Modeling ICU Occupancy

A Project Report Submitted

In Fulfilment of the Requirements

For the Course of

Internship with **Celebal Technologies**

In

Fourth year – seventh Semester of

Bachelor of Technology

Specialization

In

Artificial Intelligence and Machine Learning

Under

Ms. Poonam Kainthuria

Assistant Professor, Department of Informatics

School of Computer Science

By

500068188	60	Naman Jain
500067512	61	Naman Jajoo
500067534	81	Saurav Verma
500065751	82	Shashank Panthari

UPES

UNIVERSITY WITH A PURPOSE

DEPARTMENT OF INFORMATICS

SCHOOL OF COMPUTER SCIENCE

UNIVERSITY OF PETROLEUM AND ENERGY STUDIES, BIDHOLI, DEHRADUN, UTTRAKHAND, INDIA

June-July 2021

I. ABSTRACT

Predicting the bed occupancy of intensive care unit (ICU) is a dynamic task. The uncertainty associated with critically ill patient and random arrival patient, severity of new patient leads to the bed capacity problem, and we need to take productive measure to improve the condition. In this project, we work toward creating predictive model using regression model and improving the prediction with minimizing loss.

II. Introduction

In the current era there is increase in monitoring patient's details in digital form with the improvement in the technology we can further improve the hospital facilities by predicting ICU beds availability. Predicting free beds availability can be difficult task because of the unexpected arrival of the patients and dynamic reasons. The development of an automated model will assist physician in these matter and there will be a clear understanding of the beds availability.

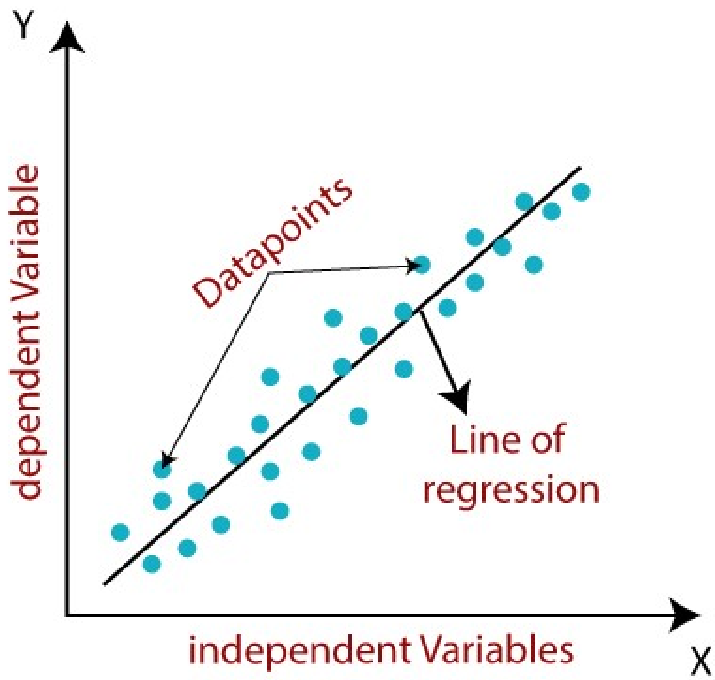
In this section, we stimulate a mathematical mode to predict the ICU bed occupancy.

Even though the model can be applied to any unit, but we believe that it is reasonable to focus more on ICU beds availability as ICU is more critical since it is expensive and its important block of hospitals.

This section is divided into 2 sections in which first we introduce the concept of linear regression on which the model is based and second is to retrieve the data and process the hospitals records then formulating linear regression model to predict ICU occupancies.

Linear regression

Linear regression is a ML algorithm user for supervised learning. Linear regression perform task to predict a dependent variable (Target) based on the given independent variable. So this regression technique is find out a linear relation between dependent and independent variable



Hypothesis function for Linear Regression :

$$y = \theta_1 + \theta_2 . x$$

How to update θ_1 and θ_2 values to get the best fit line ?

$$minimize \frac{1}{n} \sum_{i=1}^{n} (pred_i - y_i)^2$$

Cost Function (J):

$$J = \frac{1}{n} \sum_{i=1}^{n} (pred_i - y_i)^2$$

III. Problem Description

In the study we address scheduling problem of beds, which hospitals faced during unavailability of beds However, in real-life hospitals, not all patients are of the same medical specialty and not all patients of the same medical specialty take the same path. Hence, we deal with a wide variety of different patient paths and different LOS in each unit.

