

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_conditions:** 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 [1]: 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, KFold
from sklearn.linear_model import LinearRegression, Ridge, Lasso
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 [2]: # Read the healthcare dataset file
data = pd.read_csv("healthcare_data.csv")
```

```
In [3]: # Copying data to another variable to avoid any changes to original data
same_data = data.copy()
```

Data Overview

```
In [4]: # View the first 5 rows of the dataset
data.head()
```

```
Out[4]:
```

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

Out [5]:

	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 [6]: # Understand the shape of the data
data.shape
```

Out[6]: (500000, 15)

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

```
In [7]: # 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 [8]: # To view patientid and the number of times they have been admitted to the h
data['patientid'].value_counts()
```

```
Out[8]: 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 [9]: # Dropping patientid from the data as it is an identifier and will not add v
data=data.drop(columns=["patientid"])
```

```
In [10]: # Checking for duplicate values in the data
data.duplicated().sum()
```

```
Out[10]: 0
```

Observation:

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

```
In [11]: # Checking the descriptive statistics of the columns
data.describe().T
```

Out [11]:

	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 [12]: # 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 categorical variable
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

The `histogram_boxplot` function is a Python function that combines a boxplot and a histogram into one figure, creating a visual summary of the distribution of a single feature in a dataset. This helps analyze the data by showing key statistics (like mean and median) and the overall distribution. Heres a step-by-step breakdown of the function:

1. Function Parameters

data:The input DataFrame containing the data you want to analyze. feature:The column name from the DataFrame (data) whose distribution you want to plot. figsize:A tuple defining the size of the figure (default is (12, 7)). kde:A boolean flag (True or False) to indicate whether a Kernel Density Estimate (KDE) curve should be overlaid on the histogram. bins:The number of bins to use for the histogram. If None, the default binning is used.

2. Creating Subplots

Subplots: Two subplots are created: ax_box2: For the boxplot (top). ax_hist2: For the histogram (bottom). gridspec_kw: Adjusts the relative sizes of the subplots: 0.25 for the boxplot (smaller). 0.75 for the histogram (larger).

```
f2, (ax_box2, ax_hist2) = plt.subplots( nrows=2, # Number of rows = 2 (boxplot and histogram stacked vertically) sharex=True, # Shared x-axis for both plots gridspec_kw={"height_ratios": (0.25, 0.75)}, # Ratio of heights (boxplot is smaller) figsize=figsize, )
```

3. Boxplot Creation

Purpose: Creates a boxplot for the feature. showmeans=True: Adds a star (*) to indicate the mean value. color="violet": Sets the color of the boxplot. sns.boxplot(data=data, x=feature, ax=ax_box2, showmeans=True, color="violet")

4. Histogram Creation

sns.histplot: Creates a histogram of the data. kde: If True, a KDE curve is overlaid. bins: Specifies the number of bins (if provided). palette="winter": Sets the color palette for the histogram. Conditional logic: If bins is provided, the specified bin size is used. If bins is None, the default bin size is used.

```
sns.histplot( data=data, x=feature, kde=kde, ax=ax_hist2, bins=bins, palette="winter" ) if bins else sns.histplot( data=data, x=feature, kde=kde, ax=ax_hist2 )
```

5. Adding Mean and Median Lines

```
ax_hist2.axvline(data[feature].mean(), color="green", linestyle="--")
ax_hist2.axvline(data[feature].median(), color="black", linestyle="-")
```

6. Visualization Flow

Boxplot: Displays summary statistics such as the median, quartiles, and outliers. Positioned above the histogram to show the distribution at a glance. Histogram: Shows the frequency distribution of values. Overlay of mean and median lines provides additional insights into the distribution.

```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
```

Sample data

```
data = sns.load_dataset('tips')
```

Call the function

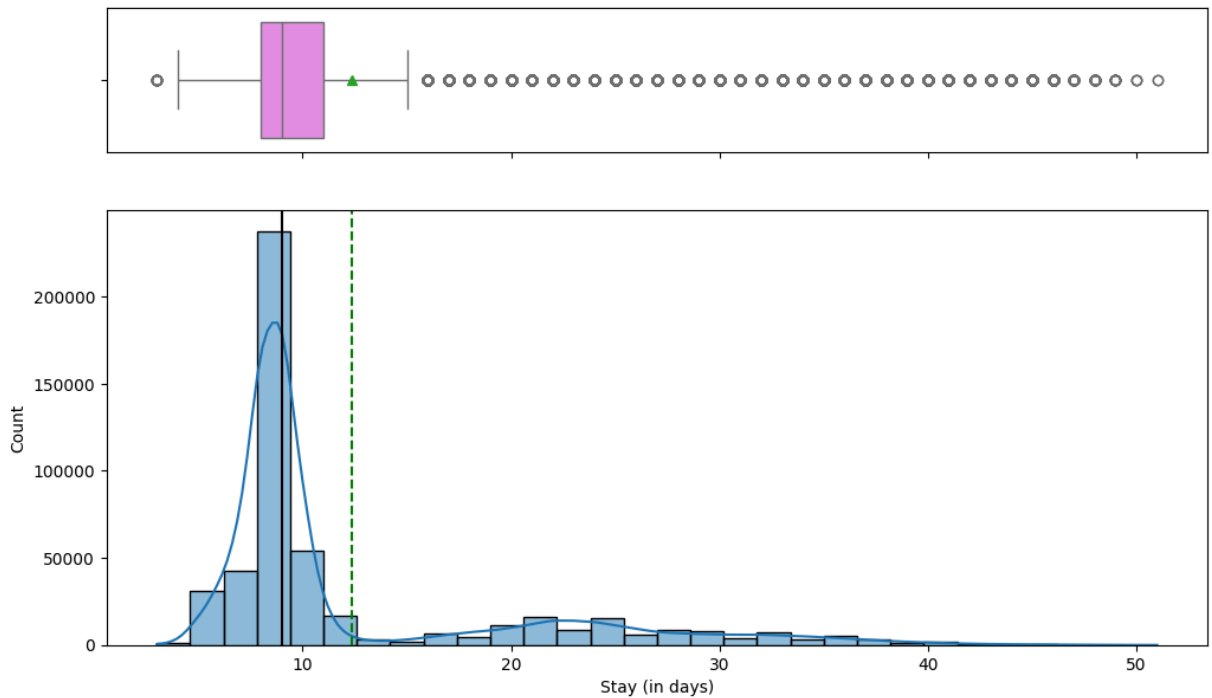
```
histogram_boxplot(data, feature='total_bill', kde=True, bins=30) plt.show()
```

```
In [13]: # 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, color = "#1f77b4")
    # Boxplot will be created and a star will indicate the mean
    sns.histplot(
        data = data, x = feature, kde = kde, ax = ax_hist2, bins = bins, palette = "magma"
    )
    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 [14]: histogram_boxplot(data, "Stay (in days)", kde = True, bins = 30)
```

