Project :- Healthcare Analytics to Optimize Hospital Resource Management.

Final Project Report

Gavin Gunawardena

gavin.gunawardena@trojans.dsu.edu

1. Task

The task is to accurately predict the Length of Stay for each patient on a case-by-case basis so that the Hospitals can use this information for optimal resource allocation and better functionality. The length of stay in the original dataset is divided into 11 different classes ranging from 0-10 days to more than 100 days. The final goal of this project would be to help management to optimize staff allocation and functionality of hospitals by analyzing what patient details tend to minimize patient stay duration. This would in turn allow the hospital to efficiently accommodate the incoming patients, maximizing the benefit it provides to its community as well as the profits it generates to sustain itself.

2. Approach

For the project implementation the plan is to fit a "Classification Model" to accurately predict the length of stay for patients at the hospital based on multiple independent parameters. Our approach towards model building proceeds with the following steps of implementation.

A. Data Formatting and Cleaning
    a. Identifying outliers and missing values.
    b. Removing outliers and imputing missing values
B. Libraries: - Numpy , Pandas, Seaborn, Plotly, Sklearn , Matplotlib
C. Platforms Used :- Python
D. Exploration and Data Analysis: - Explore data analysis and identify the correlation between variables and those variables impacting the Predictor significantly.
    a. Plotting visuals to interpret the relationship between the variables.
    b. Displaying meaningful insights based on the data.
E. Feature Engineering
    a. Optimizing the dataset for model buildings via actions such as:
        i. Dummy Encoding
        ii. Data Imputation
        iii. Converting strings to numerics
        iv. Data Scaling
F. Model Building
    a. Data Split
        i. 70% training
        ii. 30% Validation
    b. Model Selection: -
        i. Decision Tree
        ii. Random Forest
        iii. Stochastic Gradient Descent
    c. Hyperparameter Tuning

        i. Grid Search
    d. Model Scoring Metric
        i. Accuracy

G. Model Validation:
    a. Comparison of accuracy rates
    b. Confusion Matrices

H. Test Dataset
    a. Run final model on unlabeled test dataset

3. Dataset and Metric

For this project, a Healthcare Analytics Dataset retrieved from the Kaggle website will be used.

Link to dataset:

- https://www.kaggle.com/nehaprabhavalkar/av-healthcare-analytics-ii?select=healthcare

This dataset includes around 137,000 test examples and around 318,000 training examples. Data preprocessing that seemed necessary for this dataset includes: missing value imputation, dummy encoding for categorical variables, and grouping/changing of bucket ranges of categorical variable: Stay. Data Categorization of variables in the dataset:

Ordinal/Discrete	Nominal/discrete	Categorical
Case_id	Hospital_Code	Hospital_type_code
Available Extra Rooms in Hospital	City_Code_Hospital	Hospital_region_code
Bed Grade	Patientid	Department
	City_Code_Patient	Ward_Type
	Visitors with Patient	Ward_Facility_Code
	Admission_Deposit	Type of Admission
		Severity of Illness
		Age
		Stay

4. Results

Results based on preliminary work:

1. Missing Values identified in the dataset.

    a. For the attribute “Bed Grade” that represents the condition of the beds available, there are 35 records missing in the test data and 113 records missing in the training data. These will be imputed with a missing value indicator

b. There are a total of 2192 records missing in the “City_Code_Patient” field of the test data and 4,532 in the training data, representing the zip code for the patient's city. These will be imputed with a missing value indicator.

