greatlearning

Capstone Project – Report

Project Title	**Predicting a patient's LOS in a hospital**
Domain of Project	**Healthcare**
Team members	**Debesh Das** **Saptarshi Sit , Chirag Rai**
Mentor Name	**P V Subramanian**

Signature of the Mentor

greatlearning
Learning for Life

Table of Contents

1. Summary of problem statement, data & findings

1.1 Problem statement

Data Science has revolutionized healthcare and the medical industry in large ways. Inspired by this, we are using Data Science techniques to build a ML based web app which will help predict Length of Stay (LOS) of a patient at the time of his admission in a hospital, to help the hospital's resource planning and patient expectations on their LOS.

LOS of patients is a vital determinant in the efficiency of a hospital's management system, use of medical services, patient's quality of care, and functional evaluation. Traditional approach used by hospitals which involved manual consultancy with doctors, resulted in delayed and not-so-accurate predictions of LOS.

LOS is an important metric for assessing the quality of care and planning capacity within a hospital. It is a key performance indicator for the Department of Health, used to monitor hospital quality and manage patient expectation. The length of time patients spend in hospital beds is known to be a good representation of the amount of resource utilised, for example bed capacity, staffing and equipment. Average LOS should therefore be published by operation and hospital on the Department of Health's website to help patients make choices on which hospital to attend.

Hospitals are constantly adapting to clinical and financial pressures driven by policy changes, including recent attention towards reducing LoS where differences between hospitals are shown to vary widely. This continuous pressure for improvement requires hospitals to review their processes to become more cost efficient and more standardised to improve patient expectation. Gaining a better understanding of LoS provides an opportunity to reduce the time patients stay in hospital.

Recent Covid-19 Pandemic has raised alarms over the efficiency of management of hospitals. Every Covid-treatment hospital was almost full of patients occupying every available bed. Hence, LOS is one critical parameter to observe and predict if one wants to improve the efficiency of the healthcare management in a hospital.

1.2 Data

This project uses EPR data to ascertain which variables significantly affect LoS and, using these, produces a model to predict future LOS, helping to improve resource planning and enabling the creation of a decision support tool to be used for improving patient expectations.

The dataset is taken from a hackathon conducted by Analytics Vidhya in the Healthcare Analytics domain, which was later uploaded on Kaggle.

1.3 Findings

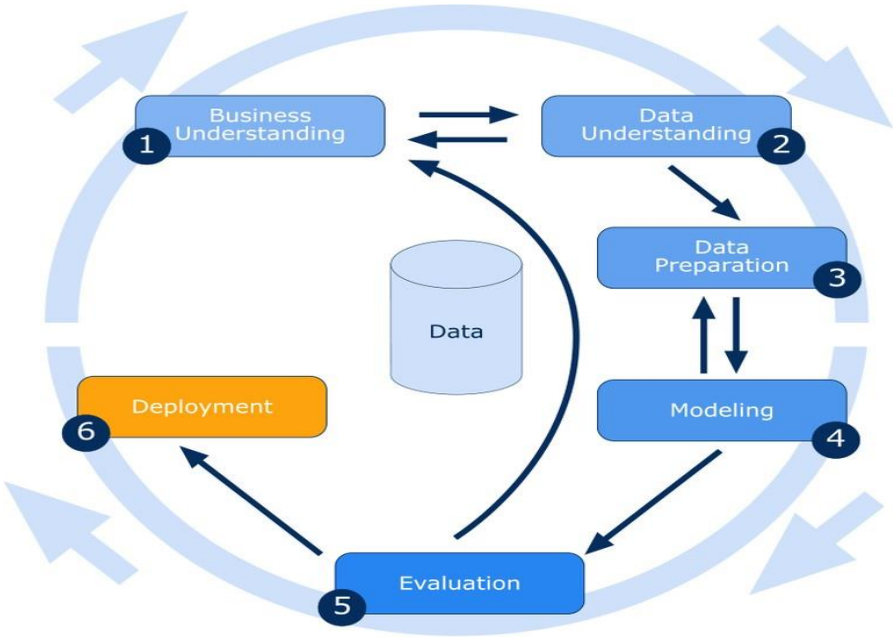
We found that most of the hospitals use the below mentioned approaches for LOS prediction

i. Manual consultancy with doctors and hospital staff about LOS for each incoming patient. This requires a lot of time of doctors' intervention as the process is executed for each patient.

ii. Use of spreadsheets in compiling, reporting, and graphically depicting statistical data. This requires technical domain expertise and is not so user-friendly.