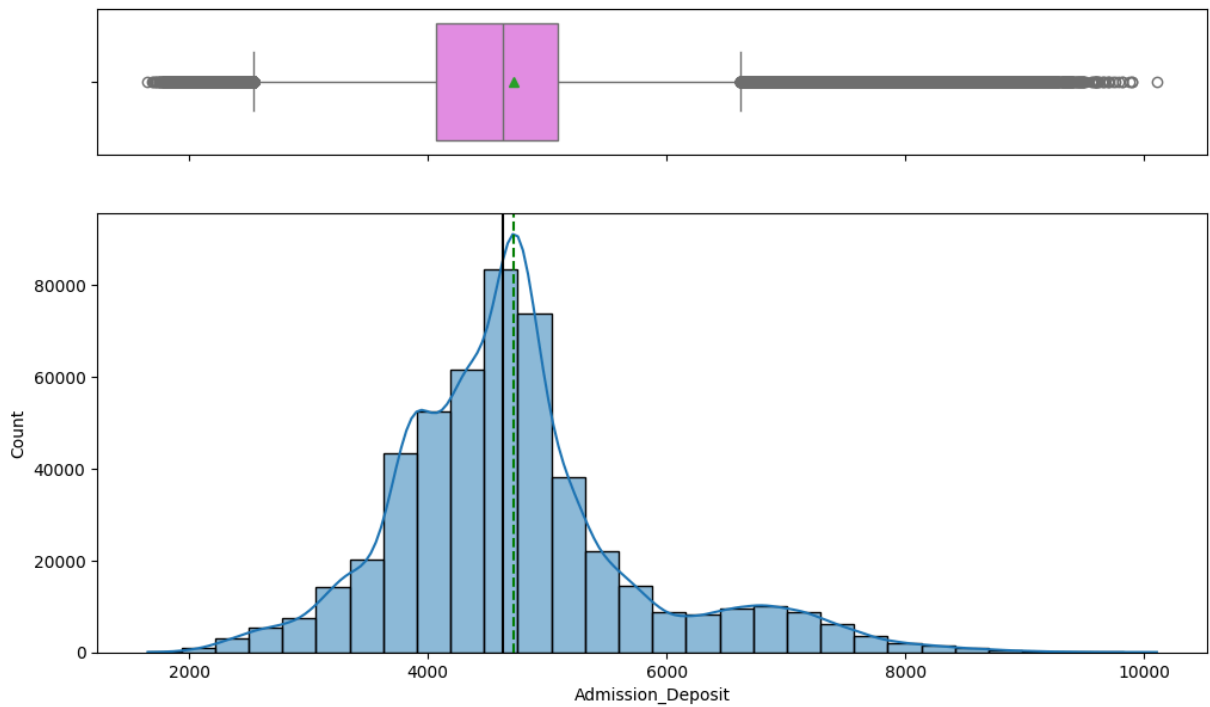


Observations:

- **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 [15]: histogram_boxplot(data, "Admission_Deposit", kde = True, bins = 30)
```

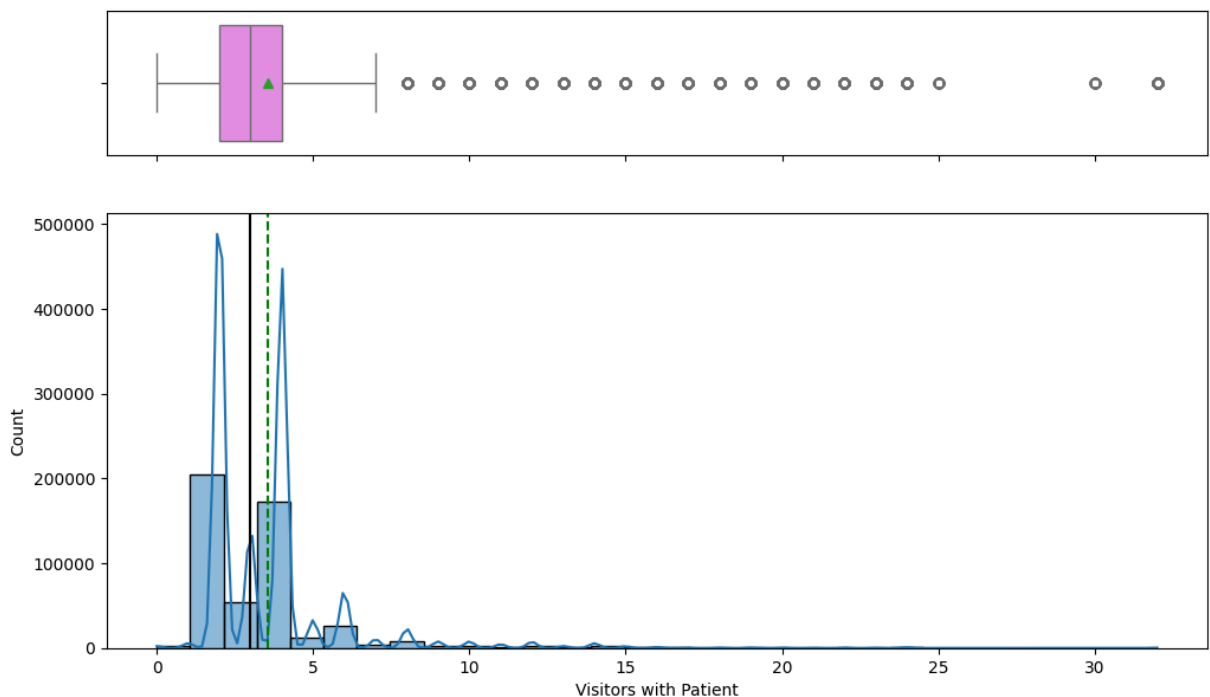


Observation:

- 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 [16]: `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 [17]: data.corr(numeric_only=True)
```

Out[17]:

	Available Extra Rooms in Hospital	staff_available	Visitors with Patient	Admission_Deposit	Stay (i days)
Available Extra Rooms in Hospital	1.000000	-0.001784	0.070459	-0.050127	-0.01921
staff_available	-0.001784	1.000000	0.000578	0.000763	0.00739
Visitors with Patient	0.070459	0.000578	1.000000	-0.069043	0.02730
Admission_Deposit	-0.050127	0.000763	-0.069043	1.000000	0.04420
Stay (in days)	-0.019219	0.007398	0.027302	0.044203	1.00000

```
In [18]: # 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[18]: <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.**

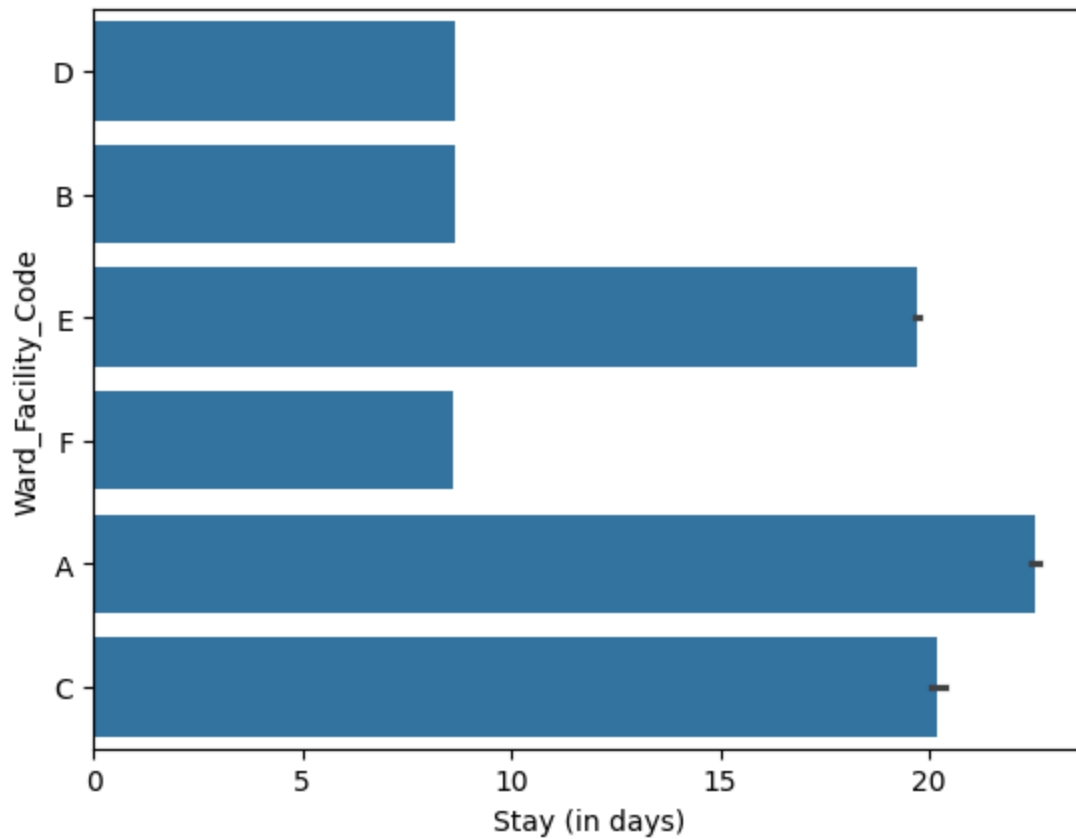
```
In [19]: # Function to plot stacked bar plots