Therefore, this in particular is a mathematical model for simulating daily bed occupancy in an intensive care unit.

IV. Methodology

In this section we will discuss how we acquire data set and how is data processed. Next, we proceed by introducing algorithm we used in our work.

Data Extraction

The data concern all adults' patients and we have nearly 4000 entries which comprises data from hospitals from multiple countries like Los Angeles, Contra Costa, Alameda, Fresno, Imperial, Inyo and more these records are from year 2005 to 2015.

The data comprises of following features like, year, COUNTY, OSHPD_ID, Facility Name, Licensed Bed Classification, License Bed Designation, Licensed Bed Day, Discharges, Census Day, Intra Hospital Transfer from Critical Care, available beds, County labels, Facility Label, Beds Required

Data pre-processing for extraction of features and labels

1) We will look for null values and we will manage the null values either by dropping complete row or by filling null values with mean as required.
2) Now we will remove columns, which have no effect on prediction and will remove those columns.
3) We will identify target class and visualization is performed as required.
4) Further, with this, we will be having a trend of patients and according to that after splitting data, we will further move to the implementation part of model.
5) At last, we will predict the availability of model with the help of our stimulation model.

V. Implementation

1. Loading important libraries.

```python
# for algebra
import numpy as np
# for data processing
import pandas as pd
# for visualization
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
# for preprocessing
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
```

2. Loading and understanding data.

```python
data = pd.read_csv('beds.csv')
data.head()
```

	Year	COUNTY	OSHPD_ID	Facility Name	Licensed Bed Classification	License Bed Designation	Licensed Bed Day	Discharges	Census Day	Intra Hospital Transfer from Critical Care
0	2005.0	Alameda	106010735.0	ALAMEDA HOSPITAL	General Acute Care	Medical/Surgical Acute (includes GYN/DOU)	30660.0	2835.0	11297.0	0
1	2005.0	Alameda	106010735.0	ALAMEDA HOSPITAL	General Acute Care	Pediatric Acute	0.0	0.0	0.0	0
2	2005.0	Alameda	106010735.0	ALAMEDA HOSPITAL	General Acute Care	Intensive Care	2920.0	170.0	1369.0	326
3	2005.0	Alameda	106010735.0	ALAMEDA HOSPITAL	General Acute Care	Coronary Care	2920.0	0.0	0.0	0
4	2005.0	Alameda	106010735.0	ALAMEDA HOSPITAL	General Acute Care	Acute Respiratory Care	0.0	0.0	0.0	0

```python
data.shape
```

```
(77085, 10)
```

```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 77085 entries, 0 to 77084
Data columns (total 10 columns):
Year                                         77084 non-null float64
COUNTY                                       77084 non-null object
OSHPD_ID                                     77084 non-null float64
Facility Name                                77084 non-null object
Licensed Bed Classification                  77084 non-null object
License Bed Designation                      49554 non-null object
Licensed Bed Day                             77084 non-null float64
Discharges                                   77084 non-null float64
Census Day                                   77084 non-null float64
Intra Hospital Transfer from Critical Care   77085 non-null int64
dtypes: float64(5), int64(1), object(4)
memory usage: 5.9+ MB
```

3. Working on null and duplicate values.

```python
data.isnull().sum()
```

```
Year                                             1
COUNTY                                           1
OSHPD_ID                                         1
Facility Name                                    1
Licensed Bed Classification                      1
License Bed Designation                      27531
Licensed Bed Day                                 1
Discharges                                       1
Census Day                                       1
Intra Hospital Transfer from Critical Care       0
dtype: int64
```

So we have a lot of null values in one category and one null value in the other ones. Let's get rid of these

```python
data = data.dropna()
```

Check once again

```python
data.isnull().sum()
```

```
Year                                          0
COUNTY                                        0
OSHPD_ID                                      0
Facility Name                                 0
Licensed Bed Classification                   0
License Bed Designation                       0
Licensed Bed Day                              0
Discharges                                    0
Census Day                                    0
Intra Hospital Transfer from Critical Care    0
dtype: int64
```