Absence of accurate LOS knowledge beforehand, causes problems to both hospital and patient. Wrong or inadequate knowledge of LOS, leads to patient receiving inadequate treatment and more expenses. For a hospital, it results in poor bed management, improper time scheduling & inefficient treatment.

2. Overview of the final process



3. Step-by-step walkthrough of the solution

- Columns were segregated as categorical and numerical according to information & logic.
- Since age column contained ranges e.g 51-60, 31-40 , etc , so we extracted only the upper bound value of age.
- We dropped ID columns & other non-important columns by logic.
- From EDA, we inferred that there are very few columns which are acting as good predictors. Hence we realised that we have to use complex non-linear algorithms to properly capture the data pattern as much as possible.
- Then we checked for missing values. We noticed that only one column had very few missing values. Hence we dropped the records which contained missing values.
- Upon checking for outliers, we noticed that all the three numerial columns contained outliers. Without dropping them we proceeded further.
- For Encoding of categorical variables
  - Manual encoding was done for ordinal data
  - One-hot encoding was done for nominal data
- Standard Scaler was applied to the data which was to be used in distance-based algorithms
- Feature Selection was done by calculating feature_importances_ ratio of each variable.
- SMOTE technique was then applied as the target column was imbalanced.
- Then we started building models using scaled data for distance-based algorithms & non-scaled data for non-distance-based algorithms. We used train_test_split for determining overall accuracy.
- From the results, we chose to hypertune 5 parameters of RandomForestClassifier to give us better results. It indeed, met our expectations.
- We fitted the final model on the whole dataset.
- We created a pickle file of our model to be used for deployment purpose.
- Using Azure Web Services , we deployed our model . This will enable the user to predict LOS of patients easily.

4. Model Evaluation

The final model which was considered is Random Forest.

The Random Forest is an extremely popular machine learning algorithm. Often, with not too much pre-processing, one can throw together a quick and dirty model with no hyperparameter tuning and acheive results that aren't awful. Random Forests, are more than just bagged trees and use a number of interesting techniques to further decrease correlation between trees and reduce overfitting. A quick look at the documentation for scikit-learn's implementation of the RandomForestClassifier shows us the *hyperparameters* we can pass in:

There's over a dozen of them but here I take a closer look what n_estimators, max_depth, min_samples_leaf, and max_features do and why each of them could potentially decrease the error of your model. We have used these 4 hyperparameters alongwith criterion for our final model

i. n_estimators

n_estimators is simply the number of trees. The more uncorrelated trees in our forest, the closer their individual errors get to averaging out**.** However, more is not always better and here are some considerations to keep in mind:

More trees = more computation. Beyond a certain point, the tradeoff may not be worth it.
On a related note, increasing n_estimators gives diminishing returns
No number of uncorrelated trees will get rid of error caused by the bias resulting from unrepresentative training data.

ii. max_depth

max_depth is the how many splits deep you want each tree to go. max_depth = 50, for example, would limit trees to at most 50 splits down any given branch. This has the consequence that our Random Forest can no more fit the training data as closely, and is consequently more stable. It has lower variance, giving our model lower error. Even though severely constraining max_depth could increase the bias of each tree given that they may not be able to capture certain patterns in the data before hitting their limit, we need not worry about this. A suitable choice of n_estimators, coupled with bagging, ensures that the bias of the forest as a whole doesn't increase in the process.

max_depth is a hyperparameter that I typically leave untouched simply because what I really care about is how many observations are at the end of a branch before I forbid the tree from splitting further. This is a better predictor of how overfit the Random Forest is.

iii. min_samples_leaf

min_samples_leaf allows us to do exactly what I described above. min_samples_leaf = 10, for instance, tells each tree to stop splitting if doing so would result in the end node of any resulting branch having less than 10 leaves. To better understand why this is useful, think about how a Decision Tree makes predictions, which I outline again within the context of our LOS prediction problem. Once done training, it predicts the LOS of a patient by passing features of that patient through the tree and finding the end node to which this patient is closest in tree space.