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



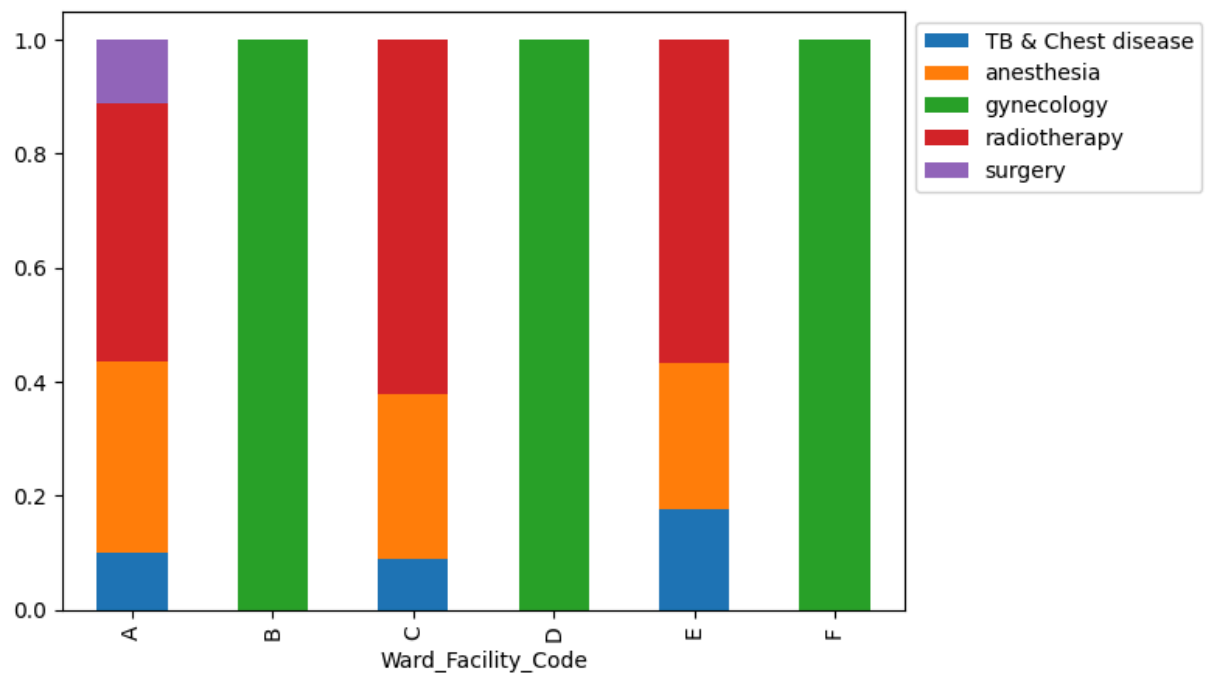
Observation:

- 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 [21]: stacked_barplot(data, "Ward_Facility_Code", "Department")
```

Department \ Ward_Facility_Code	TB & Chest disease	anesthesia	gynecology	radiotherapy
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 Ward_Facility_Code	surgery	All
A	5138	46551
All	5138	500000
B	0	103885
C	0	14597
D	0	119055
E	0	95374
F	0	120538



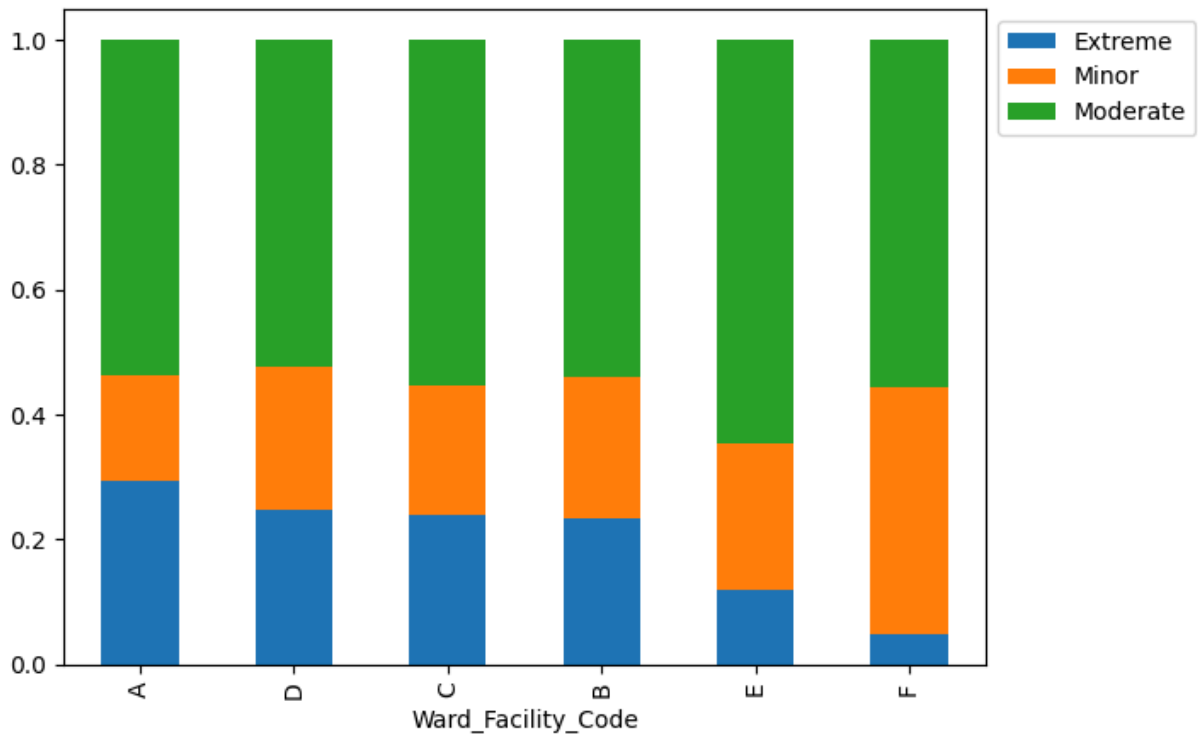
Observations:

- 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 [22]: 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

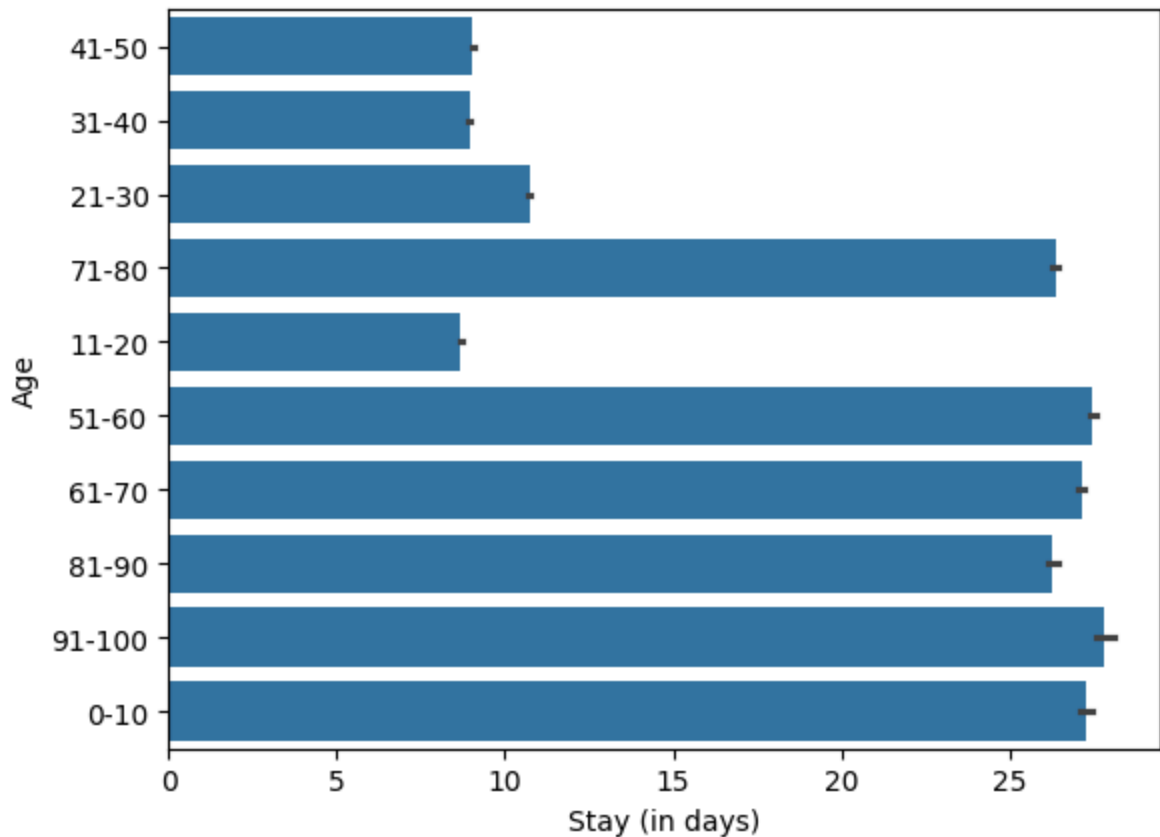


Observations:

- **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 [23]: 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 51-100 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 [24]: data.groupby(['doctor_name'])['Department'].agg(Department_Name='unique', Pat
```