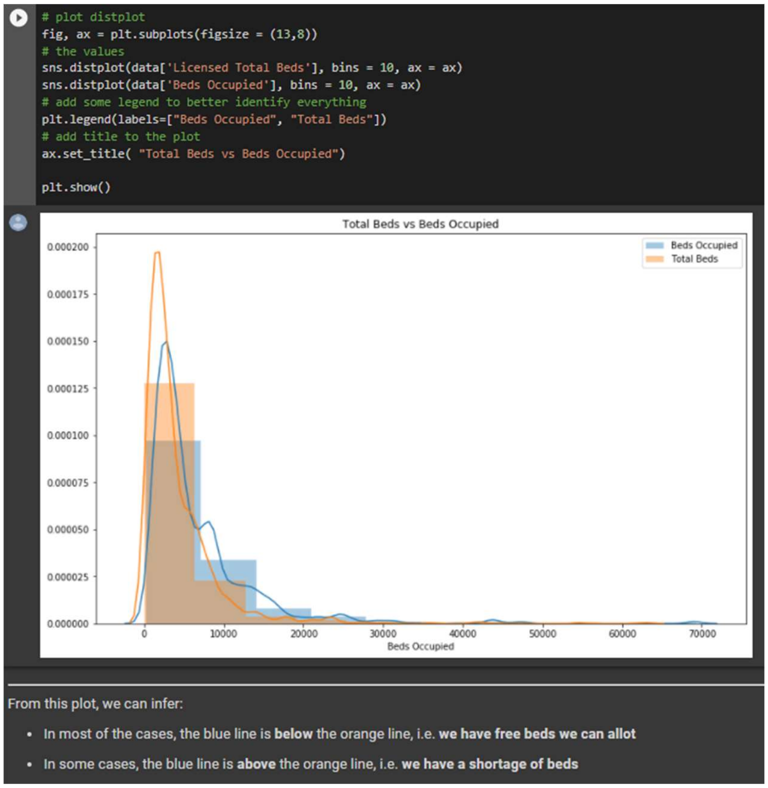
We are good,

We can also delete the duplicate values as that will:

- make data heavier that it needs to be
- lower the model efficiency and speed

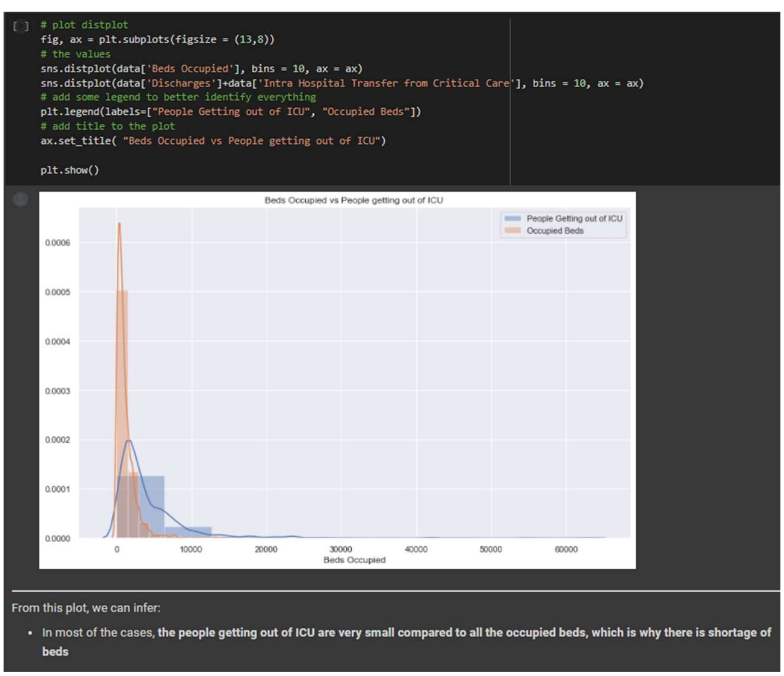
```python
data.drop_duplicates(inplace=True)
```