If this is a leaf node, which would be the case if min_samples_leaf = 1 (the default), the forest is predicting the actual LOS in the training set to which this patient happened to be closest. It is almost a certainty that splits toward the ends of the branches aren't capturing actual patterns about the LOS , but just what happened to correspond to a higher or lower LOS in the training data.

iv. max_features

This tells each tree how many features to check when looking for the best split to make. A subtlety here is that passing in max_features = 5 doesn't mean that each tree picks some subset of 5 features to model. Rather, an individual tree chooses a different random sample of 5 features for each split. Like min_samples_leaf, this doesn't allow a tree to fit too closely to the data. More importantly, the trees in the Random Forest are now even less correlated with one another since they weren't even trained on the same data. There has been some practical machine learning research showing that forests of less correlated trees perform better than forests of more correlated trees.

v. Criterion

The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain. Note: this parameter is tree-specific.

5. Comparison to benchmark

At the outset, we had set a overall f1_score of 75 % as the benchmark score.
We managed to get our best model's accuracy score close to the benchmark score.

We could not achieve a significant increase from the benchmark score because we did not use more complex ML models (like CatBoost) or deep learning algorithms (like MLP classifier)

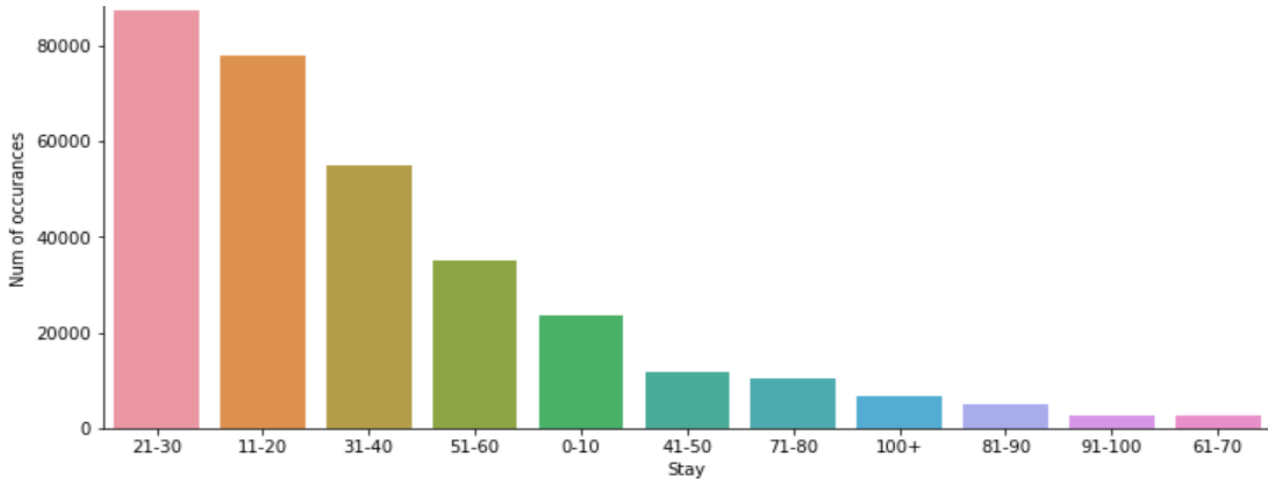
The classification_report of our model is shown below :

	precision	recall	f1-score	support
1	0.72	0.83	0.77	12445
2	0.42	0.10	0.17	1260
3	0.58	0.24	0.34	852
4	0.52	0.07	0.12	3012
5	0.69	0.79	0.74	14904
6	0.61	0.63	0.62	9615
7	0.74	0.73	0.74	12964
accuracy			0.69	55052
macro avg	0.61	0.48	0.50	55052
weighted avg	0.68	0.69	0.67	55052