Out [24]:

	Department_Name	Patients_Treated
doctor_name		
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 [25]: # 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,
    drop_first = True,
)
```

```
In [26]: # Check the data after handling categorical data
data
```

Out [26]:

	Available Extra Rooms in Hospital	staff_available	Visitors with Patient	Admission_Deposit	Stay (in days)	Department_a
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 x 42 columns

```
In [27]: # Separating independent variables and the target variable
x = data.drop('Stay (in days)',axis=1)

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

```
In [28]: # 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 [29]: # 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, R^2 , Adjusted R^2 , 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 [30]: # 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 performance
    model: regressor
    predictors: independent variables
    target: dependent variable
    """

    pred = model.predict(predictors)
    r2 = r2_score(target, pred)
    adjr2 = adj_r2_score(predictors, target, pred)
    rmse = np.sqrt(mean_squared_error(target, pred))
    mae = mean_absolute_error(target, pred)
    mape = mape_score(target, pred)

    # Predict using the ir
    # To compute R-squared
    # To compute adjusted
    # To compute RMSE
    # To compute MAE
    # 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
```

```
In [31]: x_train1 = x_train.astype(float)
y_train1 = y_train.astype(float)

import statsmodels.api as sm

# Statsmodel API does not add a constant by default. We need to add it explicitly
x_train1 = sm.add_constant(x_train1)

# Add constant to test data
```

```
x_test1 = sm.add_constant(x_test)

# Train the model
olsmodel1 = sm.OLS(y_train1, x_train1).fit()

# Get the model summary
olsmodel1.summary()
print(olsmodel1.summary())
```

OLS Regression Results

```

=====
==
Dep. Variable:          Stay (in days)    R-squared:                0.8
43
Model:                  OLS              Adj. R-squared:           0.8
43
Method:                 Least Squares     F-statistic:             5.796e+
04
Date:                   Fri, 17 Jan 2025   Prob (F-statistic):       0.
00
Time:                   02:47:30          Log-Likelihood:          -1.0246e+
06
No. Observations:      400000            AIC:                     2.049e+
06
Df Residuals:          399962            BIC:                     2.050e+
06
Df Model:               37
Covariance Type:       nonrobust
=====

```

			coef	std err	t
P> t	[0.025	0.975]			
const			19.8975	0.054	365.723
0.000	19.791	20.004			
Available Extra Rooms in Hospital			0.0786	0.002	42.310
0.000	0.075	0.082			
staff_available			-0.0009	0.002	-0.588
0.556	-0.004	0.002			
Visitors with Patient			0.0002	0.002	0.096
0.923	-0.004	0.005			
Admission_Deposit			-3.839e-05	4.78e-06	-8.028
0.000	-4.78e-05	-2.9e-05			
Department_anesthesia			6.0822	0.029	210.111
0.000	6.025	6.139			
Department_gynecology			0.4649	0.019	24.427
0.000	0.428	0.502			
Department_radiotherapy			-4.6216	0.037	-126.134
0.000	-4.693	-4.550			
Department_surgery			9.6876	0.044	218.616
0.000	9.601	9.774			
Ward_Facility_Code_B			0.2576	0.011	23.619
0.000	0.236	0.279			
Ward_Facility_Code_C			0.4349	0.035	12.604
0.000	0.367	0.502			
Ward_Facility_Code_D			0.2039	0.010	19.458
0.000	0.183	0.224			
Ward_Facility_Code_E			0.3084	0.021	14.399
0.000	0.266	0.350			
Ward_Facility_Code_F			0.0034	0.011	0.307
0.759	-0.018	0.025			
doctor_name_Dr John			7.2650	0.031	234.360
0.000	7.204	7.326			
doctor_name_Dr Mark			1.1957	0.033	36.451

0.000	1.131	1.260			
doctor_name_Dr Nathan			-0.2119	0.015	-13.959
0.000	-0.242	-0.182			
doctor_name_Dr Olivia			-0.3586	0.015	-24.577
0.000	-0.387	-0.330			
doctor_name_Dr Sam			1.2844	0.038	33.912
0.000	1.210	1.359			
doctor_name_Dr Sarah			0.8186	0.016	50.613
0.000	0.787	0.850			
doctor_name_Dr Simon			6.1611	0.069	89.090
0.000	6.026	6.297			
doctor_name_Dr Sophia			0.2168	0.020	10.606
0.000	0.177	0.257			
Age_11-20			-12.7378	0.065	-194.584
0.000	-12.866	-12.609			
Age_21-30			-10.4830	0.062	-168.463
0.000	-10.605	-10.361			
Age_31-40			-13.4355	0.062	-216.956
0.000	-13.557	-13.314			
Age_41-50			-13.5113	0.062	-216.833
0.000	-13.633	-13.389			
Age_51-60			-0.3665	0.065	-5.650
0.000	-0.494	-0.239			
Age_61-70			-0.4217	0.064	-6.591
0.000	-0.547	-0.296			
Age_71-80			-0.2193	0.066	-3.342
0.001	-0.348	-0.091			
Age_81-90			-0.0174	0.072	-0.243
0.808	-0.158	0.123			
Age_91-100			-0.2599	0.112	-2.322
0.020	-0.479	-0.041			
gender_Male			6.0065	0.038	157.327
0.000	5.932	6.081			
gender_Other			-0.0257	0.030	-0.845
0.398	-0.085	0.034			
Type of Admission_Trauma			-0.0667	0.012	-5.626
0.000	-0.090	-0.043			
Type of Admission_Urgent			0.1502	0.018	8.300
0.000	0.115	0.186			
Severity of Illness_Minor			0.0432	0.016	2.659
0.008	0.011	0.075			
Severity of Illness_Moderate			0.0987	0.016	6.253
0.000	0.068	0.130			
health_conditions_Diabetes			-0.3512	0.023	-15.038
0.000	-0.397	-0.305			
health_conditions_Heart disease			0.5943	0.027	21.861
0.000	0.541	0.648			
health_conditions_High Blood Pressure			-0.0367	0.017	-2.164
0.030	-0.070	-0.003			
health_conditions_Other			0.0065	0.014	0.460
0.646	-0.021	0.034			
Insurance_Yes			-0.0107	0.012	-0.880
0.379	-0.034	0.013			

==

Omnibus:

131769.607

Durbin-Watson:

1.9


```

98
Prob(Omnibus):          0.000   Jarque-Bera (JB):          708956.8
19
Skew:                   1.495   Prob(JB):              0.
00
Kurtosis:               8.796   Cond. No.              1.03e+
16
=====
==

```

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[2] The smallest eigenvalue is 8.91e-20. This might indicate that there are strong multicollinearity problems or that the design matrix is singular.

```
In [32]: lin_reg_test = model_performance_regression(olsmodel1, x_test1, y_test)
lin_reg_test
```

```
Out[32]:
```

	RMSE	MAE	R-squared	Adj. R-squared	MAPE
0	3.144055	2.155765	0.843028	0.842962	19.676966

Observations:

- We can observe that **R-squared** for the model is **~0.84**.
- Not all the variables are statistically significant enough to predict the outcome variable. To check which ones are statistically significant or have enough predictive power to predict the target variable, we check the **p-value** for all the independent variables. Independent variables with a p-value of higher than 0.05 are not significant for the 95% confidence level.