4. Visualisation.



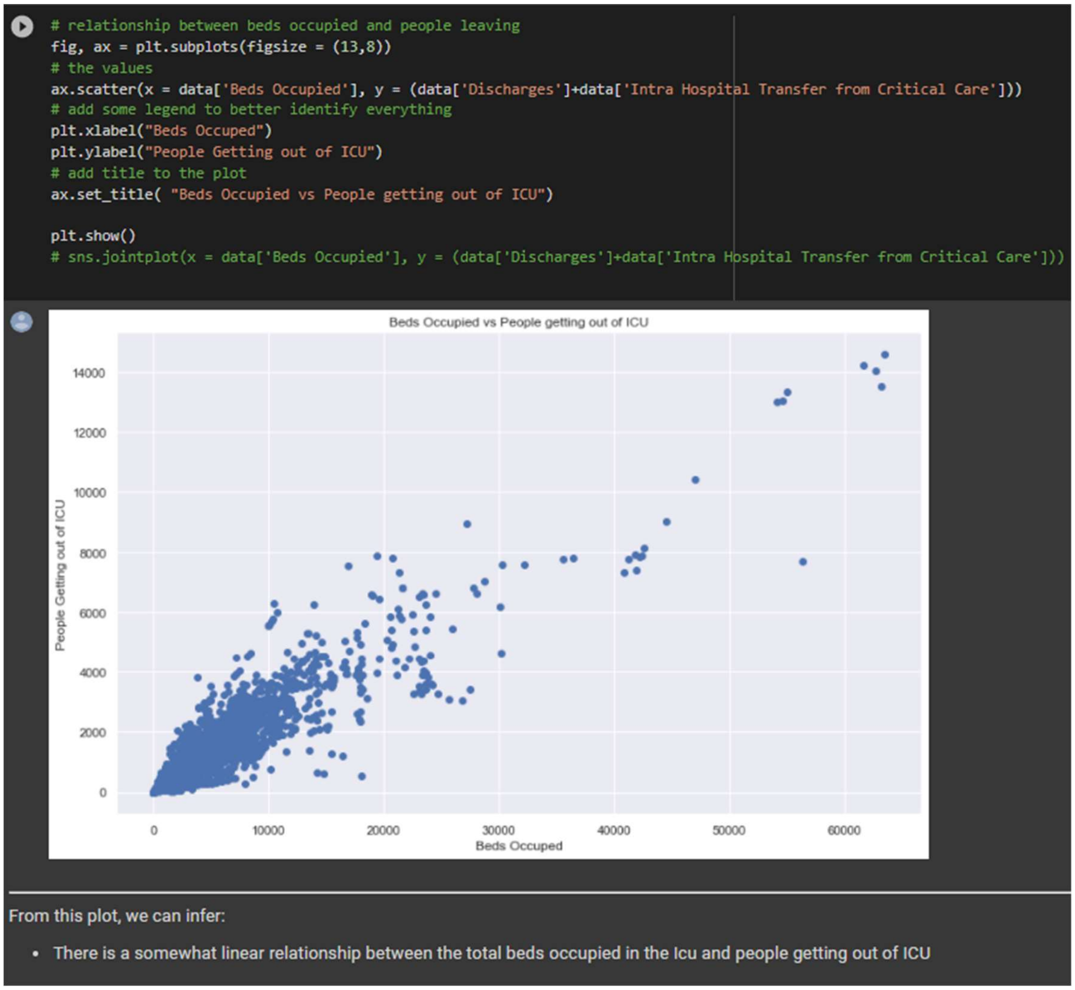
From this plot, we can infer:

- In most of the cases, the blue line is **below** the orange line, i.e. **we have free beds we can allot**
- In some cases, the blue line is **above** the orange line, i.e. **we have a shortage of beds**



From this plot, we can infer:

- In most of the cases, **the people getting out of ICU are very small compared to all the occupied beds, which is why there is shortage of beds**



From this plot, we can infer:

- There is a somewhat linear relationship between the total beds occupied in the ICU and people getting out of ICU