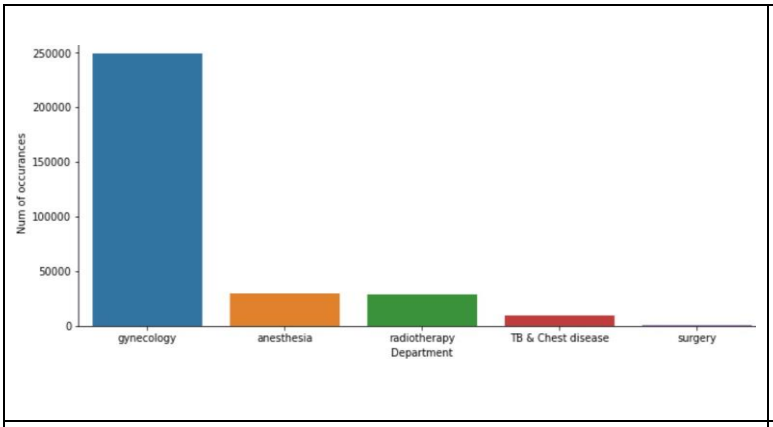
6. Visualizations

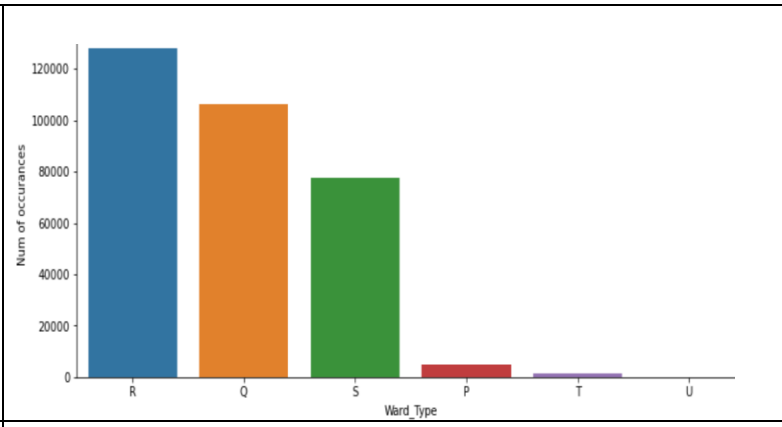
i. Univariate Analysis

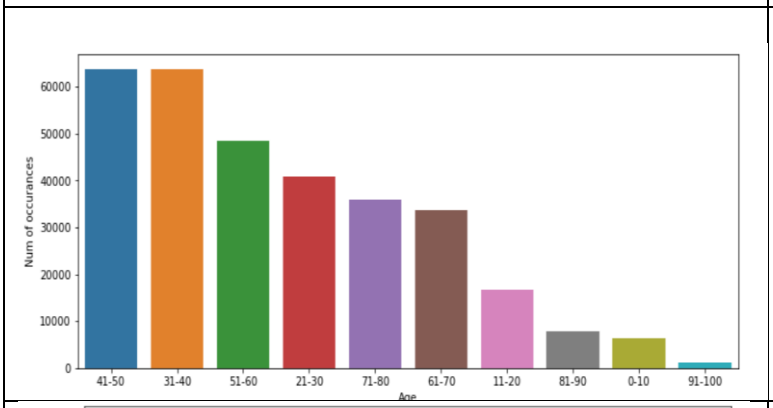
Imbalanced nature of the target column is represented below :