Interpreting the Regression Results:

1. **Adjusted R-squared:** It reflects the fit of the model.
 - R-squared values range from 0 to 1, where a higher value generally indicates a better fit, assuming certain conditions are met.
 - In our case, the value for Adj. R-squared is **0.84**
 2. **coef:** It represents the change in the output Y due to a change of one unit in the variable (everything else held constant).
 3. **std err:** It reflects the level of accuracy of the coefficients.
 - The lower it is, the more accurate the coefficients are.
 4. **P > |t|:** The p-value:
 - **Pr(>|t|):** For each independent feature there is a null hypothesis and an alternate hypothesis
- Ho:** Null Hypothesis - The independent feature is not significant

Ha: Alternate Hypothesis - The independent feature is significant

- A p-value of less than 0.05 is considered to be statistically significant.

5. **Confidence Interval:** It represents the range in which our coefficients are likely to fall (with a likelihood of 95%).

- Both the **R-squared and Adjusted R-squared of the model are around 84%**. This is a clear indication that we have been able to create a good model that can explain variance in the LOS of patients for up to 84%.
- We can examine the significance of the regression model, and try dropping insignificant variables.

```
In [33]: print("Performance on train data: ")
         model_performance_regression(olsmodel1, x_train1, y_train)
```

Performance on train data:

```
Out[33]:
```

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

```
In [34]: print("Performance on test data: ")
         model_performance_regression(olsmodel1, x_test1, y_test)
```

Performance on test data:

```
Out[34]:
```

	RMSE	MAE	R-squared	Adj. R-squared	MAPE
0	3.144055	2.155765	0.843028	0.842962	19.676966

Observations:

- **The Root Mean Squared Error** of train and test data are **very close**, indicating that **our model is not overfitting** to the training data.
- 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 - 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%.

Checking for Multicollinearity

Multicollinearity occurs when **independent variables** in a regression model are **highly correlated** to each other, such that they do not provide **unique** or independent

information. A regression coefficient is interpreted as the mean change in the target for each unit change in a feature when all other characteristics are held constant. Changes in one aspect can affect other features when they are correlated. The stronger the relationship, the more difficult it is to change one element without affecting the others. Because the features tend to change concurrently, it becomes difficult for the model to evaluate the link between each variable and the target individually.

Variation Inflation Factor (VIF) is one of the most common ways of detecting multicollinearity in data. The Variation Inflation Factor (VIF) is one of the most common methods for detecting multicollinearity in data. The VIF calculates how much the variance of a regression coefficient is inflated due to model multicollinearity.

The VIF can be calculated in two steps,

- Choose and execute a regression analysis on an independent variable for which you are attempting to calculate VIF. For example, suppose there are three independent variables X1, X2, and X3, and a target variable Y. If we wish to calculate VIF for the variable X1, we use X1 as the target variable and X2 and X3 as independent variables ($X1=b0+b1.X2+b2.X3$).
- The regression model mentioned above gives us R2 squared. The formula below is used to calculate the VIF using R2.

$$VIF_i = \frac{1}{1 - R_i^2}$$

Variance inflation factor (VIFs) tells "**what percentage of the variance is inflated for each coefficient**". For example, a VIF of 1.7 tells you that the variance of a particular coefficient is 70% bigger than what you would expect if there was no multicollinearity, i.e., if there was no correlation with other predictors.

Usually, features having a VIF score greater than 5 are dropped/treated till all the features have a VIF score of less than 5.

```
In [35]: from statsmodels.stats.outliers_influence import variance_inflation_factor

def checking_vif(train):
    vif = pd.DataFrame()
    vif["feature"] = train.columns

    # Calculating VIF for each feature
    vif["VIF"] = [
        variance_inflation_factor(train.values, i) for i in range(len(train.
```

```
]
return vif
```

```
In [36]: print(checking_vif(x_train1))
```

	feature	VIF
0	const	0.000000
1	Available Extra Rooms in Hospital	1.023185
2	staff_available	1.001928
3	Visitors with Patient	1.029215
4	Admission_Deposit	1.021075
5	Department_anesthesia	2.737453
6	Department_gynecology	inf
7	Department_radiotherapy	7.650799
8	Department_surgery	inf
9	Ward_Facility_Code_B	inf
10	Ward_Facility_Code_C	1.366865
11	Ward_Facility_Code_D	inf
12	Ward_Facility_Code_E	2.878866
13	Ward_Facility_Code_F	inf
14	doctor_name_Dr John	inf
15	doctor_name_Dr Mark	inf
16	doctor_name_Dr Nathan	inf
17	doctor_name_Dr Olivia	inf
18	doctor_name_Dr Sam	inf
19	doctor_name_Dr Sarah	inf
20	doctor_name_Dr Simon	inf
21	doctor_name_Dr Sophia	inf
22	Age_11-20	14.638244
23	Age_21-30	34.305313
24	Age_31-40	30.513493
25	Age_41-50	21.345110
26	Age_51-60	7.070136
27	Age_61-70	8.363349
28	Age_71-80	6.310419
29	Age_81-90	3.376418
30	Age_91-100	1.405016
31	gender_Male	inf
32	gender_Other	1.837919
33	Type of Admission_Trauma	1.345985
34	Type of Admission_Urgent	1.277917
35	Severity of Illness_Minor	2.080704
36	Severity of Illness_Moderate	2.499490
37	health_conditions_Diabetes	2.788974
38	health_conditions_Heart disease	1.969372
39	health_conditions_High Blood Pressure	1.560053
40	health_conditions_Other	1.247072
41	Insurance_Yes	1.006453

- All the continuous variables have VIF less than 5, which makes sense according to what we observed in correlation heatmap.

Note: It is not a good practice to consider VIF values for dummy variables as they are correlated to other categories and hence have a high VIF usually. In such a case, we can

check the p-values of coefficients.

Dropping the insignificant variables and creating the regression model again

Examining the significance of the model

It is not enough to just fit a multiple regression model to the data, it is also necessary to check whether all the regression coefficients are significant or not. The significance here means whether the population regression parameters are significantly different from zero.

From the above, it may be noted that the regression coefficients corresponding to `staff_available`, `Visitors with Patient`, and `Insurance_Yes` **are not statistically significant at significance level $\alpha = 0.05$** . In other words, the regression coefficients corresponding to these three are not significantly different from 0 in the population.

Suppose you have a nominal categorical variable having 4 categories (or levels). You would create 3 dummy variables ($k-1 = 4-1$ dummy variables) and set one category as a reference level. Suppose one of them is insignificant, then if you exclude that dummy variable, it would change the reference level as you are indirectly combining that insignificant level with the original reference level. It would have a new reference level and interpretation would change. Moreover, excluding the level may make the others insignificant. If all the categories in a column show p-value higher than 0.05, then we can drop that column.