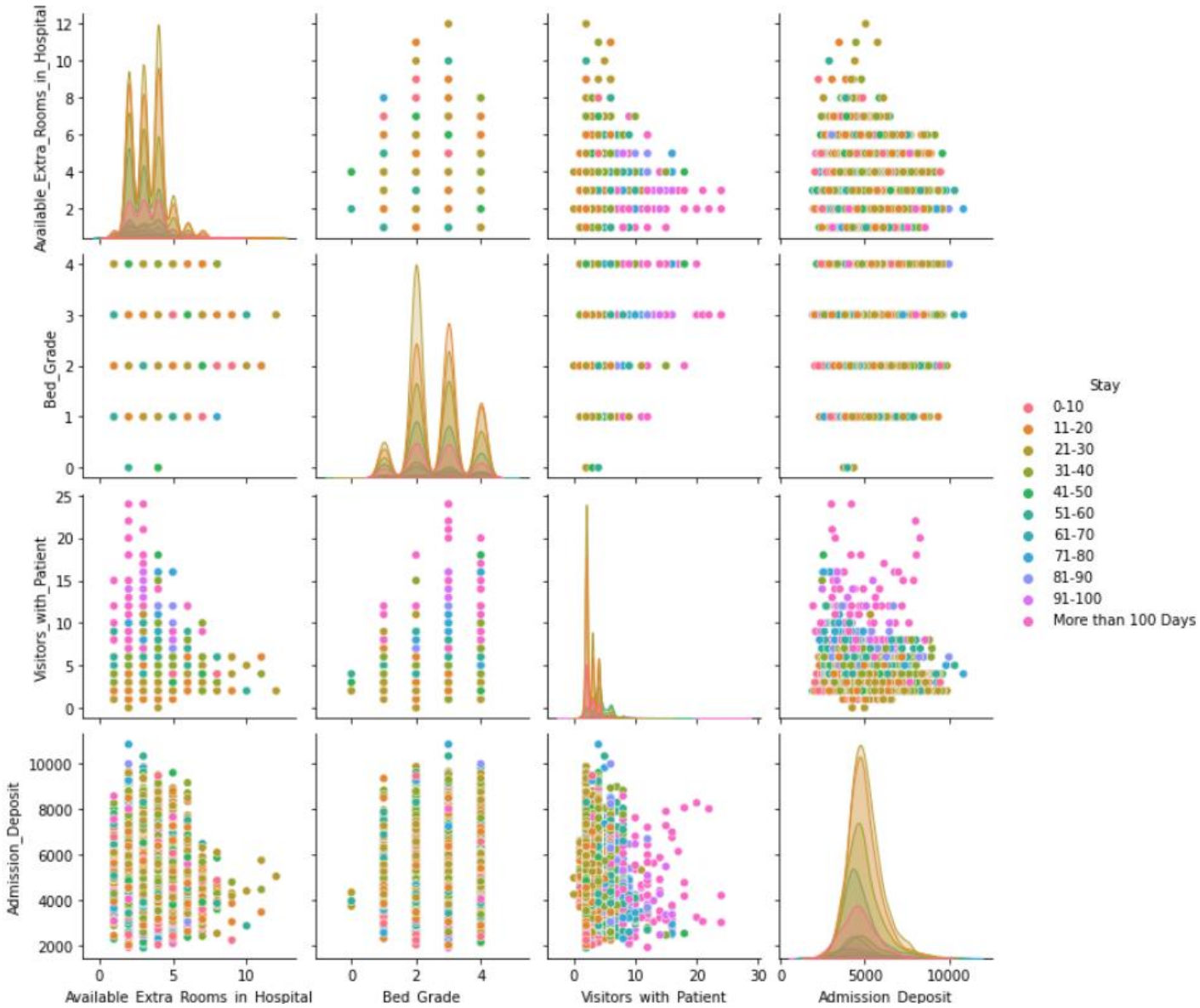
2. Dummy Encoding of the variables.

The “Stay” categorical variable which is also the dependent variable will have its amount of categories lowered from around 10 to 5 via grouping. Age ranges were also grouped into fewer categories. The categorical variable “Severity of Illness” was changed to a numeric. The variables utilizing dummy encoding were all categorical independent variables.

3. Results from Exploratory Data Analysis.

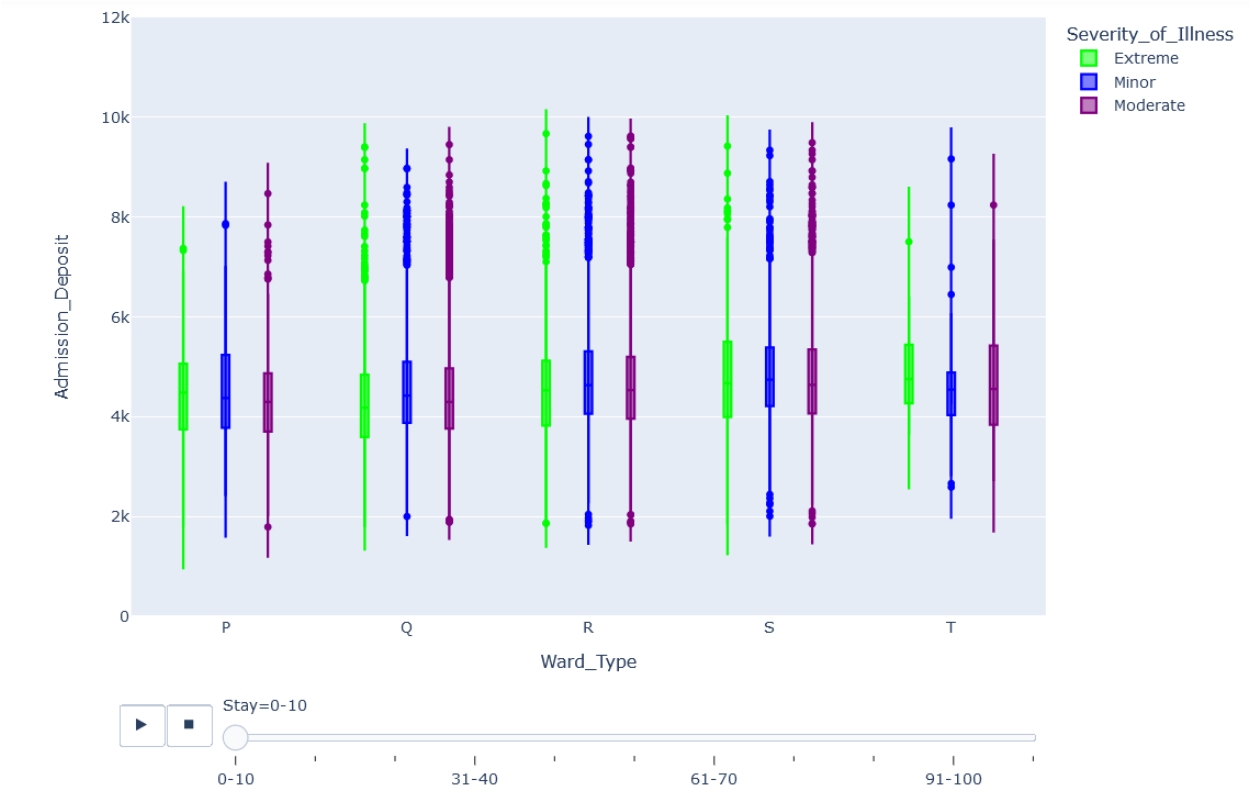
Pair Plot:

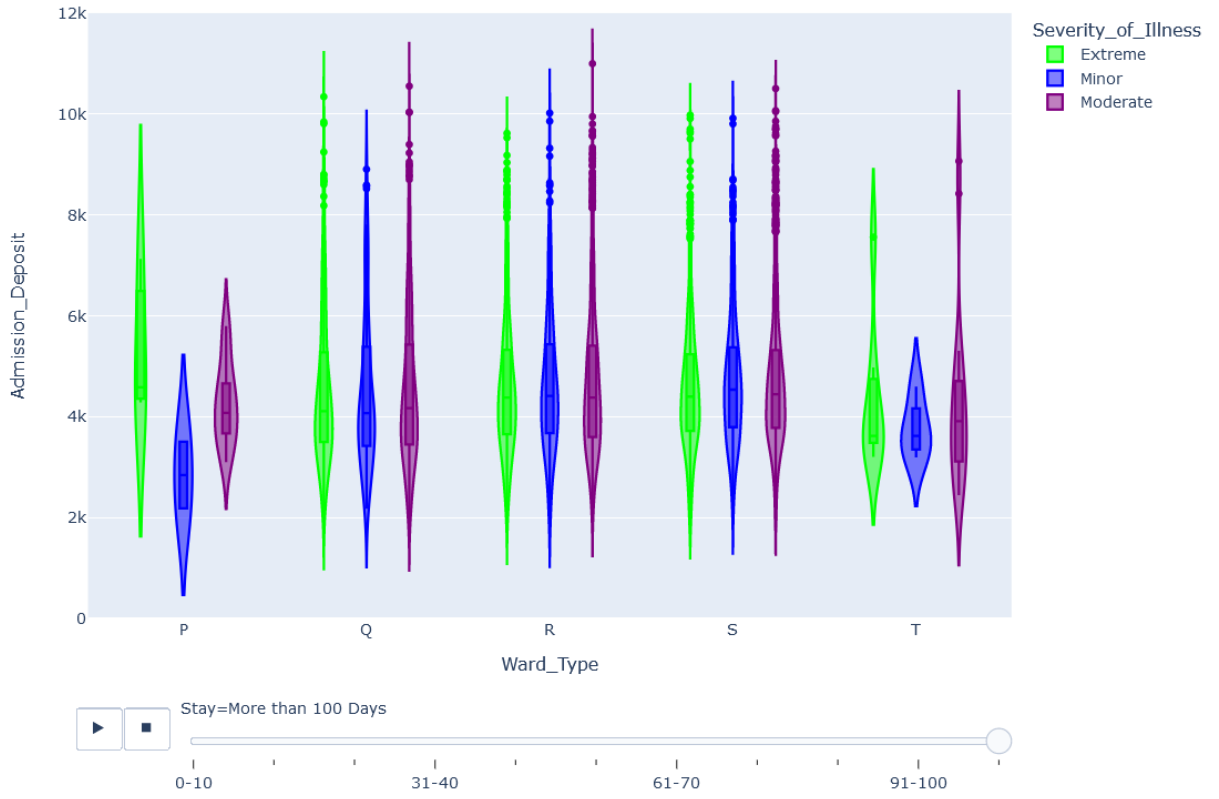
The following pair plot compares the discrete, continuous, and ordinal variables in the dataset with each other and the dependent variable, “Stay”, in order to find correlation and separation. It indicates a definite separation of classes for the “Stay”, independent variable, when the dependent variable, “Visitors_with_Patient” is involved, indicating that this would likely be a great candidate for use as a predictor variable.