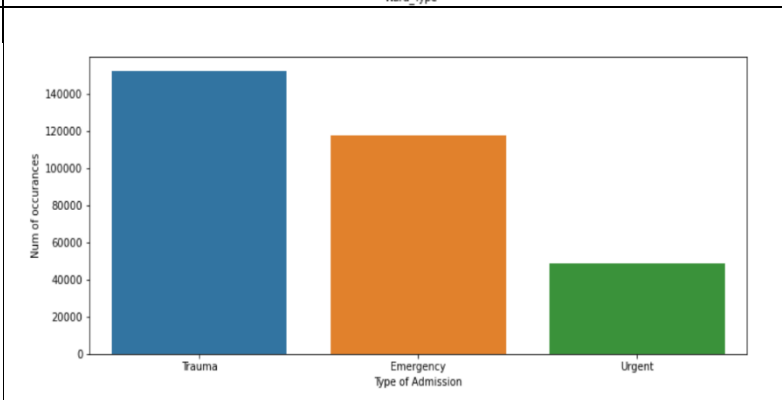


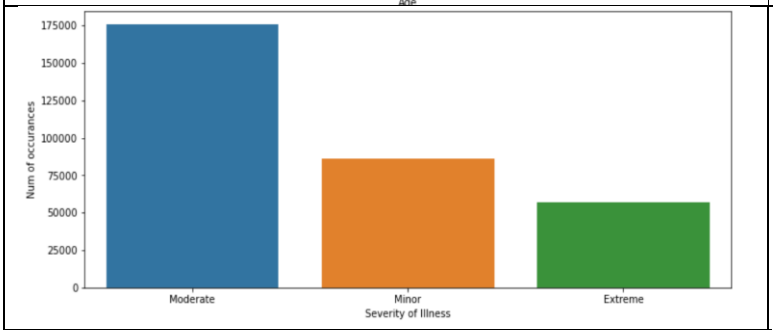
- Almost all the categorical features contained imbalanced data.
- 2 Numerical columns had a highly skewed distribution.

