



## GROUP 7 CAPSTONE FINAL PRESENTATION

# **Innovative Approach on pandemic period A Disciplinary Strategy**

**DOMAIN: RISK ANALYSIS**


By:

Ms. Anitha

Mr. Manoj Prabu

Mr. Nawshath

Mr. Sai Charan Ganesh



Mr. Suryakumar  
Ms. Vidhya Priya

## Problem Statement:

- The data from the medical industry has now been enormous and used in various platforms to analyze and predict multiple features. The prediction from those can either be targeted as medical or non-medical predictions.
- Number of days a patient stays in a hospital is a non-medical prediction that helps in deciding the risk of the patient and the availability of beds. But deciding the bed availability involves lot of techniques and factors from the medical administration.
- The reason for this increased demand in bed availability also depends on various factors like Hospital type and severity of patients.
- We finalize that here we are deciding the availability of beds by predicting the stay days.



### **Source of data:**

[https://www.kaggle.com/datasets/arashnic/covid19-hospital-treatment?select=host\\_train.csv](https://www.kaggle.com/datasets/arashnic/covid19-hospital-treatment?select=host_train.csv)

### **Carrying out the Problem statement:**

As we know the problem statement we can carry out supervised classification technique for the prediction. By using the classification techniques we identify the category of new observations(labels) based on the model we built.




## Exploratory Data Analysis:

Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypotheses and to check assumptions with the help of summary statistics and graphical representations. Since numerous features are present in the dataset , certain features need to be dropped to proceed with feature engineering and the missing value treatment which helps to gain more insights.

**Exploratory Data Analysis:**      shape : ( 318438,18 ) INDEPENDENT

VARIABLE:



Variable Name	Variable Description
Case ID	Indicates unique identity number for the case
Hospital	Name of the Hospital
Hospital type	Types of Hospital
Hospital city	The City in which the Hospital is present
Hospital region	Region of the Hospital
Available extra rooms	Number of Extra rooms available in the Hospital
Department	Department overlooking the case   ['radiotherapy' 'anesthesia' 'gynecology' 'TB & Chest disease' 'surgery']
Ward type	Types of Ward
Ward facility	Ward Facility Types
Bed Grade	Condition of Bed in the Ward
Patient ID	Indicates unique identity number for the Patient
City Code	City Code for the patient
Type of admission	Admission Type registered by the Hospital   ['Emergency' 'Trauma' 'Urgent']
Illness severity	Severity of the illness recorded at the time of admission   ['Extreme' 'Moderate' 'Minor']
Patient visitors	Visitors visiting the patient
Age	Age category   ['51-60' '71-80' '31-40' '41-50' '81-90' '61-70' '21-30' '11-20' '0-10' '91-100']
Admission Deposit	Deposit at the Admission Time

DEPENDENT VARIABLE (Labelled Data):



Stay Days	Stay Days by the patient (target)   ['0-10' '41-50' '31-40' '11-20' '51-60' '21-30' '71-80' 'More than 100 Days' '81-90' '61-70' '91-100']
-----------	--

## Data Cleansing:


Data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset.

In our data set we are dropping columns ‘Case Id’, ‘Patient Id’ since we decided that the columns are in-significant in building the model.

## Removing Duplicate Values:

Here we are removing the duplicate values that are present in the dataset which are insignificant. In our dataset we have removed **152** rows.

## Missing Values:



Null values were found in our dataset which turned out to be of minimum percentage and are only in '**Bed grade**' and '**City code patient**' columns. We have decided to drop those rows with null values.

We then carried out the various analysis by visualizing the individual numerical and categorical columns of the dataset.