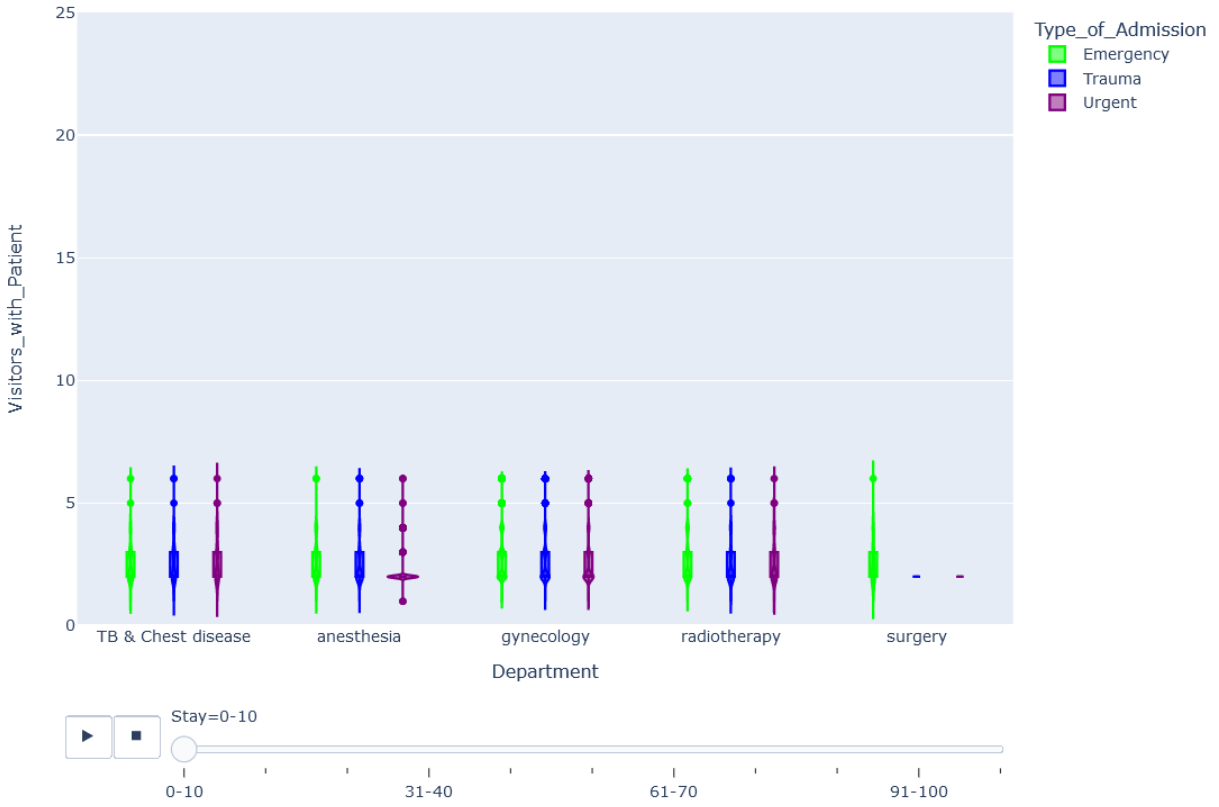


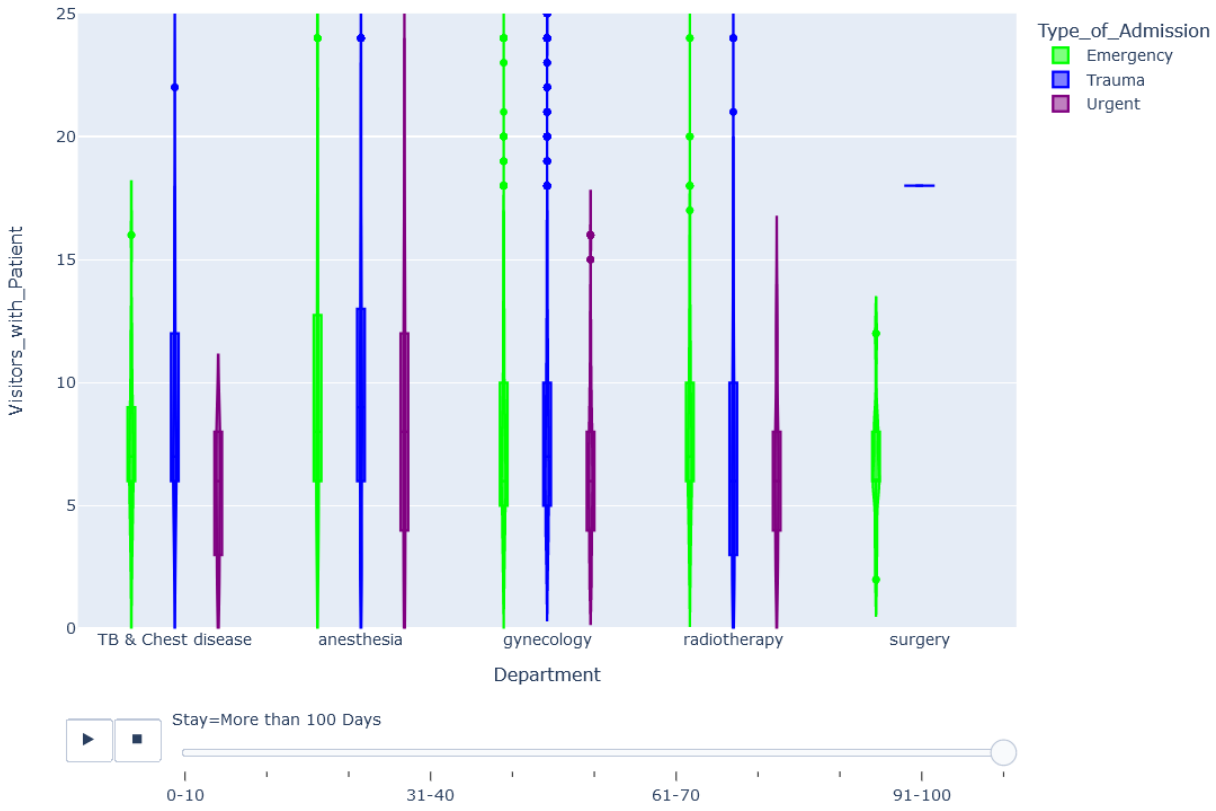
Violin Plots:

The first two violin plots compare shorter and longer stays of patients with their admission deposit, ward type, and illness severity. The third and fourth violin plots compare shorter and longer stays of patients with the amount of visitors they had at admission, the department they were admitted into, and the type of admission which is based on their condition. As shown on the violin plots, there's an obvious difference in density of patients paying less in admission deposits for longer hospital stays. Also, the distribution of patients assigned to the various wards tends to become more even with longer stay periods, which may indicate that patients are moved between wards when they stay at the hospital for longer periods of time. Furthermore, the amount of visitors a patient gets is a major tell on whether they'd be staying longer, as shown on the 3rd and 4th violin plots.