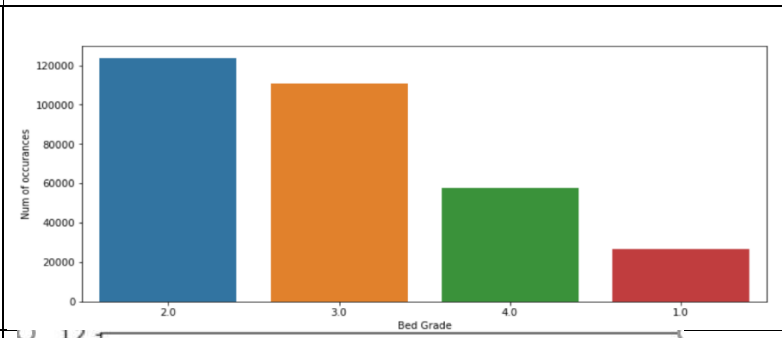


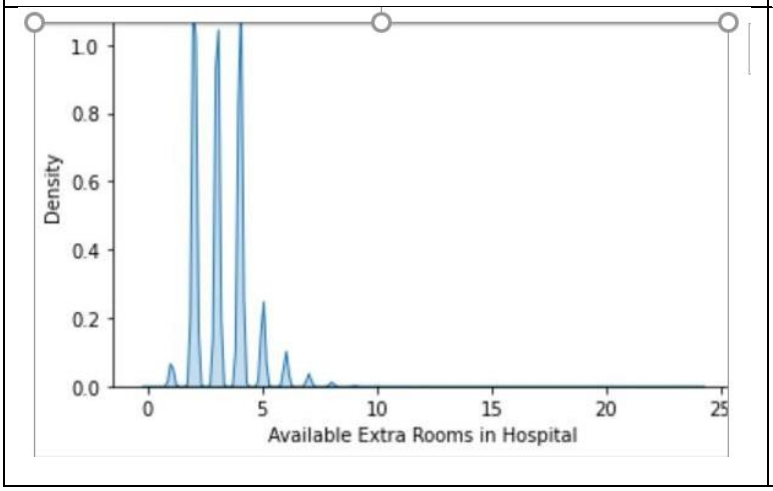


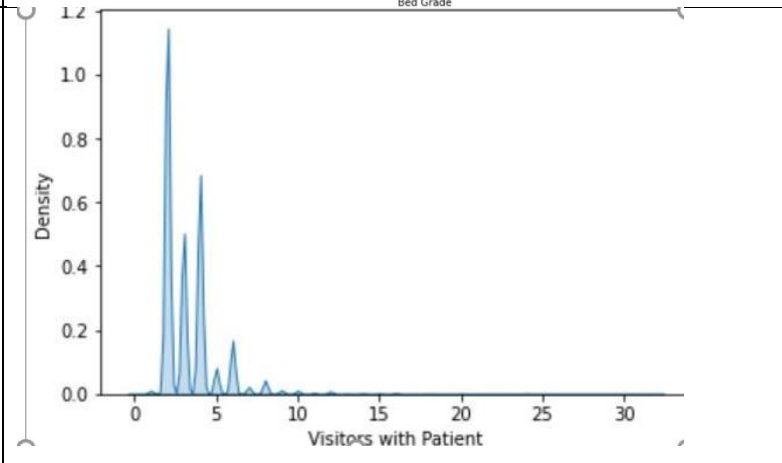






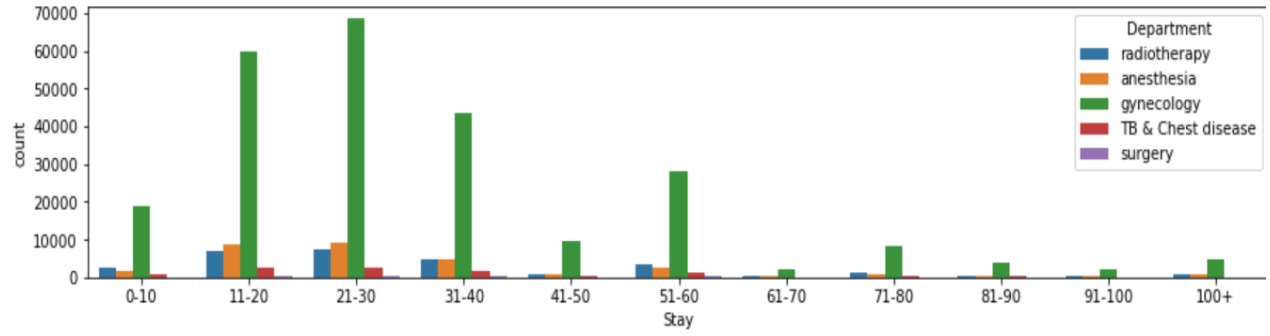


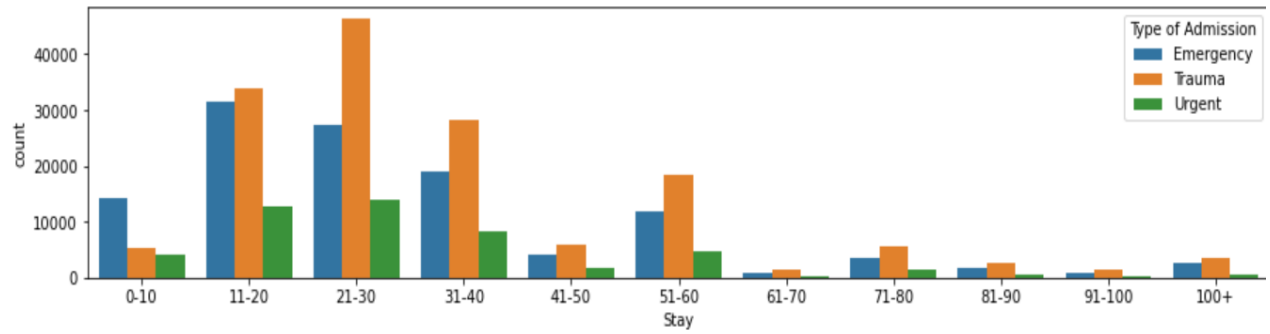


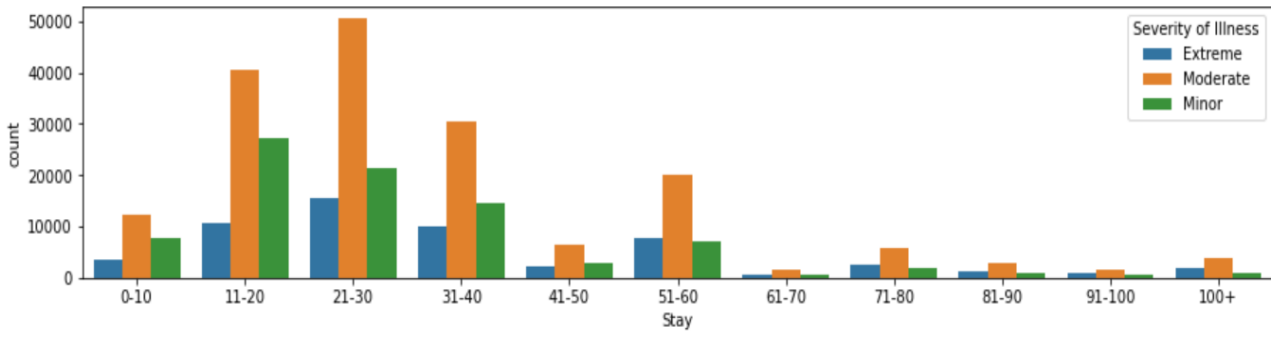


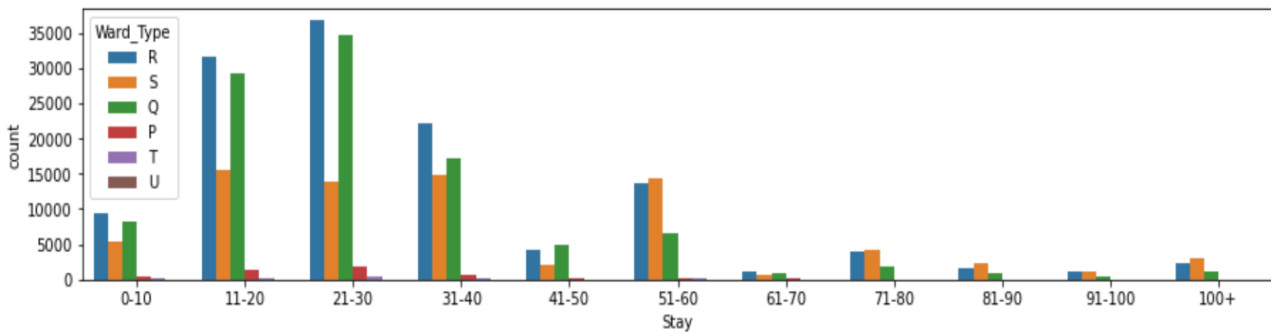
ii. Bivariate Analysis

Almost all the categorical columns were bad quality predictors of LOS.









7. Implications

Our solution will lead to the benefit of both the patient and hospital as it predicts LOS at a faster pace without wasting any of the doctor's time.
For a patient, whose LOS was correctly predicted as less, he does not incur extra expenses.
For a patient, whose LOS was correctly predicted as more, he receives adequate treatment.
For a hospital, it results in better bed management, proper time scheduling & efficient treatment.
It also leads to increased patient satisfaction which in turn leads to increase in hospital's profits.
For these reasons, we would urge the hospital to use our ML –based web app for LOS prediction.

We would recommend the hospital to use our model alongwith consultancy of a domain expert because the f1_score of our model was average i.e. 70 % .

8. Limitations

Our model cannot be relied upon completely because of its average f1_score (70%).
In the real world, our model may lead to incorrect results and as a result may affect the hospital management system and also the patients' expectation
Hence, the model's results have to be consulted with a domain expert.
To build a more accurate model, we can ask for a more balanced data from the data aggregator

9. Closing reflections

Some of the learnings are :

- We dealt with highly imbalanced and biased data alongwith bad quality predictors.
- We deployed our trained model using Microsoft Azure Services to provide a easily available solution to predict LOS of patients.

Some of the things that we would do differently next time are:

- Using more number of estimators in RandomForestClassifier
- Using more complex ML models like CatBoost
- Using Neural Network and Deep Learning models
- Build a sub-model which can predict the admission deposit based on our observations

10. References

- https://datahack.analyticsvidhya.com/contest/janatahack-healthcare-analytics-ii/#ProblemStatement
- https://www.kaggle.com/nehaprabhavalkar/av-healthcare-analytics-ii
- https://towardsdatascience.com/predicting-hospital-length-of-stay-at-time-of-admission-55dfdfe69598