## **Outlier Treatment:**


## Checking for outliers

```
q1=plan_data.quantile(.25)
q3=plan_data.quantile(.75)
iqr=q3-q1
ul=(q3+(1.5*iqr))
ll=(q1-(1.5*iqr))
df=plan_data[~((plan_data>ul)|(plan_data<ll))]
df.isna().sum()
```

Hospital	0
Hospital_type	4235
Hospital_city	0
Hospital_region	0
Available_Extra_Rooms_in_Hospital	1336
Department	0
Ward_Type	0
Ward_Facility	0
Bed_Grade	0
City_Code_Patient	24764
Type of Admission	0
Illness_Severity	0
Patient_Visitors	9597
Age	0
Admission_Deposit	12125
Stay_Days	0
dtype:	int64

Interpretation:

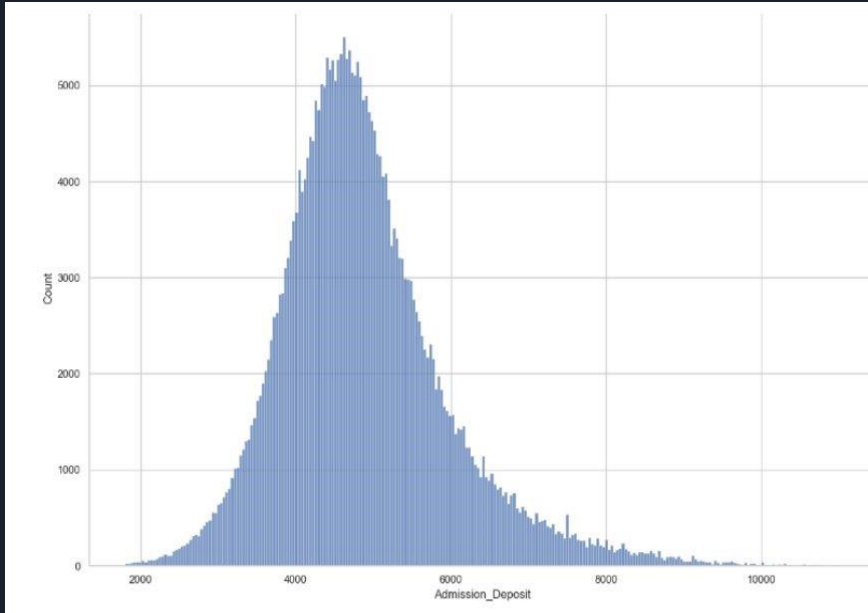




After a deep analysis we have found that there is a presence of extreme outliers in few of the features.

The observations that are beyond the boundaries are dropped.

**Univariate Analysis on Numerical column:**

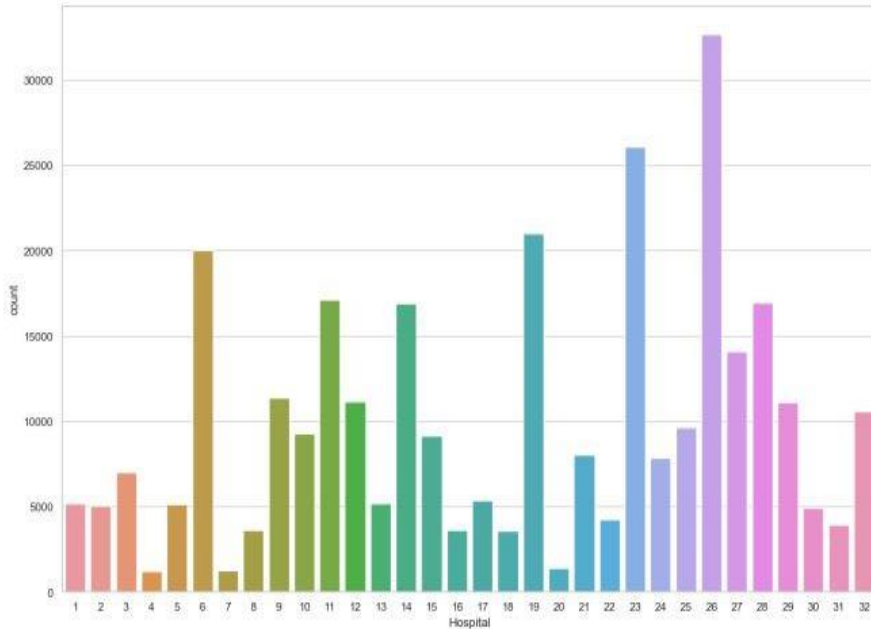


### **Interpretation:**

The amount deposited at the time of admission of a patient shows that the data is normally distributed.

But still data shows slightly right skewness.