5. Making data training ready.

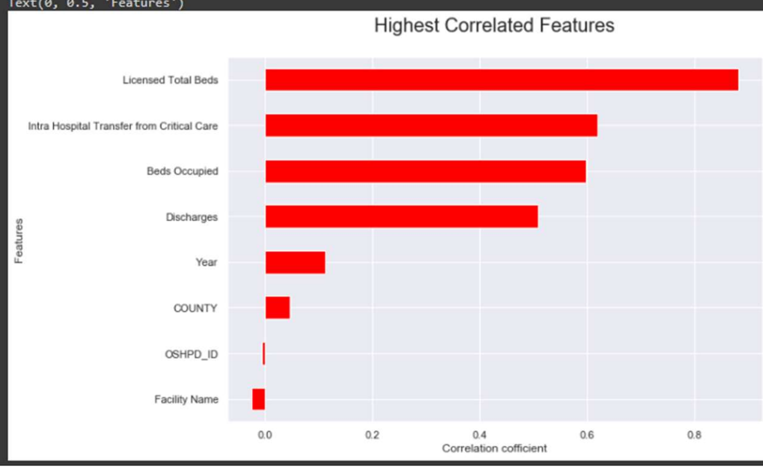
- **Step-1**: Get the object columns encoded so we can find out the correlation between them. Correlation is only calculated between continuous variables and not object type.
- **Step-2**: Get rid of excess/useless columns that do not help, and pick the best to start training. For this we can use correlation of these columns with the target variable to fish out the good attributes

```
[ ] correlation = data.corr()
    print(correlation["Available Beds"].sort_values(ascending = False))

    Available Beds                               1.000000
    Licensed Total Beds                          0.880827
    Intra Hospital Transfer from Critical Care   0.619776
    Beds Occupied                                0.597511
    Discharges                                   0.508840
    Year                                         0.112516
    COUNTY                                       0.046273
    OSHPD_ID                                    -0.005859
    Facility Name                               -0.024325
    Name: Available Beds, dtype: float64
```

```
[ ] plt.figure(figsize=(10,7))
    correlation["Available Beds"].apply(lambda x: (x)).sort_values(ascending=False).iloc[1:9][::-1].plot(kind='barh',color='red')
    plt.title('Highest Correlated Features', size=20,pad=26)
    plt.xlabel('Correlation cofficient')
    plt.ylabel('Features')

    Text(0, 0.5, 'Features')
```



6. Implementing Linear Regression.

Linear Regression

```
# import
from sklearn.linear_model import LinearRegression
```

```
# load the model with normalization
model =  LinearRegression(normalize=True)
# fit the training data
model.fit(x_train,y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=True)
```

```
y_pred_test = model.predict(x_test)
y_pred_test
```

```
array([ 425., 3349., 2095., ..., 2614., 1464., 6682.])
```

VI. Conclusion

We have predicted the number of Intensive care units present in the hospital using linear regression model. There by using simple mathematical operations we have estimated the number of beds available at the end of each day and trained our model based on that data for prediction of the number of beds that will be available in future.

VII. References

- Barado, Julio, et al. “A Mathematical Model for Simulating Daily Bed Occupancy in an Intensive Care Unit.” Critical Care Medicine, vol. 40, no. 4, Apr. 2012, pp. 1098–104. PubMed, doi:10.1097/CCM.0b013e3182374828.

- Schiele, Julian, et al. “Predicting Intensive Care Unit Bed Occupancy for Integrated Operating Room Scheduling via Neural Networks.” Naval Research Logistics (NRL), vol. 68, no. 1, 2021, pp. 65–88. Wiley Online Library, doi:10.1002/nav.21929.

- McCabe, Ruth, et al. “Modelling Intensive Care Unit Capacity under Different Epidemiological Scenarios of the COVID-19 Pandemic in Three Western European Countries.” International Journal of Epidemiology, vol. 50, no. 3, June 2021, pp. 753–67. Silverchair, doi:10.1093/ije/dyab034.

- Ruyssinck, Joeri, et al. “Random Survival Forests for Predicting the Bed Occupancy in the Intensive Care Unit.” Computational and Mathematical Methods in Medicine, vol. 2016, Oct. 2016, p. e7087053. www.hindawi.com, doi:10.1155/2016/7087053.

- Baas, Stef, et al. “Real-Time Forecasting of COVID-19 Bed Occupancy in Wards and Intensive Care Units.” Health Care Management Science, vol. 24, no. 2, June 2021, pp. 402–19. Springer Link, doi: 10.1007/s10729-021-09553-5.