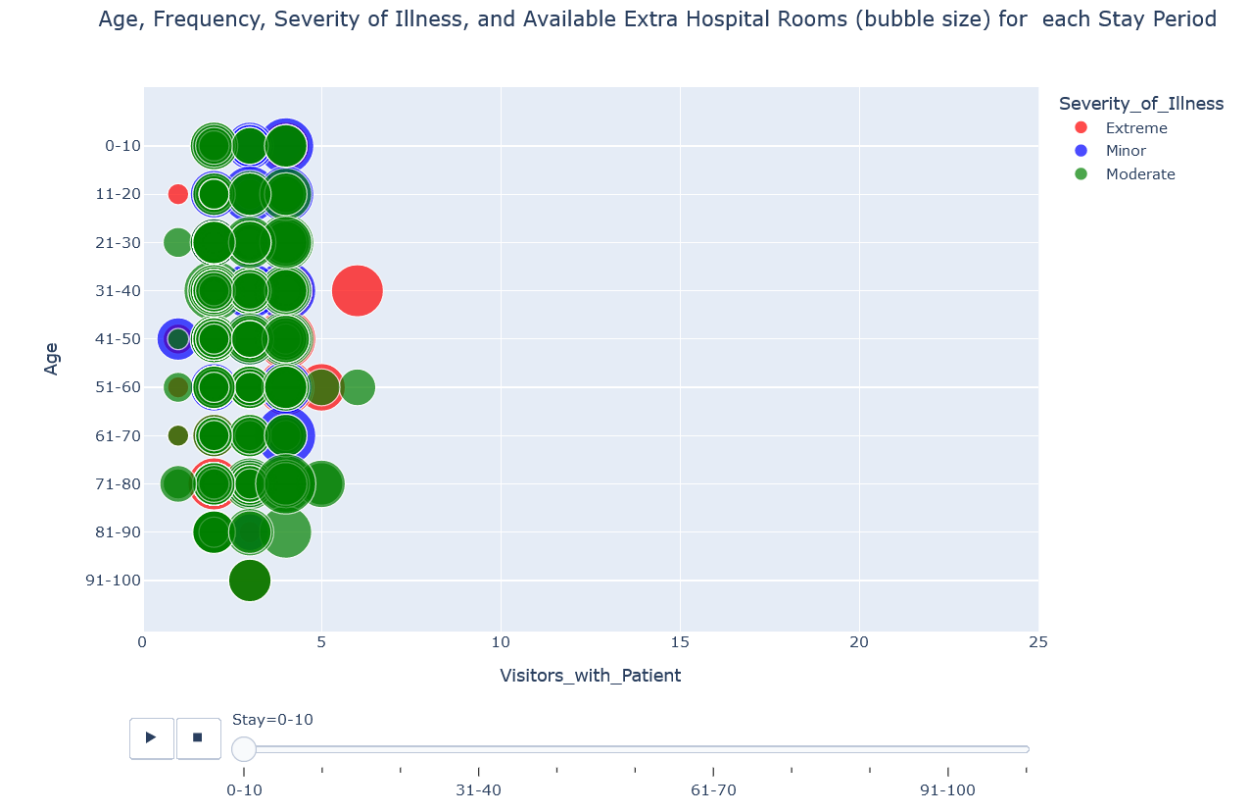

Admission_Deposit
12k
10k
8k
6k
4k
2k
0
P
Q
R
S
T
Ward_Type
Severity_of_Illness
Extreme
Minor
Moderate
Stay=More than 100 Days
0-10
31-40
61-70
91-100