Hence, we will eliminate these three features and create a new model.

```
In [37]: # Dropping variables
x_train2 = x_train1.drop(['Insurance_Yes', 'staff_available', 'Visitors with P
x_test2 = x_test1.drop(['Insurance_Yes', 'staff_available', 'Visitors with Pat

# Train the model
olsmodel2 = sm.OLS(y_train, x_train2).fit()

# Get the model summary
olsmodel2.summary()
```

Out [37]:

OLS Regression Results

Dep. Variable:	Stay (in days)	R-squared:	0.843
Model:	OLS	Adj. R-squared:	0.843
Method:	Least Squares	F-statistic:	6.307e+04
Date:	Fri, 17 Jan 2025	Prob (F-statistic):	0.00
Time:	02:47:52	Log-Likelihood:	-1.0246e+06
No. Observations:	400000	AIC:	2.049e+06
Df Residuals:	399965	BIC:	2.050e+06
Df Model:	34		
Covariance Type:	nonrobust		

	coef	std err	t	P> t	[0.025	0.975]
const	19.8886	0.053	373.758	0.000	19.784	19.993
Available Extra Rooms in Hospital	0.0786	0.002	42.391	0.000	0.075	0.082
Admission_Deposit	-3.842e-05	4.77e-06	-8.051	0.000	-4.78e-05	-2.91e-05
Department_anesthesia	6.0822	0.029	210.152	0.000	6.025	6.139
Department_gynecology	0.4628	0.019	24.518	0.000	0.426	0.500
Department_radiotherapy	-4.6217	0.037	-126.175	0.000	-4.694	-4.550
Department_surgery	9.6854	0.044	218.982	0.000	9.599	9.772
Ward_Facility_Code_B	0.2569	0.011	23.599	0.000	0.236	0.278
Ward_Facility_Code_C	0.4350	0.034	12.611	0.000	0.367	0.503
Ward_Facility_Code_D	0.2032	0.010	19.495	0.000	0.183	0.224
Ward_Facility_Code_E	0.3085	0.021	14.438	0.000	0.267	0.350
Ward_Facility_Code_F	0.0027	0.011	0.248	0.804	-0.019	0.024
doctor_name_Dr John	7.2624	0.031	236.018	0.000	7.202	7.323
doctor_name_Dr Mark	1.1947	0.033	36.462	0.000	1.130	1.259
doctor_name_Dr Nathan	-0.2125	0.015	-14.014	0.000	-0.242	-0.183
doctor_name_Dr Olivia	-0.3592	0.015	-24.646	0.000	-0.388	-0.331
doctor_name_Dr Sam	1.2833	0.038	33.914	0.000	1.209	1.358
doctor_name_Dr Sarah	0.8184	0.016	50.617	0.000	0.787	0.850
doctor_name_Dr Simon	6.1594	0.069	89.118	0.000	6.024	6.295
doctor_name_Dr Sophia	0.2162	0.020	10.584	0.000	0.176	0.256

Age_11-20	-12.7379	0.065	-194.593	0.000	-12.866	-12.610
Age_21-30	-10.4831	0.062	-168.470	0.000	-10.605	-10.361
Age_31-40	-13.4355	0.062	-216.959	0.000	-13.557	-13.314
Age_41-50	-13.5114	0.062	-216.835	0.000	-13.634	-13.389
Age_51-60	-0.3666	0.065	-5.652	0.000	-0.494	-0.239
Age_61-70	-0.4217	0.064	-6.592	0.000	-0.547	-0.296
Age_71-80	-0.2193	0.066	-3.343	0.001	-0.348	-0.091
Age_81-90	-0.0175	0.072	-0.244	0.807	-0.158	0.123
Age_91-100	-0.2599	0.112	-2.323	0.020	-0.479	-0.041
gender_Male	6.0040	0.038	157.705	0.000	5.929	6.079
gender_Other	-0.0258	0.030	-0.848	0.396	-0.085	0.034
Type of Admission_Trauma	-0.0667	0.012	-5.625	0.000	-0.090	-0.043
Type of Admission_Urgent	0.1502	0.018	8.302	0.000	0.115	0.186
Severity of Illness_Minor	0.0432	0.016	2.657	0.008	0.011	0.075
Severity of Illness_Moderate	0.0987	0.016	6.253	0.000	0.068	0.130
health_conditions_Diabetes	-0.3510	0.023	-15.028	0.000	-0.397	-0.305
health_conditions_Heart disease	0.5952	0.027	21.907	0.000	0.542	0.648
health_conditions_High Blood Pressure	-0.0369	0.017	-2.177	0.029	-0.070	-0.004
health_conditions_Other	0.0067	0.014	0.475	0.635	-0.021	0.034
Omnibus:	131768.874	Durbin-Watson:	1.998			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	708952.459			
Skew:	1.495	Prob(JB):	0.00			
Kurtosis:	8.796	Cond. No.	1.03e+16			

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[2] The smallest eigenvalue is 8.91e-20. This might indicate that there are strong multicollinearity problems or that the design matrix is singular.

Checking the performance of the model on the train and test datasets

```
In [38]: print("Performance on train data: ")
         model_performance_regression(olsmodel2, x_train2, y_train)
```

Performance on train data:

```
Out[38]:
```

	RMSE	MAE	R-squared	Adj. R-squared	MAPE
0	3.135098	2.146237	0.842812	0.842797	19.591701

```
In [39]: print("Performance on test data: ")
         lin_reg_test = model_performance_regression(olsmodel2, x_test2, y_test)
         lin_reg_test
```

Performance on test data:

```
Out[39]:
```

	RMSE	MAE	R-squared	Adj. R-squared	MAPE
0	3.144053	2.155762	0.843028	0.842967	19.676967

Observation:

- RMSE, MAE, and MAPE of train and test data are very close, indicating that the **model is not overfitting and has generalized well over the unseen data.**

Checking for assumptions and rebuilding the model

In this step, we will check for the below assumptions in the model, to verify if they hold true or not. If the assumptions of model are not satisfied, then the model might give false results. Hence, if any of the assumptions is not true, then we will rebuild the model after fixing those issues.

1. Mean of residuals should be 0
2. Normality of error terms
3. Linearity of variables
4. No Heteroscedasticity

Mean of residuals should be 0 and normality of error terms

```
In [40]: # Residuals
         residual = olsmodel2.resid
```

```
In [41]: residual.mean()
```

```
Out[41]: 5.535474656426232e-13
```


- The mean of residuals is very close to 0. Hence, the corresponding assumption is satisfied.

Tests for Normality

What is the test?

- Error terms/Residuals should be normally distributed.
- If the error terms are non-normally distributed, confidence intervals may become too wide or narrow. Once the confidence interval becomes unstable, it leads to difficulty in estimating coefficients based on the minimization of least squares.

What does non-normality indicate?

- It suggests that there are a few unusual data points that must be studied closely to make a better model.

How to check the normality?

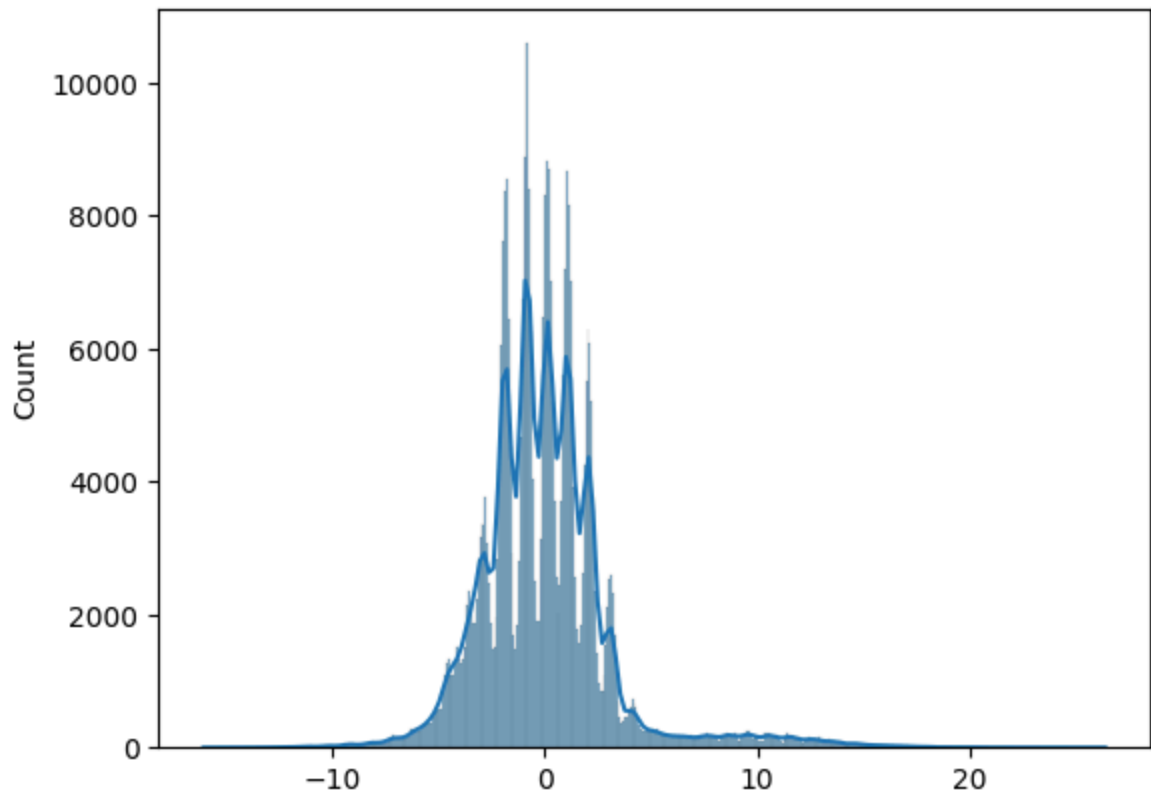
- It can be checked via QQ Plot. Residuals following normal distribution will make a straight line plot otherwise not.
- Another test to check for normality is the Shapiro-Wilk test.