### **Univariate Analysis on Categorical column:**



### Interpretation:

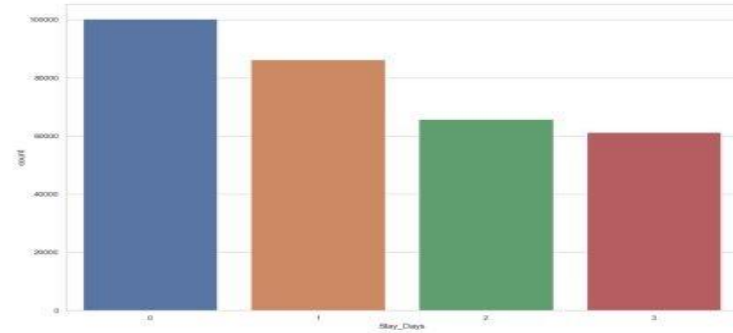
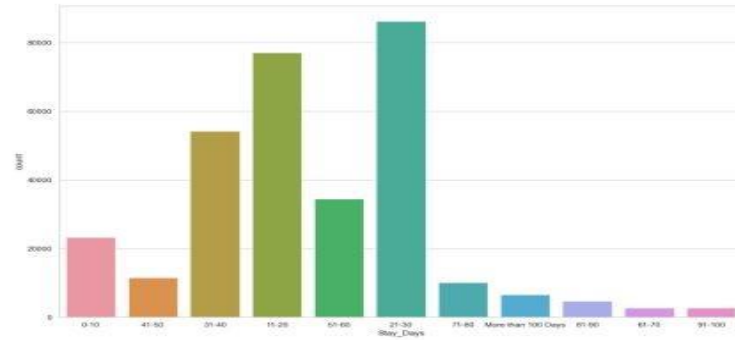
Comparing to other hospitals, hospital 26 may have more number of patients details.

Hospital 6, 19, 23 are may have patients details more than 20000 but lesser than hospital 26.

Hospital 4, 7, 8, 16, 18, 20, 30, 31 are may have patients details lesser than 5000.

**Target variable:**

**Univariate Analysis on**



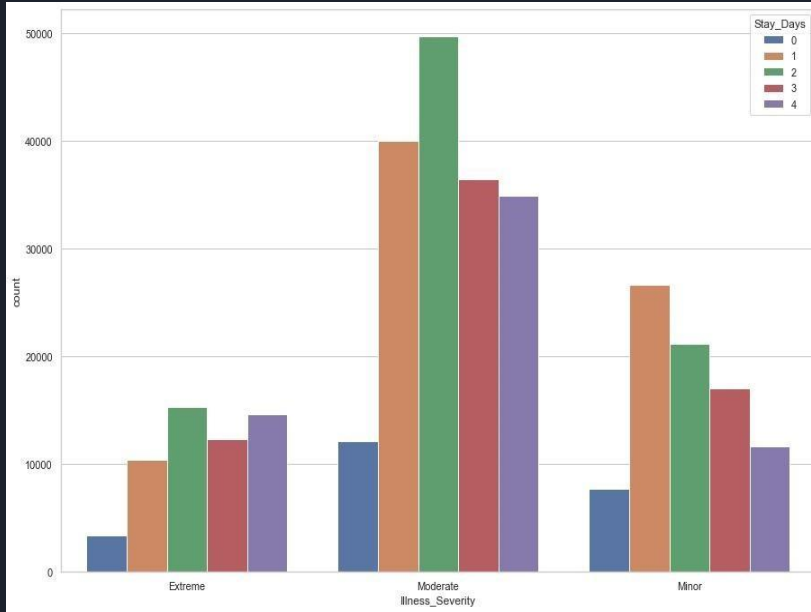
## Interpretation:

By visualizing the overall target features. The data was imbalanced, to reduce the imbalance target feature we are binning it.