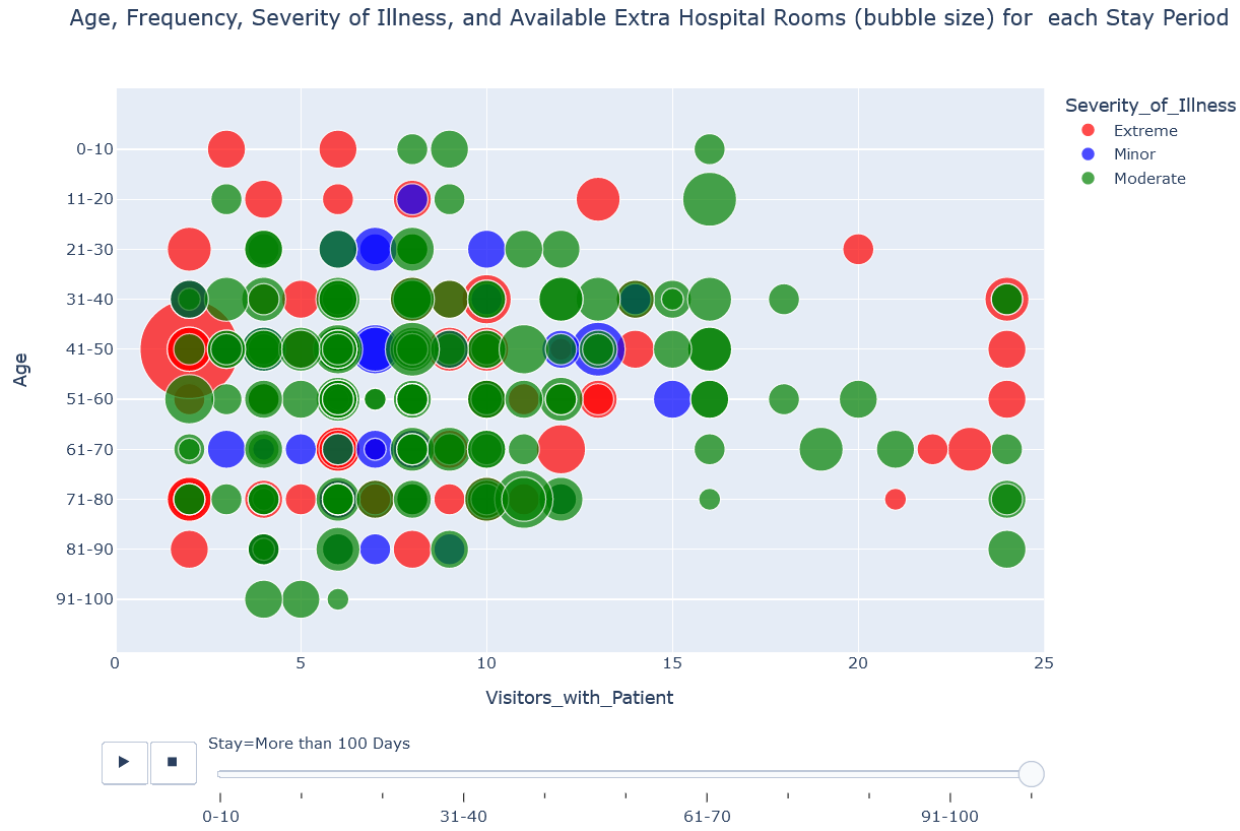

Visitors_with_Patient
25
20
15
10
5
0
TB & Chest disease
anesthesia
gynecology
radiotherapy
surgery
Department
Type_of_Admission
Emergency
Trauma
Urgent
Stay=0-10
0-10
31-40
61-70
91-100



Bubble Plot (Size represents Available Extra Rooms in Hospital):

The following bubble plot compares stay length of hospital patients with their age, the visitors they had on admission, their illness severity, and the available extra rooms in the hospital at admission which is indicated by bubble size. They indicate that patients with shorter stay lengths tend to have less visitors when they are admitted although not much else as shown by the spread of colors, sizes and by the vertical spread.


Age, Frequency, Severity of Illness, and Available Extra Hospital Rooms (bubble size) for each Stay Period
Severity_of_Illness
Extreme
Minor
Moderate
Age
0-10
11-20
21-30
31-40
41-50
51-60
61-70
71-80
81-90
91-100
0
5
10
15
20
25
Visitors_with_Patient
Stay=0-10
0-10
31-40
61-70
91-100


Age, Frequency, Severity of Illness, and Available Extra Hospital Rooms (bubble size) for each Stay Period
Severity_of_Illness
Extreme
Minor
Moderate
Age
0-10
11-20
21-30
31-40
41-50
51-60
61-70
71-80
81-90
91-100
0
5
10
15
20
25
Visitors_with_Patient
Stay=More than 100 Days
0-10
31-40
61-70
91-100

Checking for and Updating Dataset to compensate for Class Imbalance

This was done by first checking for minority classes and then oversampling them in the training dataset after splitting the dataset into training and validation. This was done so that the models could be trained on rows(observations) for each class.:

This…:

	category	rows	rows %
0	1	132395	59.394992
1	2	71554	32.100527
2	3	12382	5.554808
3	4	6575	2.949674

became this after oversampling.:

	category	rows	rows %
0	1	132395	25.0
1	2	132395	25.0
2	3	132395	25.0
3	4	132395	25.0

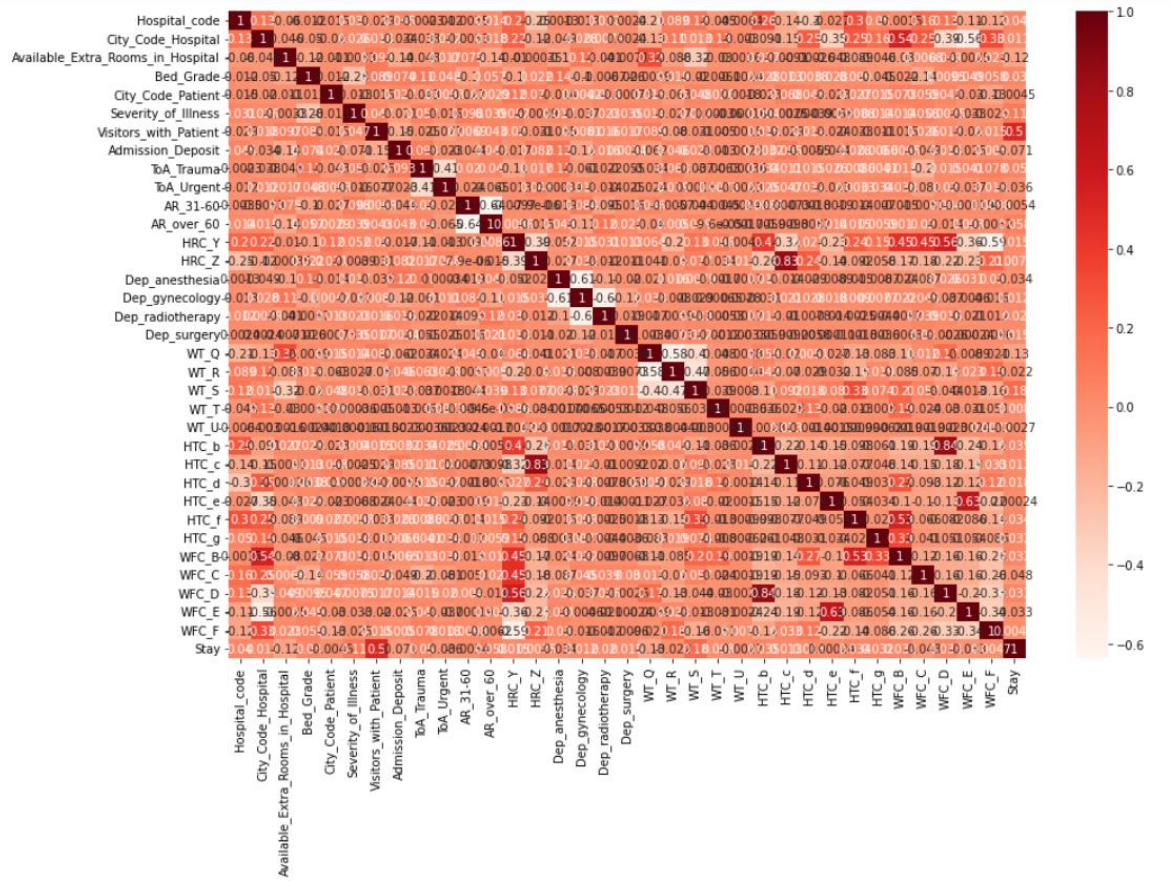
Results based on the intermediary tasks.

1. Variable Selection

Variable selection was tested via Pearson Coefficients (with r values above .0) and Ridge Regression Coefficients (with coefficients equal to or above the mean coefficient of all of the variables). Pearson Coefficient method was ultimately decided on as the variable selection measure due to its chosen variables having better results in testing.

Pearson Coefficient results:

The table below displays the selected variables and their r values.



	Stay
Stay	1.000000
Visitors_with_Patient	0.534789
WT_S	0.195805
Severity_of_Illness	0.158758
ToA_Trauma	0.109565
Bed_Grade	0.076540
HC_26	0.076522
AR_81-90	0.060214
AR_71-80	0.054879
HTC_b	0.054119
WFC_D	0.052453
CCH_2	0.052453
HC_2	0.051608
HC_13	-0.040738
HC_19	-0.042941
AR_31-40	-0.045211
AR_11-20	-0.053175
HC_24	-0.055020
Admission_Deposit	-0.057710
HC_27	-0.058997
ToA_Urgent	-0.061866
AR_21-30	-0.062554
WFC_C	-0.072183
CCH_7	-0.072183
Available_Extra_Rooms_in_Hospital	-0.151007
WT_Q	-0.165738

Ridge Regression results:

	col_name	Above_Thres			
0	Available_Extra_Rooms_in_Hospital	True	14	AR_81-90	True
12	AR_61-70	True	13	AR_71-80	True
1	Bed_Grade	True	61	HC_24	True
27	WT_T	True	11	AR_51-60	True
26	WT_S	True	9	AR_31-40	True
25	WT_R	True	10	AR_41-50	True
24	WT_Q	True	2	Severity_of_Illness	True
22	Dep_radiotherapy	True	3	Visitors_with_Patient	True
21	Dep_gynecology	True	5	ToA_Trauma	True

Variables for Ridge Regression were selected based on whether their coefficient was higher than .03.

2. Feature Engineering:- Changes were made to the existing variables to tune our dataset and make it more explainable to the models. Categorical ordinal variables were converted to numerics based on their ordinality.

- Severity of Illness Numeric:-(Key: 0=Minor, 1=Moderate, 2=Severe)
- Department:(Key: 0=TB & Chest disease, 1=anesthesia, 2=radiotherapy, 3=gynecology)
- Stay:-(#Key: 1:['0-10','11-20'] 2:['21-30','31-40'], 3:['41-50','51-60''],4:['61-70','71-80'],4:['81-90','91-100'],5:['More than 100 Days'])

3. Data Standardization

We performed data standardization to scale our data into the same range of values, since our dataset consisted of several different parameters. We utilized Sci-Kit Learn's Standard Scaler and fit it on the training dataset. Then we used the same object to transform Test and Validation datasets so that features of the dataset have the same weights in order to improve model accuracy and performance.

4. Data Split

This step consisted of a split of the training dataset into training and validation datasets. The split ratio we decided on was 70% for training and 30% for validation.

5. Model Testing

It was decided to implement 3 models, Decision Tree, Random Forest, and Stochastic Gradient Descent. Models are selected based on their popularity and accuracy with classification problems, and also the ability of decision tree and random forest models to provide visuals of their decision process which can be used for convincing stakeholders and helping them understand the logic behind model decisions.

6. Model Tuning

Experiments we made to evaluate our approach were all based on trying to maximize the accuracy rates of our models. When first running our models, we were surprised to find very low accuracy rates on our test datasets. These accuracy rates were between 40 and 50 percent and thus we tried various methods to improve them.

At first we tried hyperparameter tuning, but via this, we were only able to improve accuracy rates by around 5% on average. For this project, we decided to, as mentioned earlier, utilize decision tree, random forest, and stochastic gradient descent. Due to the nature of these models, we had some flexibility in regard to hyperparameters to possibly solve the accuracy rate issues we were facing. With the decision tree, we were able to make a gain of about 9% in accuracy on the test dataset via lowering the max depth, increasing the minimum samples split, and adjusting the criterion of the splits from gini to entropy. With the random forest model, we were able to improve accuracy by about 5% via similar adjustments as the ones we made for the decision tree model. Finally for stochastic gradient descent, we were not able to make more than a decimal value impact on the accuracy rate by adjusting the hyperparameters and thus gave up after cycling through many of the different options it allowed. These adjustments included adjusting the loss function and penalty options. We did find that changing the loss function for SGD from hinge to log and penalty from L2 to L1 allowed us some accuracy gains, but as mentioned earlier, they were only improvements by about three tenths. This hyperparameter testing was automated via grid search in scikit learn.

Eventually we realized that we could make great gains on our accuracy rates by “moving the goalposts”, or changing our end goal to something that was much easier to attain. Originally, our dependent variable had 11 different possible categories. Adjusting this to 3 increased the accuracy rates of our models by about 30%. We settled on this method and saw it as a success as it allowed us to obtain satisfactory accuracy rates (near 80%) for a machine learning model while not further complicating our processes.

Results Based on the Final Work.

1. Final Model Selected

Random Forest was chosen as the final model as it gives us a very high accuracy rate on the validation dataset, as well as the highest F1 score, which is the harmonic mean of recall and validation. A high performing F1 score indicates better accuracy on all of the classes, which is important as the validation dataset is highly imbalanced.:

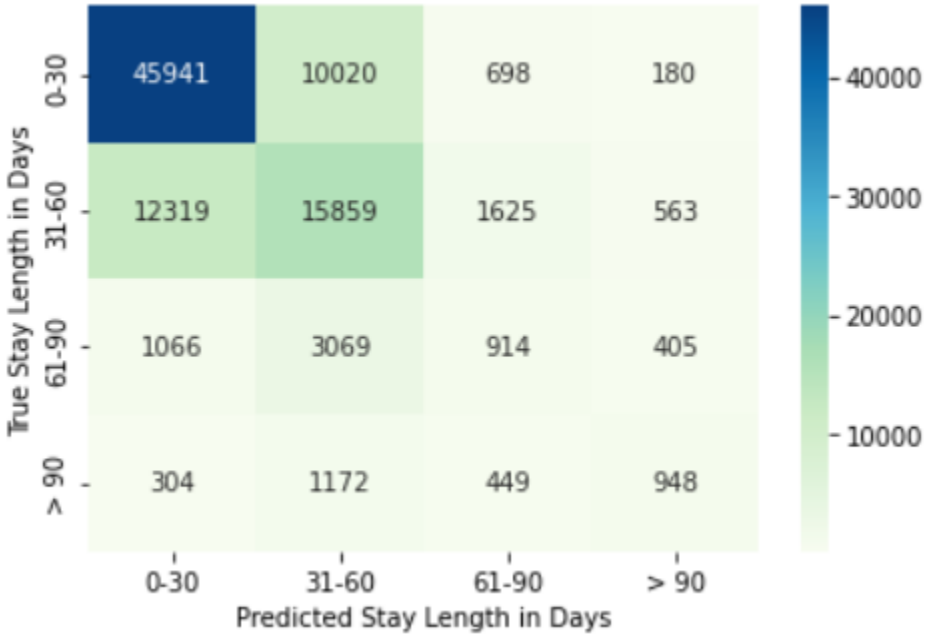
2. Displaying Model Accuracy Score Results

	Model	Prediction Accuracy - Validation	Recall - Validation	Precision - Validation	F1 - Validation
0	Decision Tree	61.01	43.81	43.09	43.44
1	Random Forest	66.64	45.70	49.93	47.37
2	Stochastic Gradient Descent	60.14	48.77	39.61	39.13
3	XGBoost	65.82	51.80	46.37	39.13

3. Displaying Confusion Matrix of our chosen model

A confusion matrix was made via the chosen model to compare the true values to the predicted values for the decision tree.

Random Forest Confusion Matrix:



This confusion matrix shows how the random forest was very accurate with the majority of the data but still made many misclassifications, especially for the 21-40 and 0-20 categories which it often incorrectly predicted to be part of each other.

4. Displaying End Results on the Test Dataset Predictions

Stay	Count
0-20	89908
21-40	19452
41-60	17406
61-80	10291