What if the residuals are not-normal?

- We can apply transformations like a log, exponential, arcsinh, etc. as per our data.

```
In [42]: # Plot histogram of residuals
sns.histplot(residual, kde=True)
```

```
Out[42]: <Axes: ylabel='Count'>
```



- The residuals have a close to normal distribution. The assumption of normality is satisfied.

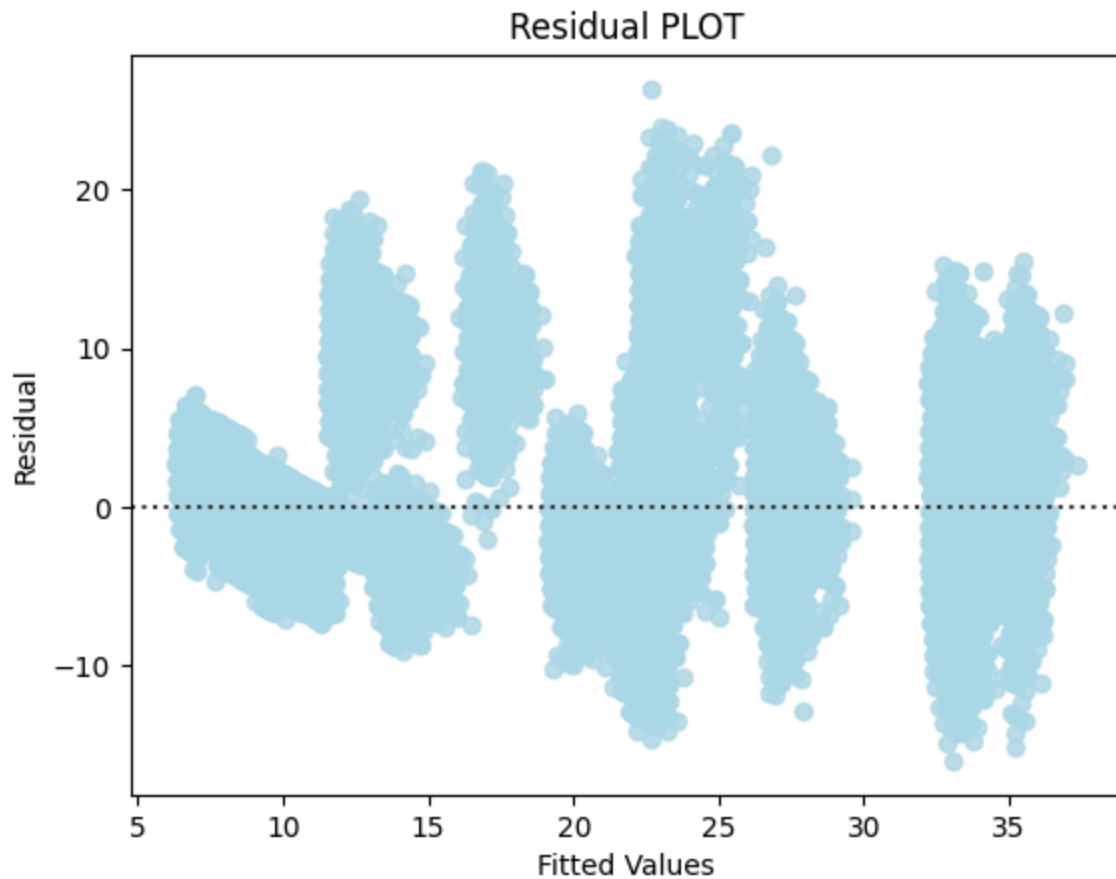
Linearity of Variables

It states that the predictor variables must have a linear relation with the dependent variable.

To test the assumption, we'll plot residuals and fitted values on a plot and ensure that residuals do not form a strong pattern. They should be randomly and uniformly scattered on the x-axis.

```
In [43]: # Predicted values
         fitted = olsmodel2.fittedvalues

         # Plotting Residuals VS Fitted Values
         sns.residplot(x = fitted, y = residual, color="lightblue")
         plt.xlabel("Fitted Values")
         plt.ylabel("Residual")
         plt.title("Residual PLOT")
         plt.show()
```



Observation:

- We can observe that there is no pattern in the residuals vs fitted values scatter plot, i.e., the linearity assumption is satisfied.

Now, let's check the final assumption.

No Heteroscedasticity

- **Homoscedasticity:** If the variance of the residuals is symmetrically distributed across the regression line, then the data is said to be homoscedastic.
- **Heteroscedasticity:** If the variance is unequal for the residuals across the regression line, then the data is said to be heteroscedastic. In this case, the residuals can form an arrow shape or any other non-symmetrical shape.
- We will use **Goldfeld–Quandt** test to check homoscedasticity:
 - **Null hypothesis:** Residuals are homoscedastic
 - **Alternate hypothesis:** Residuals are heteroscedastic
 - **alpha = 0.05**

```
In [44]: import statsmodels.stats.api as sms
from statsmodels.compat import lzip

name = ["F statistic", "p-value"]
test = sms.het_goldfeldquandt(residual, x_train2)
lzip(name, test)
```

```
Out[44]: [('F statistic', 1.00391641464113), ('p-value', 0.19107310271684264)]
```

Observation:

- As we can observe from the above test, the p-value is greater than 0.05. So, we fail to reject the null-hypothesis, i.e., residuals are homoscedastic.

All the assumptions for the linear regression model are satisfied. With our model's adjusted R-squared value of around 0.84, we can capture **84% of the variation** in the data.

The p-values for the independent variables are less than 0.05 in our final model, indicating that they are statistically significant toward Length of Stay (in days) prediction.

Now, let's check and interpret the coefficients of the model.

```
In [45]: coef = olsmodel2.params
coef
```

```
Out[45]: const 19.888611
Available Extra Rooms in Hospital 0.078611
Admission_Deposit -0.000038
Department_anesthesia 6.082191
Department_gynecology 0.462834
Department_radiotherapy -4.621748
Department_surgery 9.685354
Ward_Facility_Code_B 0.256912
Ward_Facility_Code_C 0.435021
Ward_Facility_Code_D 0.203221
Ward_Facility_Code_E 0.308490
Ward_Facility_Code_F 0.002702
doctor_name_Dr John 7.262411
doctor_name_Dr Mark 1.194666
doctor_name_Dr Nathan -0.212506
doctor_name_Dr Olivia -0.359233
doctor_name_Dr Sam 1.283345
doctor_name_Dr Sarah 0.818361
doctor_name_Dr Simon 6.159364
doctor_name_Dr Sophia 0.216213
Age_11-20 -12.737881
Age_21-30 -10.483082
Age_31-40 -13.435514
Age_41-50 -13.511377
Age_51-60 -0.366602
Age_61-70 -0.421730
Age_71-80 -0.219312
Age_81-90 -0.017517
Age_91-100 -0.259892
gender_Male 6.004001
gender_Other -0.025809
Type of Admission_Trauma -0.066684
Type of Admission_Urgent 0.150156
Severity of Illness_Minor 0.043165
Severity of Illness_Moderate 0.098727
health_conditions_Diabetes -0.350972
health_conditions_Heart disease 0.595161
health_conditions_High Blood Pressure -0.036897
health_conditions_Other 0.006721
dtype: float64
```

```
In [46]: # Let us write the equation of the model
Equation = "Stay (in days) ="
print(Equation, end='\t')
for i in range(len(coef)):
    print('(', coef[i], ') * ', coef.index[i], '+', end = ' ')
```