## **Bivariate Analysis:**

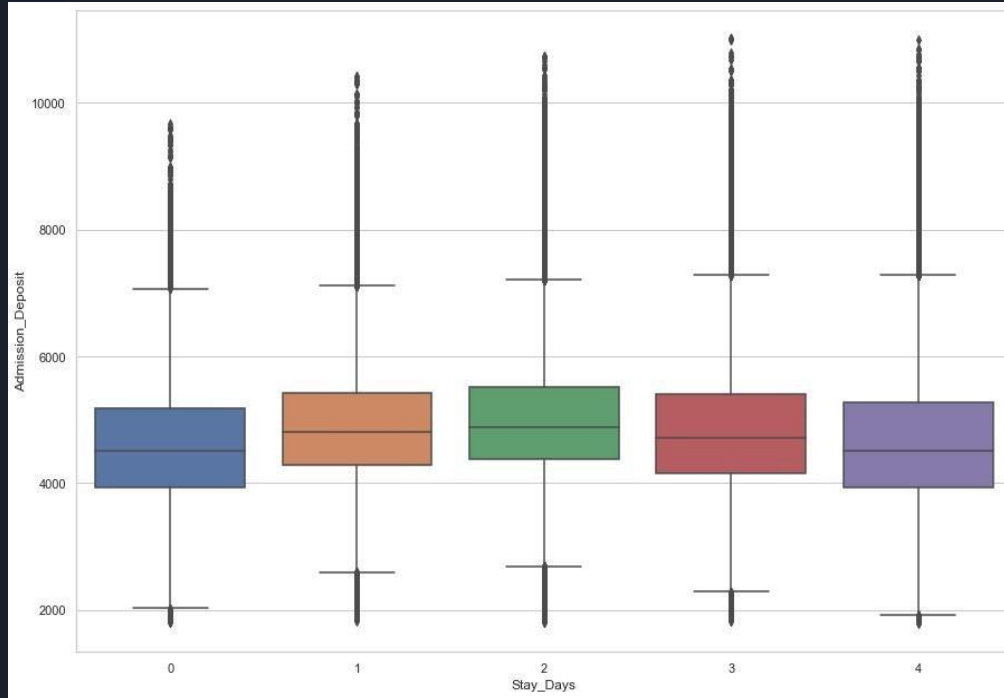
It is an analysis between the independent variables and the dependent variables. In our project we have done bivariate analysis comparing various features with 'stay days' to get insights on how the data is with reference to the dependent feature.



## Interpretation:

- It is found that most of the patient who is staying are of moderate severity patients
- Extreme severity patients who are staying in the hospital are lesser than the minor severity.

Boxplot visualization of Target variable:

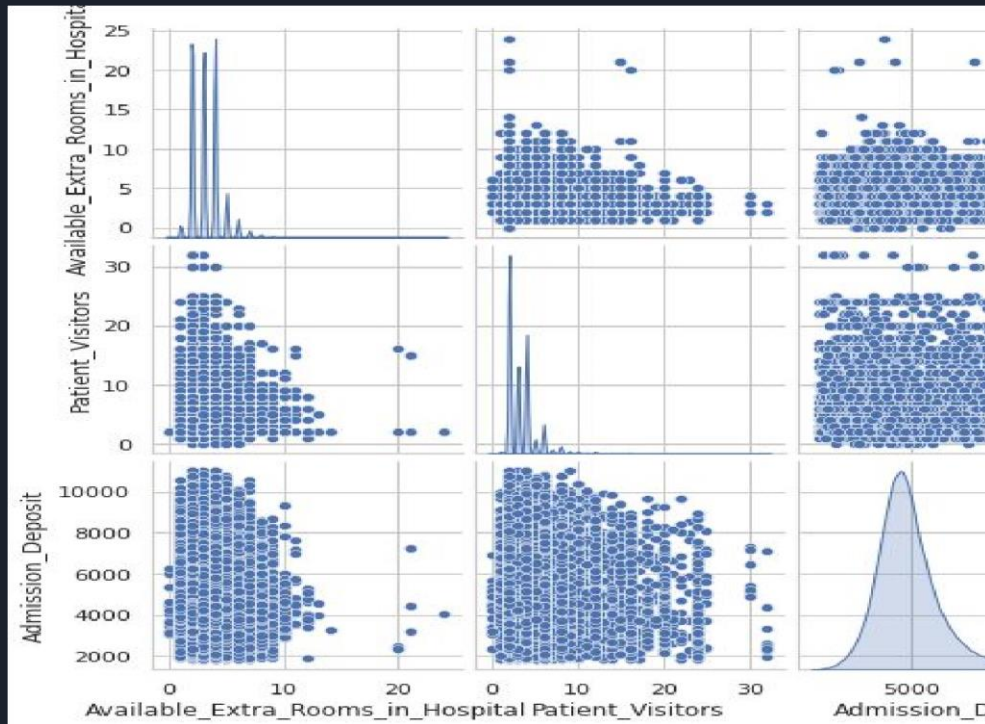


### Interpretation:

- Most of the patients who have stayed in the hospital have deposited an amount more than 7500, which are the outliers.
- There are few patients who have stayed in the hospital but deposited the amount lesser than the usual amount paid by other patients.

### Multivariate Analysis:

Multivariate analysis refers to statistical techniques used to analyze data with multiple variables or factors.



### Interpretation:

The features 'Available extra rooms in hospital, patient visitors, admission deposit' shows skewness to a certain extent (All positively skewed).

The features represented in the scatterplots show no patterns which is interpreted as lesser correlation between the features.





All the scatterplot shows there are significant number of outliers in all the features.

## Feature Engineering:

### ENCODING:

We have encoded target column of the dataset in 3 different ways:

1. Overall target variable has been binned to 4 classes
2. Under-sampling was carried and encoded as it is
3. Under-sampling was carried and target variable binned into 5 classes

Other categorical columns: Illness\_severity and Age are encoded ordinally, rest are encoded using One Hot encoding.

## Feature Encoding

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

df.Age.replace({'0-10':0, '11-20':1, '21-30':2, '31-40':3, '41-50':4, '51-60':5, '61-70':6, '71-80':7, '81-90':8, '91-100':9, '101-110':10}, inplace=True)

df.Stay_Days.replace({'0-10':0, '11-20':1, '21-30':2, '31-40':3, '41-50':4, '51-60':5, '61-70':6, '71-80':7, '81-90':8, '91-100':9, 'More than 100 Days':10}, inplace=True)


pd.options.display.max_columns=None

dummy_encode=pd.get_dummies(df,columns=['Hospital', 'Hospital_city', 'Hospital_region', 'Hospital_type', 'Department', 'Ward_Type', 'Ward_Facility', 'Bed_Grade', 'City_Code_Patient', 'Type of Admission'])
```

## TRANSFORMATION:

Transformation is a process of converting data from one form to another to make it more suitable for analysis or modeling. Some common transformation techniques include scaling, normalization, feature extraction, and dimensionality reduction.

Here we have used **POWER TRANSFORMATION** is done to transform the numerical columns.



```
pt=PowerTransformer()  
for i in num_data.columns:  
    dummy_encode[i]=pt.fit_transform(dummy_encode[[i]])  
dummy_encode
```

## SCALING:

Scaling refers to the process of transforming numerical data into a standardized range or distribution. This is typically done to ensure that features with larger values do not dominate the analysis or model training process.

Here **STANDARDISATION** is done on the numerical columns, this involves subtracting the mean and dividing by the standard deviation of the data, which results in a scaled data set with a mean of zero and a standard deviation of one.




## Scaling the numerical features.

```
scaler=StandardScaler()  
for i in num_data.columns:  
    scaler.fit(dummy_encode[[i]])  
    dummy_encode[i]=scaler.transform(dummy_encode[[i]])  
dummy_encode
```

## Model Building:

### Train test split:

Split arrays or matrices into random train and test subsets so that the model building can be carried out using the train and test data.




The data is split into dependent feature Y (target) and independent features X. The data is then split into training and testing sets in order to avoid data leakage. The default 80:20 split is done.

```
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2,stratify=y,random_state=102)
```

## WHOLE DATA :

Model	Accuracy	Precision	Recall	f1-score
Logistic Regression	0.4950	0.4522	0.4950	0.4517
Decision tree	0.4124	0.4128	0.4124	0.4126
Random Forest	0.4803	0.4707	0.4803	0.4722



Ada Boost	0.4845	0.4620	0.4845	0.4632
Bagging classifier	0.4664	0.4612	0.4664	0.4605
Gradient Booster	0.5149	0.5100	0.5149	0.4753
XGBoost	0.5232	0.5183	0.5232	0.5019

## Undersampling:

- Undersampling is a technique to balance uneven datasets by keeping all of the data in the minority class and decreasing the size of the majority class.
- The goal of undersampling is to balance the classes in the dataset and improve the performance of the machine learning model, especially when the minority class is of greater interest or importance.




Undersampling is done with all the data in the target column and also binning the target column.

## Splitting the dataset with respect to the target variables

```
u1=undersample[undersample['Stay_Days']=='0-10']  
u2=undersample[undersample['Stay_Days']=='11-20']  
u3=undersample[undersample['Stay_Days']=='21-30']  
u4=undersample[undersample['Stay_Days']=='31-40']  
u5=undersample[undersample['Stay_Days']=='41-50']  
u6=undersample[undersample['Stay_Days']=='51-60']  
u7=undersample[undersample['Stay_Days']=='61-70']  
u8=undersample[undersample['Stay_Days']=='71-80']  
u9=undersample[undersample['Stay_Days']=='81-90']  
u10=undersample[undersample['Stay_Days']=='91-100']  
u11=undersample[undersample['Stay_Days']=='More than 100 Days']
```

## Creating Undersampling Data:



```
du1=u1.sample(n=2500,random_state=102)
du2=u2.sample(n=2500,random_state=102)
du3=u3.sample(n=2500,random_state=102)
du4=u4.sample(n=2500,random_state=102)
du5=u5.sample(n=2500,random_state=102)
du6=u6.sample(n=2500,random_state=102)
du7=u7.sample(n=2500,random_state=102)
du8=u8.sample(n=2500,random_state=102)
du9=u9.sample(n=2500,random_state=102)
du10=u10.sample(n=2500,random_state=102)
du11=u11.sample(n=2500,random_state=102)
```

## Concatinating the sample datasets

```
udf=pd.concat([du1,du2,du3,du4,du5,du6,du7,du8,du9,du10,du11])
udf.head()
```

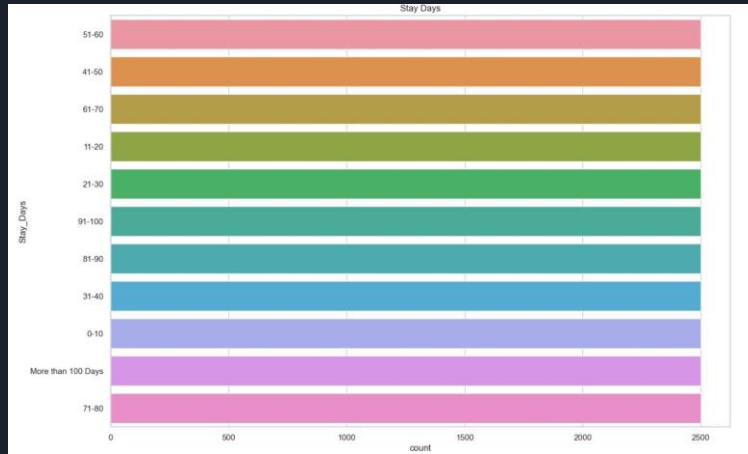
```
#Shuffling the order of the instances
udf=udf.sample(frac=1,random_state=102)
```

```
#Resetting the index
udf.reset_index(inplace=True,drop=True)
```

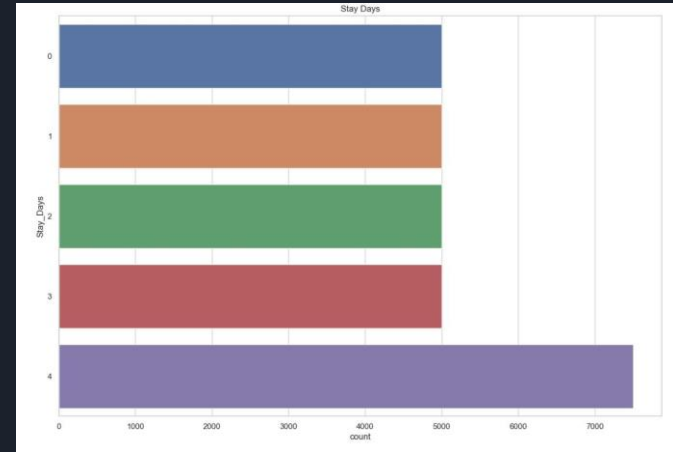
The Target column has 11 categories, from each categories 2500 numbers of sample is taken randomly and combine it into the single dataframe.



## Undersampled Target variables - Visualization :




Undersampled Target Column




Undersampled Target Column Binned

## Model Building Results:




Model_Name	Accuracy score	precision score	recall score	f1 score
Logistic Regression	0.2570	0.2400	0.2570	0.2383
Decision Tree Classifier	0.2132	0.2127	0.21327	0.2128
Random Forest Classifier	0.2718	0.2651	0.2718	0.2665
AdaBoost Classifier	0.2233	0.2120	0.2233	0.2098
Bagging Classifier	0.2622	0.2592	0.2622	0.2573
Gradient Boosting Classifier	0.2937	0.2848	0.2937	0.2778



XG Boost Classifier	0.2933	0.2845	0.2933	0.2851
---------------------	--------	--------	--------	--------

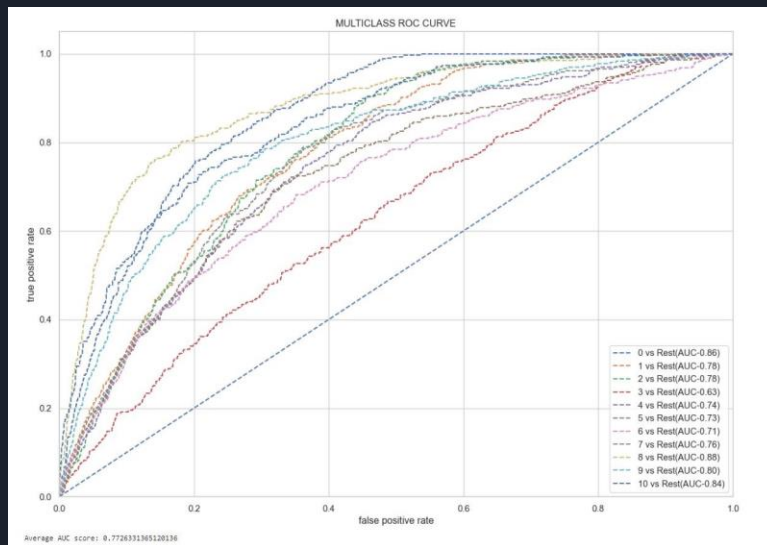
## Model Building Results (Target column Binned):

Model_Name	Accuracy score	precision score	recall score	f1 score
Logistic Regression	0.4162	0.3646	0.4162	0.3649
Decision Tree Classifier	0.3456	0.3434	0.3456	0.3444
Random Forest Classifier	0.4164	0.3902	0.4164	0.3968
AdaBoost Classifier	0.4158	0.3779	0.4158	0.3832



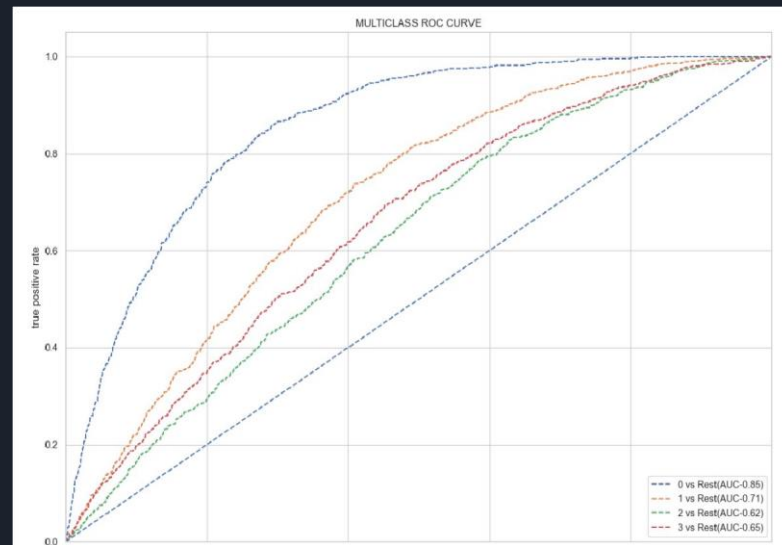
Bagging Classifier	0.3914	0.3796	0.3914	0.3832
Gradient Boosting Classifier	0.4163	0.3946	0.4163	0.3961
XG Boost Classifier	0.4294	0.4006	0.4294	0.4077

**Comparing ROC curves of under-sampled and binned data:**



Average AUC score: 0.7726331365120136

**XG Boost Roc curve**



Average AUC score: 0.7094837207446313

**XG Boost Roc curve(Binned Data)**



# Hyper Parameter tuning

```

In [ ]: params={
        'max_depth':[3,4,5],
        'learning_rate':[0.1,0.5,1.0],
        'n_estimators':[50,200,500]
      }

In [ ]: gs=GridSearchCV(estimator=xgbc, param_grid=params, cv=5, scoring='accuracy')
gs.fit(xtrain, ytrain)


Out[ ]: GridSearchCV(cv=5,
                    estimator=XGBClassifier(base_score=None, booster=None,
                                           callbacks=None, colsample_bylevel=None,
                                           colsample_bynode=None,
                                           colsample_bytree=None,
                                           early_stopping_rounds=None,
                                           enable_categorical=False, eval_metric=None,
                                           feature_types=None, gamma=None,
                                           gpu_id=None, grow_policy=None,
                                           importance_type=None,
                                           interaction_constraints=None,
                                           learning_rate=None, ...
                                           max_cat_threshold=None,
                                           max_cat_to_onehot=None,
                                           max_delta_step=None, max_depth=None,
                                           max_leaves=None, min_child_weight=None,
                                           missing=nan, monotone_constraints=None,
                                           n_estimators=100, n_jobs=None,
                                           num_parallel_tree=None,
                                           objective='multi:softprob', predictor=None, ...),
                    param_grid={'learning_rate': [0.1, 0.5, 1.0],
                                'max_depth': [3, 4, 5],
                                'n_estimators': [50, 200, 500]},
                    scoring='accuracy')

In [ ]: gs.best_estimator_
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.5, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=4, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              n_estimators=50, n_jobs=None, num_parallel_tree=None,
              objective='multi:softprob', predictor=None, ...)

```

Hyperparameter tuning is a method for selecting the best hyperparameters for a machine-learning model by searching over a predefined grid of hyperparameter values using cross-validation. This involves creating a model pipeline, specifying a cross-validation strategy, and using the GridSearchCV class to perform the search over the hyperparameter grid. The goal is to find the optimal set of hyperparameters that yields the best performance on the validation set, which may significantly improve the model's performance and lead to better predictions.

## COMPARATIVE MODELS (XGBoost):




From the overall dataset and undersampling dataset models, the XGBoostClassifier has better performance on the models. To find the best model, we are tuning the overall and undersampling datasets for XGBoostClassifier. After tuning the XGBoostClassifier, we found that the overall dataset XGBoostClassifier gives better performance than the other models.

Model	Accuracy score	Precision score	Recall score	F1 score	AUC score
Overall XGBoost	0.5221	0.5174	0.5221	0.4935	0.7686
Undersampled XGBoost	0.4393	0.4100	0.4393	0.4121	0.7148

Business Suggestion:



- 
- When dealing with multi-class classification problems like our dataset, the cost of false positives and false negatives can be high for all classes.
  - In a medical diagnosis problem, misdiagnosing a disease as present when it is not (a false positive) can lead to unnecessary treatment and potential harm to the patient, while misdiagnosing a disease as absent when it is present (a false negative) can lead to a missed opportunity for treatment and potentially life-threatening consequences,there by making the patients stay for longer period time leading to shortage in beds.
  - Therefore we suggest to shift such patients to other wards or discharge so that we can make bed available for patients with serious needs in the future at the pandemic ward.



**THANK YOU !**