available Extra Rooms in Hospital',
   'Admission_Deposit',
   'Department_anesthesia',
   'Department_gynecology',
   'Department_radiotherapy',
   'Department_surgery',
   'Ward_Facility_Code_C',
   'Ward_Facility_Code_E',
   'Ward_Facility_Code_F',
   'doctor_name_Dr Sarah',
   'doctor_name_Dr Sophia',
   'Age_11-20',
   'Age_21-30',
   'Age_31-40',
   'Age_41-50',
   'Type of Admission_Trauma',
   'Type of Admission_Urgent',
   'health_conditions_Diabetes',
   'health_conditions_Heart disease'),
  'ci_bound': 0.0017944661327931836,
  'std_dev': 0.0013961556157569182,
  'std_err': 0.0006980778078784591},
 20: {'feature_idx': (0,
   3,
   4,
```

```
    5,
    6,
    7,
    9,
    11,
    12,
    18,
    20,
    21,
    22,
    23,
    24,
    28,
    32,
    33,
    36,
    37),
   'cv_scores': array([0.84169653, 0.84389602, 0.84109857, 0.84214173, 0.844
81937]),
   'avg_score': 0.8427304449567397,
   'feature_names': ('Available Extra Rooms in Hospital',
    'Admission_Deposit',
    'Department_anesthesia',
    'Department_gynecology',
    'Department_radiotherapy',
    'Department_surgery',
    'Ward_Facility_Code_C',
    'Ward_Facility_Code_E',
    'Ward_Facility_Code_F',
    'doctor_name_Dr Sarah',
    'doctor_name_Dr Sophia',
    'Age_11-20',
    'Age_21-30',
    'Age_31-40',
    'Age_41-50',
    'Age_81-90',
    'Type of Admission_Trauma',
    'Type of Admission_Urgent',
    'health_conditions_Diabetes',
    'health_conditions_Heart disease'),
   'ci_bound': 0.0017994745120910982,
   'std_dev': 0.0014000523050032847,
   'std_err': 0.0007000261525016424},
 21: {'feature_idx': (0,
    3,
    4,
    5,
    6,
    7,
    9,
    11,
    12,
    18,
    20,
    21,
    22,
```

```
   23,
   24,
   26,
   28,
   32,
   33,
   36,
   37),
  'cv_scores': array([0.84171921, 0.84389189, 0.84112059, 0.84214082, 0.844
82953]),
  'avg_score': 0.8427404093833673,
  'feature_names': ('Available Extra Rooms in Hospital',
   'Admission_Deposit',
   'Department_anesthesia',
   'Department_gynecology',
   'Department_radiotherapy',
   'Department_surgery',
   'Ward_Facility_Code_C',
   'Ward_Facility_Code_E',
   'Ward_Facility_Code_F',
   'doctor_name_Dr Sarah',
   'doctor_name_Dr Sophia',
   'Age_11-20',
   'Age_21-30',
   'Age_31-40',
   'Age_41-50',
   'Age_61-70',
   'Age_81-90',
   'Type of Admission_Trauma',
   'Type of Admission_Urgent',
   'health_conditions_Diabetes',
   'health_conditions_Heart disease'),
  'ci_bound': 0.0017917269538227866,
  'std_dev': 0.0013940244414582257,
  'std_err': 0.0006970122207291128},
 22: {'feature_idx': (0,
   3,
   4,
   5,
   6,
   7,
   9,
   11,
   12,
   18,
   20,
   21,
   22,
   23,
   24,
   25,
   26,
   28,
   32,
   33,
   36,
```

```
  37),
 'cv_scores': array([0.84172767, 0.84391208, 0.84112203, 0.84215577, 0.844
83515]),
 'avg_score': 0.8427505409320879,
 'feature_names': ('Available Extra Rooms in Hospital',
  'Admission_Deposit',
  'Department_anesthesia',
  'Department_gynecology',
  'Department_radiotherapy',
  'Department_surgery',
  'Ward_Facility_Code_C',
  'Ward_Facility_Code_E',
  'Ward_Facility_Code_F',
  'doctor_name_Dr Sarah',
  'doctor_name_Dr Sophia',
  'Age_11-20',
  'Age_21-30',
  'Age_31-40',
  'Age_41-50',
  'Age_51-60',
  'Age_61-70',
  'Age_81-90',
  'Type of Admission_Trauma',
  'Type of Admission_Urgent',
  'health_conditions_Diabetes',
  'health_conditions_Heart disease'),
 'ci_bound': 0.0017945182669480528,
 'std_dev': 0.0013961961779005905,
 'std_err': 0.0006980980889502952},
23: {'feature_idx': (0,
  3,
  4,
  5,
  6,
  7,
  9,
  11,
  12,
  15,
  18,
  20,
  21,
  22,
  23,
  24,
  25,
  26,
  28,
  32,
  33,
  36,
  37),
 'cv_scores': array([0.84173583, 0.84392189, 0.84113036, 0.84216222, 0.844
83587]),
 'avg_score': 0.8427572335567,
 'feature_names': ('Available Extra Rooms in Hospital',
```

```
   'Admission_Deposit',
   'Department_anesthesia',
   'Department_gynecology',
   'Department_radiotherapy',
   'Department_surgery',
   'Ward_Facility_Code_C',
   'Ward_Facility_Code_E',
   'Ward_Facility_Code_F',
   'doctor_name_Dr Nathan',
   'doctor_name_Dr Sarah',
   'doctor_name_Dr Sophia',
   'Age_11-20',
   'Age_21-30',
   'Age_31-40',
   'Age_41-50',
   'Age_51-60',
   'Age_61-70',
   'Age_81-90',
   'Type of Admission_Trauma',
   'Type of Admission_Urgent',
   'health_conditions_Diabetes',
   'health_conditions_Heart disease'),
  'ci_bound': 0.0017921537617258982,
  'std_dev': 0.0013943565125070403,
  'std_err': 0.00069717825625352},
 24: {'feature_idx': (0,
   3,
   4,
   5,
   6,
   7,
   9,
   11,
   12,
   15,
   18,
   20,
   21,
   22,
   23,
   24,
   25,
   26,
   28,
   32,
   33,
   35,
   36,
   37),
  'cv_scores': array([0.84174792, 0.84392579, 0.84114267, 0.84217859, 0.844
85057]),
  'avg_score': 0.8427691076570655,
  'feature_names': ('Available Extra Rooms in Hospital',
   'Admission_Deposit',
   'Department_anesthesia',
   'Department_gynecology',
```

```
   'Department_radiotherapy',
   'Department_surgery',
   'Ward_Facility_Code_C',
   'Ward_Facility_Code_E',
   'Ward_Facility_Code_F',
   'doctor_name_Dr Nathan',
   'doctor_name_Dr Sarah',
   'doctor_name_Dr Sophia',
   'Age_11-20',
   'Age_21-30',
   'Age_31-40',
   'Age_41-50',
   'Age_51-60',
   'Age_61-70',
   'Age_81-90',
   'Type of Admission_Trauma',
   'Type of Admission_Urgent',
   'Severity of Illness_Moderate',
   'health_conditions_Diabetes',
   'health_conditions_Heart disease'),
  'ci_bound': 0.0017908723313871152,
  'std_dev': 0.0013933595161686873,
  'std_err': 0.0006966797580843436},
 25: {'feature_idx': (0,
   3,
   4,
   5,
   6,
   7,
   9,
   10,
   11,
   12,
   15,
   18,
   20,
   21,
   22,
   23,
   24,
   25,
   26,
   28,
   32,
   33,
   35,
   36,
   37),
  'cv_scores': array([0.84175575, 0.84392909, 0.84114694, 0.84217963, 0.844
85894]),
  'avg_score': 0.8427740700470843,
  'feature_names': ('Available Extra Rooms in Hospital',
   'Admission_Deposit',
   'Department_anesthesia',
   'Department_gynecology',
   'Department_radiotherapy',
```

```
    'Department_surgery',
    'Ward_Facility_Code_C',
    'Ward_Facility_Code_D',
    'Ward_Facility_Code_E',
    'Ward_Facility_Code_F',
    'doctor_name_Dr Nathan',
    'doctor_name_Dr Sarah',
    'doctor_name_Dr Sophia',
    'Age_11-20',
    'Age_21-30',
    'Age_31-40',
    'Age_41-50',
    'Age_51-60',
    'Age_61-70',
    'Age_81-90',
    'Type of Admission_Trauma',
    'Type of Admission_Urgent',
    'Severity of Illness_Moderate',
    'health_conditions_Diabetes',
    'health_conditions_Heart disease'),
   'ci_bound': 0.0017919222426922843,
   'std_dev': 0.0013941763828333575,
   'std_err': 0.0006970881914166787},
 26: {'feature_idx': (0,
    3,
    4,
    5,
    6,
    7,
    9,
    10,
    11,
    12,
    15,
    18,
    20,
    21,
    22,
    23,
    24,
    25,
    26,
    27,
    28,
    32,
    33,
    35,
    36,
    37),
   'cv_scores': array([0.84176058, 0.84392785, 0.84114221, 0.84218609, 0.844
86261]),
   'avg_score': 0.8427758697137149,
   'feature_names': ('Available Extra Rooms in Hospital',
    'Admission_Deposit',
    'Department_anesthesia',
    'Department_gynecology',
```

```
   'Department_radiotherapy',
   'Department_surgery',
   'Ward_Facility_Code_C',
   'Ward_Facility_Code_D',
   'Ward_Facility_Code_E',
   'Ward_Facility_Code_F',
   'doctor_name_Dr Nathan',
   'doctor_name_Dr Sarah',
   'doctor_name_Dr Sophia',
   'Age_11-20',
   'Age_21-30',
   'Age_31-40',
   'Age_41-50',
   'Age_51-60',
   'Age_61-70',
   'Age_71-80',
   'Age_81-90',
   'Type of Admission_Trauma',
   'Type of Admission_Urgent',
   'Severity of Illness_Moderate',
   'health_conditions_Diabetes',
   'health_conditions_Heart disease'),
  'ci_bound': 0.0017928808470780086,
  'std_dev': 0.0013949222096126764,
  'std_err': 0.0006974611048063382},
 27: {'feature_idx': (0,
   3,
   4,
   5,
   6,
   7,
   9,
   10,
   11,
   12,
   15,
   18,
   20,
   21,
   22,
   23,
   24,
   25,
   26,
   27,
   28,
   29,
   32,
   33,
   35,
   36,
   37),
  'cv_scores': array([0.84175977, 0.84393144, 0.8411444 , 0.84218713, 0.844
86661]),
  'avg_score': 0.8427778696285013,
  'feature_names': ('Available Extra Rooms in Hospital',
```

```
  'Admission_Deposit',
  'Department_anesthesia',
  'Department_gynecology',
  'Department_radiotherapy',
  'Department_surgery',
  'Ward_Facility_Code_C',
  'Ward_Facility_Code_D',
  'Ward_Facility_Code_E',
  'Ward_Facility_Code_F',
  'doctor_name_Dr Nathan',
  'doctor_name_Dr Sarah',
  'doctor_name_Dr Sophia',
  'Age_11-20',
  'Age_21-30',
  'Age_31-40',
  'Age_41-50',
  'Age_51-60',
  'Age_61-70',
  'Age_71-80',
  'Age_81-90',
  'Age_91-100',
  'Type of Admission_Trauma',
  'Type of Admission_Urgent',
  'Severity of Illness_Moderate',
  'health_conditions_Diabetes',
  'health_conditions_Heart disease'),
 'ci_bound': 0.0017945631286873577,
 'std_dev': 0.001396231081857884,
 'std_err': 0.0006981155409289419},
28: {'feature_idx': (0,
  3,
  4,
  5,
  6,
  7,
  9,
  10,
  11,
  12,
  15,
  18,
  20,
  21,
  22,
  23,
  24,
  25,
  26,
  27,
  28,
  29,
  32,
  33,
  35,
  36,
  37,
```

```
  38),
 'cv_scores': array([0.84175803, 0.84393421, 0.8411476 , 0.8421875 , 0.844
86829]),
 'avg_score': 0.8427791238823727,
 'feature_names': ('Available Extra Rooms in Hospital',
  'Admission_Deposit',
  'Department_anesthesia',
  'Department_gynecology',
  'Department_radiotherapy',
  'Department_surgery',
  'Ward_Facility_Code_C',
  'Ward_Facility_Code_D',
  'Ward_Facility_Code_E',
  'Ward_Facility_Code_F',
  'doctor_name_Dr Nathan',
  'doctor_name_Dr Sarah',
  'doctor_name_Dr Sophia',
  'Age_11-20',
  'Age_21-30',
  'Age_31-40',
  'Age_41-50',
  'Age_51-60',
  'Age_61-70',
  'Age_71-80',
  'Age_81-90',
  'Age_91-100',
  'Type of Admission_Trauma',
  'Type of Admission_Urgent',
  'Severity of Illness_Moderate',
  'health_conditions_Diabetes',
  'health_conditions_Heart disease',
  'health_conditions_High Blood Pressure'),
 'ci_bound': 0.001795122469620699,
 'std_dev': 0.0013966662681068388,
 'std_err': 0.0006983331340534194},
29: {'feature_idx': (0,
  3,
  4,
  5,
  6,
  7,
  9,
  10,
  11,
  12,
  15,
  18,
  20,
  21,
  22,
  23,
  24,
  25,
  26,
  27,
  28,
```

```
   29,
   32,
   33,
   34,
   35,
   36,
   37,
   38),
  'cv_scores': array([0.84176298, 0.84393899, 0.84113744, 0.84219395, 0.844
87037]),
  'avg_score': 0.8427807461672205,
  'feature_names': ('Available Extra Rooms in Hospital',
   'Admission_Deposit',
   'Department_anesthesia',
   'Department_gynecology',
   'Department_radiotherapy',
   'Department_surgery',
   'Ward_Facility_Code_C',
   'Ward_Facility_Code_D',
   'Ward_Facility_Code_E',
   'Ward_Facility_Code_F',
   'doctor_name_Dr Nathan',
   'doctor_name_Dr Sarah',
   'doctor_name_Dr Sophia',
   'Age_11-20',
   'Age_21-30',
   'Age_31-40',
   'Age_41-50',
   'Age_51-60',
   'Age_61-70',
   'Age_71-80',
   'Age_81-90',
   'Age_91-100',
   'Type of Admission_Trauma',
   'Type of Admission_Urgent',
   'Severity of Illness_Minor',
   'Severity of Illness_Moderate',
   'health_conditions_Diabetes',
   'health_conditions_Heart disease',
   'health_conditions_High Blood Pressure'),
  'ci_bound': 0.0017983700221936693,
  'std_dev': 0.001399192974339606,
  'std_err': 0.000699596487169803},
 30: {'feature_idx': (0,
   3,
   4,
   5,
   6,
   7,
   9,
   10,
   11,
   12,
   15,
   18,
   20,
```

```
   21,
   22,
   23,
   24,
   25,
   26,
   27,
   28,
   29,
   32,
   33,
   34,
   35,
   36,
   37,
   38,
   40),
  'cv_scores': array([0.84176214, 0.84393978, 0.84113795, 0.84219458, 0.844
87021]),
  'avg_score': 0.8427809322153685,
  'feature_names': ('Available Extra Rooms in Hospital',
   'Admission_Deposit',
   'Department_anesthesia',
   'Department_gynecology',
   'Department_radiotherapy',
   'Department_surgery',
   'Ward_Facility_Code_C',
   'Ward_Facility_Code_D',
   'Ward_Facility_Code_E',
   'Ward_Facility_Code_F',
   'doctor_name_Dr Nathan',
   'doctor_name_Dr Sarah',
   'doctor_name_Dr Sophia',
   'Age_11-20',
   'Age_21-30',
   'Age_31-40',
   'Age_41-50',
   'Age_51-60',
   'Age_61-70',
   'Age_71-80',
   'Age_81-90',
   'Age_91-100',
   'Type of Admission_Trauma',
   'Type of Admission_Urgent',
   'Severity of Illness_Minor',
   'Severity of Illness_Moderate',
   'health_conditions_Diabetes',
   'health_conditions_Heart disease',
   'health_conditions_High Blood Pressure',
   'Insurance_Yes'),
  'ci_bound': 0.0017984071192350256,
  'std_dev': 0.0013992218370981009,
  'std_err': 0.0006996109185490503},
 31: {'feature_idx': (0,
   3,
   4,
```

```
   5,
   6,
   7,
   8,
   9,
   10,
   11,
   12,
   15,
   18,
   20,
   21,
   22,
   23,
   24,
   25,
   26,
   27,
   28,
   29,
   32,
   33,
   34,
   35,
   36,
   37,
   38,
   40),
  'cv_scores': array([0.84176214, 0.84393978, 0.84113795, 0.84219458, 0.844
87021]),
  'avg_score': 0.8427809322153685,
  'feature_names': ('Available Extra Rooms in Hospital',
   'Admission_Deposit',
   'Department_anesthesia',
   'Department_gynecology',
   'Department_radiotherapy',
   'Department_surgery',
   'Ward_Facility_Code_B',
   'Ward_Facility_Code_C',
   'Ward_Facility_Code_D',
   'Ward_Facility_Code_E',
   'Ward_Facility_Code_F',
   'doctor_name_Dr Nathan',
   'doctor_name_Dr Sarah',
   'doctor_name_Dr Sophia',
   'Age_11-20',
   'Age_21-30',
   'Age_31-40',
   'Age_41-50',
   'Age_51-60',
   'Age_61-70',
   'Age_71-80',
   'Age_81-90',
   'Age_91-100',
   'Type of Admission_Trauma',
   'Type of Admission_Urgent',
```

```
  'Severity of Illness_Minor',
  'Severity of Illness_Moderate',
  'health_conditions_Diabetes',
  'health_conditions_Heart disease',
  'health_conditions_High Blood Pressure',
  'Insurance_Yes'),
 'ci_bound': 0.0017984071192350042,
 'std_dev': 0.001399221837098084,
 'std_err': 0.000699610918549042},
32: {'feature_idx': (0,
  3,
  4,
  5,
  6,
  7,
  8,
  9,
  10,
  11,
  12,
  15,
  16,
  18,
  20,
  21,
  22,
  23,
  24,
  25,
  26,
  27,
  28,
  29,
  32,
  33,
  34,
  35,
  36,
  37,
  38,
  40),
 'cv_scores': array([0.84176214, 0.84393978, 0.84113795, 0.84219458, 0.844
87021]),
 'avg_score': 0.8427809322153685,
 'feature_names': ('Available Extra Rooms in Hospital',
  'Admission_Deposit',
  'Department_anesthesia',
  'Department_gynecology',
  'Department_radiotherapy',
  'Department_surgery',
  'Ward_Facility_Code_B',
  'Ward_Facility_Code_C',
  'Ward_Facility_Code_D',
  'Ward_Facility_Code_E',
  'Ward_Facility_Code_F',
  'doctor_name_Dr Nathan',
```

```
  'doctor_name_Dr Olivia',
  'doctor_name_Dr Sarah',
  'doctor_name_Dr Sophia',
  'Age_11-20',
  'Age_21-30',
  'Age_31-40',
  'Age_41-50',
  'Age_51-60',
  'Age_61-70',
  'Age_71-80',
  'Age_81-90',
  'Age_91-100',
  'Type of Admission_Trauma',
  'Type of Admission_Urgent',
  'Severity of Illness_Minor',
  'Severity of Illness_Moderate',
  'health_conditions_Diabetes',
  'health_conditions_Heart disease',
  'health_conditions_High Blood Pressure',
  'Insurance_Yes'),
 'ci_bound': 0.0017984071192350068,
 'std_dev': 0.001399221837098086,
 'std_err': 0.0006996109185490433},
33: {'feature_idx': (0,
  3,
  4,
  5,
  6,
  7,
  8,
  9,
  10,
  11,
  12,
  15,
  16,
  17,
  18,
  20,
  21,
  22,
  23,
  24,
  25,
  26,
  27,
  28,
  29,
  32,
  33,
  34,
  35,
  36,
  37,
  38,
  40),
```

```
  'cv_scores': array([0.84176243, 0.84394197, 0.84113616, 0.84218883, 0.844
8723 ]),
  'avg_score': 0.8427803380446244,
  'feature_names': ('Available Extra Rooms in Hospital',
   'Admission_Deposit',
   'Department_anesthesia',
   'Department_gynecology',
   'Department_radiotherapy',
   'Department_surgery',
   'Ward_Facility_Code_B',
   'Ward_Facility_Code_C',
   'Ward_Facility_Code_D',
   'Ward_Facility_Code_E',
   'Ward_Facility_Code_F',
   'doctor_name_Dr Nathan',
   'doctor_name_Dr Olivia',
   'doctor_name_Dr Sam',
   'doctor_name_Dr Sarah',
   'doctor_name_Dr Sophia',
   'Age_11-20',
   'Age_21-30',
   'Age_31-40',
   'Age_41-50',
   'Age_51-60',
   'Age_61-70',
   'Age_71-80',
   'Age_81-90',
   'Age_91-100',
   'Type of Admission_Trauma',
   'Type of Admission_Urgent',
   'Severity of Illness_Minor',
   'Severity of Illness_Moderate',
   'health_conditions_Diabetes',
   'health_conditions_Heart disease',
   'health_conditions_High Blood Pressure',
   'Insurance_Yes'),
  'ci_bound': 0.0018007868930800973,
  'std_dev': 0.0014010733819990102,
  'std_err': 0.0007005366909995051},
 34: {'feature_idx': (0,
   2,
   3,
   4,
   5,
   6,
   7,
   8,
   9,
   10,
   11,
   12,
   15,
   16,
   17,
   18,
   20,
```

```
   21,
   22,
   23,
   24,
   25,
   26,
   27,
   28,
   29,
   32,
   33,
   34,
   35,
   36,
   37,
   38,
   40),
  'cv_scores': array([0.84176239, 0.84394137, 0.84113397, 0.84218859, 0.844
87215]),
  'avg_score': 0.8427796949690372,
  'feature_names': ('Available Extra Rooms in Hospital',
   'Visitors with Patient',
   'Admission_Deposit',
   'Department_anesthesia',
   'Department_gynecology',
   'Department_radiotherapy',
   'Department_surgery',
   'Ward_Facility_Code_B',
   'Ward_Facility_Code_C',
   'Ward_Facility_Code_D',
   'Ward_Facility_Code_E',
   'Ward_Facility_Code_F',
   'doctor_name_Dr Nathan',
   'doctor_name_Dr Olivia',
   'doctor_name_Dr Sam',
   'doctor_name_Dr Sarah',
   'doctor_name_Dr Sophia',
   'Age_11-20',
   'Age_21-30',
   'Age_31-40',
   'Age_41-50',
   'Age_51-60',
   'Age_61-70',
   'Age_71-80',
   'Age_81-90',
   'Age_91-100',
   'Type of Admission_Trauma',
   'Type of Admission_Urgent',
   'Severity of Illness_Minor',
   'Severity of Illness_Moderate',
   'health_conditions_Diabetes',
   'health_conditions_Heart disease',
   'health_conditions_High Blood Pressure',
   'Insurance_Yes'),
  'ci_bound': 0.0018012958608325652,
  'std_dev': 0.001401469376201882,
```

```
  'std_err': 0.0007007346881009442},
 35: {'feature_idx': (0,
   1,
   2,
   3,
   4,
   5,
   6,
   7,
   8,
   9,
   10,
   11,
   12,
   15,
   16,
   17,
   18,
   20,
   21,
   22,
   23,
   24,
   25,
   26,
   27,
   28,
   29,
   32,
   33,
   34,
   35,
   36,
   37,
   38,
   40),
  'cv_scores': array([0.84176142, 0.84394176, 0.84113311, 0.84218763, 0.844
8709 ]),
  'avg_score': 0.8427789615641776,
  'feature_names': ('Available Extra Rooms in Hospital',
   'staff_available',
   'Visitors with Patient',
   'Admission_Deposit',
   'Department_anesthesia',
   'Department_gynecology',
   'Department_radiotherapy',
   'Department_surgery',
   'Ward_Facility_Code_B',
   'Ward_Facility_Code_C',
   'Ward_Facility_Code_D',
   'Ward_Facility_Code_E',
   'Ward_Facility_Code_F',
   'doctor_name_Dr Nathan',
   'doctor_name_Dr Olivia',
   'doctor_name_Dr Sam',
   'doctor_name_Dr Sarah',
```

```
  'doctor_name_Dr Sophia',
  'Age_11-20',
  'Age_21-30',
  'Age_31-40',
  'Age_41-50',
  'Age_51-60',
  'Age_61-70',
  'Age_71-80',
  'Age_81-90',
  'Age_91-100',
  'Type of Admission_Trauma',
  'Type of Admission_Urgent',
  'Severity of Illness_Minor',
  'Severity of Illness_Moderate',
  'health_conditions_Diabetes',
  'health_conditions_Heart disease',
  'health_conditions_High Blood Pressure',
  'Insurance_Yes'),
 'ci_bound': 0.0018014433618171496,
 'std_dev': 0.0014015841369790323,
 'std_err': 0.0007007920684895 16},
36: {'feature_idx': (0,
  1,
  2,
  3,
  4,
  5,
  6,
  7,
  8,
  9,
  10,
  11,
  12,
  15,
  16,
  17,
  18,
  20,
  21,
  22,
  23,
  24,
  25,
  26,
  27,
  28,
  29,
  32,
  33,
  34,
  35,
  36,
  37,
  38,
  39,
```

```
    40),
   'cv_scores': array([0.84175766, 0.84394198, 0.84113194, 0.84218758, 0.844
87043]),
   'avg_score': 0.8427779197388698,
   'feature_names': ('Available Extra Rooms in Hospital',
    'staff_available',
    'Visitors with Patient',
    'Admission_Deposit',
    'Department_anesthesia',
    'Department_gynecology',
    'Department_radiotherapy',
    'Department_surgery',
    'Ward_Facility_Code_B',
    'Ward_Facility_Code_C',
    'Ward_Facility_Code_D',
    'Ward_Facility_Code_E',
    'Ward_Facility_Code_F',
    'doctor_name_Dr Nathan',
    'doctor_name_Dr Olivia',
    'doctor_name_Dr Sam',
    'doctor_name_Dr Sarah',
    'doctor_name_Dr Sophia',
    'Age_11-20',
    'Age_21-30',
    'Age_31-40',
    'Age_41-50',
    'Age_51-60',
    'Age_61-70',
    'Age_71-80',
    'Age_81-90',
    'Age_91-100',
    'Type of Admission_Trauma',
    'Type of Admission_Urgent',
    'Severity of Illness_Minor',
    'Severity of Illness_Moderate',
    'health_conditions_Diabetes',
    'health_conditions_Heart disease',
    'health_conditions_High Blood Pressure',
    'health_conditions_Other',
    'Insurance_Yes'),
   'ci_bound': 0.00180237000492073,
   'std_dev': 0.001402305096795004,
   'std_err': 0.0007011525483975019},
  37: {'feature_idx': (0,
    1,
    2,
    3,
    4,
    5,
    6,
    7,
    8,
    9,
    10,
    11,
    12,
```

```
    15,
    16,
    17,
    18,
    19,
    20,
    21,
    22,
    23,
    24,
    25,
    26,
    27,
    28,
    29,
    32,
    33,
    34,
    35,
    36,
    37,
    38,
    39,
    40),
  'cv_scores': array([0.8417447 , 0.84394054, 0.8411334 , 0.84218957, 0.844
87178]),
  'avg_score': 0.8427759963463547,
  'feature_names': ('Available Extra Rooms in Hospital',
   'staff_available',
   'Visitors with Patient',
   'Admission_Deposit',
   'Department_anesthesia',
   'Department_gynecology',
   'Department_radiotherapy',
   'Department_surgery',
   'Ward_Facility_Code_B',
   'Ward_Facility_Code_C',
   'Ward_Facility_Code_D',
   'Ward_Facility_Code_E',
   'Ward_Facility_Code_F',
   'doctor_name_Dr Nathan',
   'doctor_name_Dr Olivia',
   'doctor_name_Dr Sam',
   'doctor_name_Dr Sarah',
   'doctor_name_Dr Simon',
   'doctor_name_Dr Sophia',
   'Age_11-20',
   'Age_21-30',
   'Age_31-40',
   'Age_41-50',
   'Age_51-60',
   'Age_61-70',
   'Age_71-80',
   'Age_81-90',
   'Age_91-100',
   'Type of Admission_Trauma',
```

```
  'Type of Admission_Urgent',
  'Severity of Illness_Minor',
  'Severity of Illness_Moderate',
  'health_conditions_Diabetes',
  'health_conditions_Heart disease',
  'health_conditions_High Blood Pressure',
  'health_conditions_Other',
  'Insurance_Yes'),
 'ci_bound': 0.0018043646007731885,
 'std_dev': 0.001403856960131784,
 'std_err': 0.000701928480065892},
38: {'feature_idx': (0,
  1,
  2,
  3,
  4,
  5,
  6,
  7,
  8,
  9,
  10,
  11,
  12,
  15,
  16,
  17,
  18,
  19,
  20,
  21,
  22,
  23,
  24,
  25,
  26,
  27,
  28,
  29,
  31,
  32,
  33,
  34,
  35,
  36,
  37,
  38,
  39,
  40),
 'cv_scores': array([0.8417445 , 0.84393137, 0.84113339, 0.84218943, 0.844
86828]),
 'avg_score': 0.8427733960444744,
 'feature_names': ('Available Extra Rooms in Hospital',
  'staff_available',
  'Visitors with Patient',
  'Admission_Deposit',
```

```
   'Department_anesthesia',
   'Department_gynecology',
   'Department_radiotherapy',
   'Department_surgery',
   'Ward_Facility_Code_B',
   'Ward_Facility_Code_C',
   'Ward_Facility_Code_D',
   'Ward_Facility_Code_E',
   'Ward_Facility_Code_F',
   'doctor_name_Dr Nathan',
   'doctor_name_Dr Olivia',
   'doctor_name_Dr Sam',
   'doctor_name_Dr Sarah',
   'doctor_name_Dr Simon',
   'doctor_name_Dr Sophia',
   'Age_11-20',
   'Age_21-30',
   'Age_31-40',
   'Age_41-50',
   'Age_51-60',
   'Age_61-70',
   'Age_71-80',
   'Age_81-90',
   'Age_91-100',
   'gender_Other',
   'Type of Admission_Trauma',
   'Type of Admission_Urgent',
   'Severity of Illness_Minor',
   'Severity of Illness_Moderate',
   'health_conditions_Diabetes',
   'health_conditions_Heart disease',
   'health_conditions_High Blood Pressure',
   'health_conditions_Other',
   'Insurance_Yes'),
  'ci_bound': 0.0018011225521462203,
  'std_dev': 0.001401334536155995,
  'std_err': 0.0007006672680779975},
 39: {'feature_idx': (0,
   1,
   2,
   3,
   4,
   5,
   6,
   7,
   8,
   9,
   10,
   11,
   12,
   13,
   15,
   16,
   17,
   18,
   19,
```

```
   20,
   21,
   22,
   23,
   24,
   25,
   26,
   27,
   28,
   29,
   31,
   32,
   33,
   34,
   35,
   36,
   37,
   38,
   39,
   40),
  'cv_scores': array([0.84174483, 0.84393285, 0.84112937, 0.84218622, 0.844
85046]),
  'avg_score': 0.8427687463538682,
  'feature_names': ('Available Extra Rooms in Hospital',
   'staff_available',
   'Visitors with Patient',
   'Admission_Deposit',
   'Department_anesthesia',
   'Department_gynecology',
   'Department_radiotherapy',
   'Department_surgery',
   'Ward_Facility_Code_B',
   'Ward_Facility_Code_C',
   'Ward_Facility_Code_D',
   'Ward_Facility_Code_E',
   'Ward_Facility_Code_F',
   'doctor_name_Dr John',
   'doctor_name_Dr Nathan',
   'doctor_name_Dr Olivia',
   'doctor_name_Dr Sam',
   'doctor_name_Dr Sarah',
   'doctor_name_Dr Simon',
   'doctor_name_Dr Sophia',
   'Age_11-20',
   'Age_21-30',
   'Age_31-40',
   'Age_41-50',
   'Age_51-60',
   'Age_61-70',
   'Age_71-80',
   'Age_81-90',
   'Age_91-100',
   'gender_Other',
   'Type of Admission_Trauma',
   'Type of Admission_Urgent',
   'Severity of Illness_Minor',
```

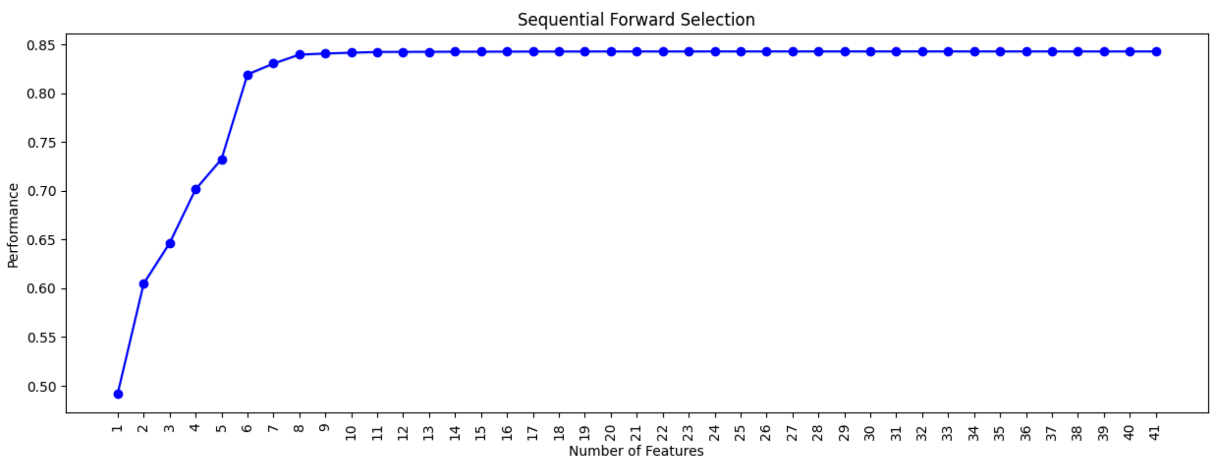
```
   'Severity of Illness_Moderate',
   'health_conditions_Diabetes',
   'health_conditions_Heart disease',
   'health_conditions_High Blood Pressure',
   'health_conditions_Other',
   'Insurance_Yes'),
  'ci_bound': 0.0017960998670823037,
  'std_dev': 0.0013974267165376608,
  'std_err': 0.0006987133582688304},
 40: {'feature_idx': (0,
   1,
   2,
   3,
   4,
   5,
   6,
   7,
   8,
   9,
   10,
   11,
   12,
   13,
   14,
   15,
   16,
   17,
   18,
   19,
   20,
   21,
   22,
   23,
   24,
   25,
   26,
   27,
   28,
   29,
   31,
   32,
   33,
   34,
   35,
   36,
   37,
   38,
   39,
   40),
  'cv_scores': array([0.84174483, 0.84393285, 0.84112937, 0.84218622, 0.844
85046]),
  'avg_score': 0.8427687463538682,
  'feature_names': ('Available Extra Rooms in Hospital',
   'staff_available',
   'Visitors with Patient',
   'Admission_Deposit',
```

'Department_anesthesia',
'Department_gynecology',
'Department_radiotherapy',
'Department_surgery',
'Ward_Facility_Code_B',
'Ward_Facility_Code_C',
'Ward_Facility_Code_D',
'Ward_Facility_Code_E',
'Ward_Facility_Code_F',
'doctor_name_Dr John',
'doctor_name_Dr Mark',
'doctor_name_Dr Nathan',
'doctor_name_Dr Olivia',
'doctor_name_Dr Sam',
'doctor_name_Dr Sarah',
'doctor_name_Dr Simon',
'doctor_name_Dr Sophia',
'Age_11-20',
'Age_21-30',
'Age_31-40',
'Age_41-50',
'Age_51-60',
'Age_61-70',
'Age_71-80',
'Age_81-90',
'Age_91-100',
'gender_Other',
'Type of Admission_Trauma',
'Type of Admission_Urgent',
'Severity of Illness_Minor',
'Severity of Illness_Moderate',
'health_conditions_Diabetes',
'health_conditions_Heart disease',
'health_conditions_High Blood Pressure',
'health_conditions_Other',
'Insurance_Yes'),
'ci_bound': 0.001796099867082185,
'std_dev': 0.0013974267165375687,
'std_err': 0.0006987133582687842},
41: {'feature_idx': (0,
1,
2,
3,
4,
5,
6,
7,
8,
9,
10,
11,
12,
13,
14,
15,
16,

```
   17,
   18,
   19,
   20,
   21,
   22,
   23,
   24,
   25,
   26,
   27,
   28,
   29,
   30,
   31,
   32,
   33,
   34,
   35,
   36,
   37,
   38,
   39,
   40),
  'cv_scores': array([0.84174483, 0.84393285, 0.84112937, 0.84218622, 0.844
85046]),
  'avg_score': 0.8427687463538682,
  'feature_names': ('Available Extra Rooms in Hospital',
   'staff_available',
   'Visitors with Patient',
   'Admission_Deposit',
   'Department_anesthesia',
   'Department_gynecology',
   'Department_radiotherapy',
   'Department_surgery',
   'Ward_Facility_Code_B',
   'Ward_Facility_Code_C',
   'Ward_Facility_Code_D',
   'Ward_Facility_Code_E',
   'Ward_Facility_Code_F',
   'doctor_name_Dr John',
   'doctor_name_Dr Mark',
   'doctor_name_Dr Nathan',
   'doctor_name_Dr Olivia',
   'doctor_name_Dr Sam',
   'doctor_name_Dr Sarah',
   'doctor_name_Dr Simon',
   'doctor_name_Dr Sophia',
   'Age_11-20',
   'Age_21-30',
   'Age_31-40',
   'Age_41-50',
   'Age_51-60',
   'Age_61-70',
   'Age_71-80',
   'Age_81-90',
```

```
  'Age_91-100',
  'gender_Male',
  'gender_Other',
  'Type of Admission_Trauma',
  'Type of Admission_Urgent',
  'Severity of Illness_Minor',
  'Severity of Illness_Moderate',
  'health_conditions_Diabetes',
  'health_conditions_Heart disease',
  'health_conditions_High Blood Pressure',
  'health_conditions_Other',
  'Insurance_Yes'),
 'ci_bound': 0.0017960998670816415,
 'std_dev': 0.0013974267165371458,
 'std_err': 0.0006987133582685728}}
```

In [59]:
```python
# To plot the performance of the model with addition of each feature
from mlxtend.plotting import plot_sequential_feature_selection as plot_sfs

fig1 = plot_sfs(sfs.get_metric_dict(), kind="std_err", figsize=(15, 5))
plt.title("Sequential Forward Selection")
plt.xticks(rotation=90)
plt.show()
```



Observations:

- We can observe that the performance increases till the 8th feature and then becomes constant.
- The decision to choose the *k_features* now depends on the R^2 vs the complexity of the model.
  - With 8 features, we are getting an R^2 of 0.840.
  - With 20 features, we are getting an R^2 of 0.844.
  - With 42 features, we are getting an R^2 of 0.843.
- The increase in R^2 is not very significant as we are getting approximately the same values with a less complex model.
- So we'll use 8 features only to build the Linear Regression model, but you can experiment by taking a different number.

- Number of features chosen can also depend on the business context and use case of the model.

Let's run the Sequential Feature Selector again to find the best 8 features for the model.

In [60]:
```
reg = LinearRegression()

# Forward feature selection with 8 features
sfs = SFS(
    reg,
    k_features=8,
    forward=True,
    floating=False,
    scoring="r2",
    n_jobs=-1,
    verbose=2,
    cv=5,
)

# Perform SFFS
sfs = sfs.fit(x_train, y_train)
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done  13 tasks      | elapsed:    3.5s
[Parallel(n_jobs=-1)]: Done  35 out of  41 | elapsed:    8.3s remaining:
1.4s
[Parallel(n_jobs=-1)]: Done  41 out of  41 | elapsed:    9.7s finished

[2025-01-24 22:27:13] Features: 1/8 -- score: 0.4918898861031266[Parallel(n_
jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done  13 tasks      | elapsed:    3.6s
[Parallel(n_jobs=-1)]: Done  34 out of  40 | elapsed:    8.5s remaining:
1.5s
[Parallel(n_jobs=-1)]: Done  40 out of  40 | elapsed:    9.8s finished

[2025-01-24 22:27:23] Features: 2/8 -- score: 0.6046160397618205[Parallel(n_
jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done  32 out of  39 | elapsed:    8.3s remaining:
1.8s
[Parallel(n_jobs=-1)]: Done  39 out of  39 | elapsed:    9.8s finished

[2025-01-24 22:27:33] Features: 3/8 -- score: 0.646190914266793[Parallel(n_j
obs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done  31 out of  38 | elapsed:    8.1s remaining:
1.8s
[Parallel(n_jobs=-1)]: Done  38 out of  38 | elapsed:    9.7s finished

[2025-01-24 22:27:43] Features: 4/8 -- score: 0.7013054914237962[Parallel(n_
jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done  29 out of  37 | elapsed:    8.0s remaining:
2.2s
[Parallel(n_jobs=-1)]: Done  37 out of  37 | elapsed:    9.6s finished

[2025-01-24 22:27:53] Features: 5/8 -- score: 0.7323069421611099[Parallel(n_
jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 out of  36 | elapsed:    7.7s remaining:
2.2s
[Parallel(n_jobs=-1)]: Done  36 out of  36 | elapsed:    9.5s finished

[2025-01-24 22:28:02] Features: 6/8 -- score: 0.8191351388509375[Parallel(n_
jobs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done  26 out of  35 | elapsed:    7.2s remaining:
2.5s
[Parallel(n_jobs=-1)]: Done  35 out of  35 | elapsed:    9.3s finished

[2025-01-24 22:28:12] Features: 7/8 -- score: 0.830328286206495[Parallel(n_j
obs=-1)]: Using backend LokyBackend with 14 concurrent workers.
[Parallel(n_jobs=-1)]: Done  25 out of  34 | elapsed:    7.1s remaining:
2.5s
[Parallel(n_jobs=-1)]: Done  34 out of  34 | elapsed:    9.3s finished

[2025-01-24 22:28:21] Features: 8/8 -- score: 0.8395075505200218
```

In [61]:
```
# Selecting the features which are important for the model
feat_cols = list(sfs.k_feature_idx_)
print(feat_cols)
```

```
[4, 5, 6, 7, 21, 22, 23, 24]
```

In [62]:
```
# Checking the names of the important features
x_train.columns[feat_cols]
```

Out[62]:
```
Index(['Department_anesthesia', 'Department_gynecology',
       'Department_radiotherapy', 'Department_surgery', 'Age_11-20',
       'Age_21-30', 'Age_31-40', 'Age_41-50'],
      dtype='object')
```

Now, we will fit the Linear Regression model using these 8 features only.

In [63]:
```
# Creating the new x_train data
x_train_final = x_train[x_train.columns[feat_cols]]
```

In [64]:
```
# Creating the new x_test data
x_test_final = x_test[x_train_final.columns]
```

In [65]:
```
# Fitting Linear Regression model on the new training data
lin_reg_model2 = LinearRegression()
lin_reg_model2.fit(x_train_final, y_train)
```

Out[65]:
```
▾ LinearRegression
LinearRegression()
```

In [66]:
```
# Checking model performance on the training data
lin_reg_model2_train_perf = model_performance_regression(lin_reg_model2, x_t
lin_reg_model2_train_perf
```

Out[66]:

	RMSE	MAE	R-squared	Adj. R-squared	MAPE
0	3.167762	2.16747	0.83952	0.839516	19.769004

In [67]:
```
# Checking model performance on the testing data
lin_reg_model2_test_perf = model_performance_regression(lin_reg_model2, x_te
lin_reg_model2_test_perf
```

Out[67]:

	RMSE	MAE	R-squared	Adj. R-squared	MAPE
0	3.175516	2.174951	0.839871	0.839858	19.83425

Observations:

- The performance looks approximately the same as the previous model with all the variables.
- Let's compare the two models we built.

In [68]:
```
# Training performance comparison

models_train_comp_df = pd.concat(
    [linear_reg.T, lin_reg_model2_train_perf.T], axis=1,
```

```
)

models_train_comp_df.columns = [
    "Linear Regression sklearn",
    "Linear Regression sklearn (SFS features)",
]

print("Training performance comparison:")
models_train_comp_df
```

```
Training performance comparison:
```

Out[68]:

	Linear Regression sklearn	Linear Regression sklearn (SFS features)
RMSE	3.135093	3.167762
MAE	2.146244	2.167470
R-squared	0.842813	0.839520
Adj. R-squared	0.842796	0.839516
MAPE	19.591833	19.769004

In [69]:

```
# Testing performance comparison

models_test_comp_df = pd.concat(
    [linear_reg_test.T, lin_reg_model2_test_perf.T], axis=1,
)

models_test_comp_df.columns = [
    "Linear Regression sklearn",
    "Linear Regression sklearn (SFS features)",
]

print("Test performance comparison:")
models_test_comp_df
```

```
Test performance comparison:
```

Out[69]:

	Linear Regression sklearn	Linear Regression sklearn (SFS features)
RMSE	3.144055	3.175516
MAE	2.155765	2.174951
R-squared	0.843028	0.839871
Adj. R-squared	0.842964	0.839858
MAPE	19.676966	19.834250

- The new model (**lin_reg_model2**) uses 8 features in comparison to 42 features for the previous model (**linear_reg**), i.e., the number of features has reduced by ~81%.
- The performance of the new model, however, is very close to our previous model.
- Depending upon time sensitivity and storage restrictions, we can choose between the models.

Next Steps

- We have explored building a Linear Regression model for this problem statement of predicting the likely length of stay of a patient for a hospital visit, and we've also identifies the most important features for the model, and trained the model using only those features, without compromising the model performance by much.
- However, being a linear model, it is more interpretable than a model with high predictive power. The performance metrics of our attempt at prediction can be improved with more complex and non-linear models.
- In the coming section, we will explore building models on more complex regularized versions of Linear Regression, and also get into non-linear tree-based regression models, to see if we can improve on the model's predictive performance.