$$\begin{aligned} \text{Stay (in days)} = & (19.88861085586438) * \text{const} + (0.07861088283159079) * \\ & \text{Available Extra Rooms in Hospital} + (-3.842459927514132e-05) * \text{Admission_} \\ & \text{Deposit} + (6.082190541593219) * \text{Department_anesthesia} + (0.4628344124822 \\ & 663) * \text{Department_gynecology} + (-4.621748096792417) * \text{Department_radiotherapy} + (9.685354270363284) * \\ & \text{Department_surgery} + (0.2569121669054769) * \text{Ward_Facility_Code_B} + (0.43502142725846366) * \text{Ward_Facility_Code_C} \\ & + (0.20322070575899695) * \text{Ward_Facility_Code_D} + (0.3084901213984358) * \\ & \text{Ward_Facility_Code_E} + (0.0027015398090287773) * \text{Ward_Facility_Code_F} + (7.262410614557624) * \\ & \text{doctor_name_Dr John} + (1.1946664296565659) * \text{doctor_name_Dr Mark} + (-0.21250613917350206) * \\ & \text{doctor_name_Dr Nathan} + (-0.3592329257012288) * \text{doctor_name_Dr Olivia} + (1.2833451288128765) * \text{doctor_name_Dr Sam} + (0.8183606659496501) * \\ & \text{doctor_name_Dr Sarah} + (6.159364331852158) * \text{doctor_name_Dr Simon} + (0.21621281139694046) * \text{doctor_name_Dr Sophia} + (-12.737881405975003) * \\ & \text{Age_11-20} + (-10.483081659420062) * \text{Age_21-30} + (-13.435513797183848) * \text{Age_31-40} + (-13.511376906568888) * \\ & \text{Age_41-50} + (-0.366601795753167) * \text{Age_51-60} + (-0.42173008604893825) * \text{Age_61-70} + (-0.21931186150246826) * \\ & \text{Age_71-80} + (-0.01751716731957309) * \text{Age_81-90} + (-0.2598922911130551) * \text{Age_91-100} + (6.004001496980577) * \\ & \text{gender_Male} + (-0.025808757078083618) * \text{gender_Other} + (-0.0666842357085861) * \text{Type of Admission_Trauma} + (0.15015611007282056) * \text{Type of Admission_Urgent} + (0.04316457003788236) * \\ & \text{Severity of Illness_Minor} + (0.0987269422443891) * \text{Severity of Illness_Moderate} + (-0.3509723193864268) * \text{health_conditions_Diabetes} + (0.595161388375446) * \text{health_conditions_Heart disease} + (-0.03689689844279388) * \text{health_conditions_High Blood Pressure} + (0.00672149097736913) * \text{health_conditions_Other} + \end{aligned}$$

Interpreting the Regression Coefficients

- The Stay (in days) decreases with an increase in Department_radiotherapy. 1 unit increase in the Department_radiotherapy leads to a decrease of Stay (in days) ~ 4.62 times the Stay (in days) than the Department_TB&Chest_Disease that serves as a reference variable when everything else is constant.
- The Stay (in days) increases with an increase in Department_anesthesia. 1 unit increase in Department_anesthesia leads to an increase of Stay (in days) ~ 6.08 times the Stay (in days) than the Department_TB&Chest_Disease that serves as a reference variable when everything else is constant. This is understandable, as anesthesia is used in severe cases which results in more days of stay.
- The Stay (in days) increases with an increase in Department_surgery. 1 unit increase in Department_surgery leads to an increase of Stay (in days) ~ 9.68 times the Stay (in days) than the Department_TB&Chest_Disease that serves as a reference variable when everything else is constant. This is understandable, as surgery is conducted in severe cases which results in more days of stay.
- The Stay (in days) increases with an increase in doctor_name_Dr Simon. 1 unit increase in doctor_name_Dr Simon leads to an increase of Stay (in days) ~ 6.14 times the Stay (in days) than the doctor_name_Dr Isaac that serves as a reference

variable when everything else is constant. This is understandable, as surgery cases are handled by Dr. Simon.

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 verified that the assumptions of Linear Regression are being satisfied, to make our final model statistically correct to draw inferences from.
- 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.

```
In [47]: data.corr()
```

Out [47]:

	Available Extra Rooms in Hospital	staff_available	Visitors with Patient	Admission_Deposit
Available Extra Rooms in Hospital	1.000000	-0.001784	0.070459	-0.050127
staff_available	-0.001784	1.000000	0.000578	0.000763
Visitors with Patient	0.070459	0.000578	1.000000	-0.069043
Admission_Deposit	-0.050127	0.000763	-0.069043	1.000000
Stay (in days)	-0.019219	0.007398	0.027302	0.044203
Department_anesthesia	-0.002900	-0.004316	0.027194	0.030720
Department_gynecology	0.045627	-0.005341	-0.058815	-0.042772
Department_radiotherapy	-0.047900	0.000147	0.023313	0.019934
Department_surgery	-0.005756	0.043335	-0.006618	0.003054
Ward_Facility_Code_B	-0.049206	-0.002726	-0.080750	0.006685
Ward_Facility_Code_C	-0.019121	-0.001632	0.007937	-0.033609
Ward_Facility_Code_D	0.065034	0.000375	0.026025	-0.091519
Ward_Facility_Code_E	0.000332	-0.003850	0.097181	0.022285
Ward_Facility_Code_F	0.031380	-0.003578	-0.013087	0.038421
doctor_name_Dr John	-0.022992	-0.002268	0.030849	0.022800
doctor_name_Dr Mark	-0.005498	-0.004116	0.044871	0.028371
doctor_name_Dr Nathan	0.009957	-0.003938	-0.013607	-0.016453
doctor_name_Dr Olivia	0.018587	-0.000995	-0.016432	-0.010668
doctor_name_Dr Sam	-0.038263	-0.000109	0.018494	0.014427
doctor_name_Dr Sarah	0.015735	-0.001095	-0.024254	-0.012448
doctor_name_Dr Simon	-0.002856	0.024937	-0.004652	0.001208
doctor_name_Dr Sophia	0.011258	-0.000760	-0.017703	-0.013705
Age_11-20	-0.010645	0.000382	-0.026922	0.007289
Age_21-30	-0.012317	-0.001562	-0.033020	0.016350
Age_31-40	0.056501	-0.001813	0.016700	-0.037817
Age_41-50	0.013918	0.000281	0.030415	-0.028265
Age_51-60	-0.031621	0.001266	-0.000369	0.018392
Age_61-70	-0.033325	0.002176	0.008050	0.023397
Age_71-80	-0.024464	0.000905	-0.003775	0.026680

	Available Extra Rooms in Hospital	staff_available	Visitors with Patient	Admission_Deposit
Age_81-90	-0.019761	0.003327	0.011166	0.011337
Age_91-100	0.000673	-0.001783	0.011447	0.008901
gender_Male	-0.034592	0.004147	0.044909	0.031710
gender_Other	-0.015330	0.001771	0.019560	0.017880
Type of Admission_Trauma	-0.035638	0.000400	-0.024419	0.016784
Type of Admission_Urgent	0.047514	-0.002287	0.000235	0.003466
Severity of Illness_Minor	0.023532	-0.001796	-0.010046	0.015279
Severity of Illness_Moderate	0.008689	0.002228	0.028970	-0.003885
health_conditions_Diabetes	-0.001340	-0.001305	-0.001514	-0.000181
health_conditions_Heart disease	-0.000081	0.000567	-0.001718	-0.001711
health_conditions_High Blood Pressure	0.001700	-0.000479	-0.005739	0.000718
health_conditions_Other	0.001230	-0.000805	-0.004961	0.001630
Insurance_Yes	0.001900	0.001481	0.032858	0.002609

In []: