Supervised Learning / Multi Class Classification

1-Problem Statement: Janatahack: Healthcare Analytics II

Recent Covid-19 Pandemic has raised alarms over one of the most overlooked area to focus: Healthcare Management. While healthcare management has various use cases for using data science, patient length of stay is one critical parameter to observe and predict if one wants to improve the efficiency of the healthcare management in a hospital.

This parameter helps hospitals to identify patients of high LOS risk (patients who will stay longer) at the time of admission. Once identified, patients with high LOS risk can have their treatment plan optimized to miminize LOS and lower the chance of staff/visitor infection. Also, prior knowledge of LOS can aid in logistics such as room and bed allocation planning.

Suppose you have been hired as Data Scientist of HealthMan – a not for profit organization dedicated to manage the functioning of Hospitals in a professional and optimal manner. The task is to accurately predict the Length of Stay for each patient on case by case basis so that the Hospitals can use this information for optimal resource allocation and better functioning. The length of stay is divided into 11 different classes ranging from 0-10 days to more than 100 days

2-Data Description

Data columns (total 18 columns), rows = 318438.

Column	Description
case_id	Case_ID registered in Hospital (Dytpe int64).
Hospital_code	Unique code for the Hospital (Dytpe int64).
Hospital_type_code	Unique code for the type of Hospital (Dytpe object). ['a'- 'b'- 'c'- 'e'- 'd'- 'f'- 'g']
City_Code_Hospital	City Code of the Hospital (Dytpe int64).
Hospital_region_code	Region Code of the Hospital (Dytpe object). ['X'- 'Y'- 'Z']
Available Extra Rooms in Hospital	Number of Extra rooms available in the Hospital (Dytpe int64).
Department	Department overlooking the case (Dytpe object). ['anesthesia'- 'gynecology'- 'radiotherapy'- 'surgery']
Ward_Type	Code for the Ward type (Dytpe object). ['R'- 'Q'- 'S'- 'P'- 'T'-'U']

Ward_Facility_Code	Code for the Ward Facility (Dytpe object). ['F'- 'E'- 'D'- 'C'- 'B'- 'A']
Bed Grade	Condition of Bed in the Ward (Dytpe float64).
patientid	Unique Patient Id (Dytpe int64).
City_Code_Patient	City Code for the patient (Dytpe float64).
Type of Admission	Admission Type registered by the Hospital (Dytpe object).
Severity of Illness	Severity of the illness recorded at the time of admission (Dytpe object).
Visitors with Patient	Number of Visitors with the patient (Dytpe int64).
Age	Age of the patient (Dytpe object). ['0-10', '11-20', '21-30', '31-40', '41-50', '51-60', '61-70', '71-80', '81-90', '91-100']
Admission_Deposit	Deposit at the Admission Time (Dytpe float64).
Stay	Stay Days by the patient (Dytpe object). ['0-10', '11-20', '21-30', '31-40', '41-50', '51-60', '61-70', '71-80', '81-90', '91-100', 'More than 100 Days'], 11classes.

3-Data Visualization

Data visualization helps us to understand the data by see how the data looks like and what kind of correlation is held by the attributes of data and then determine the features correspond to the output.

pairs plot is the most effective tools (also called a scatterplot matrix). A pairs plot allows us to see both distribution of single variables and relationships between two variables.

to used pairs plot tool, download seaborn library.

Pairs plot can be generated by two ways, First by using scatter_matrix() function on Pandas DataFrame and plotted with the help of pyplot, Second, by using pairpolt() function on Seaborn.

For example

plot Scatterplot matrix for the Healthcare Analytics dataset.

```
# Seaborn visualization library

import seaborn as sns

# Create the default pairplot
```

```
sns.pairplot(df)
```

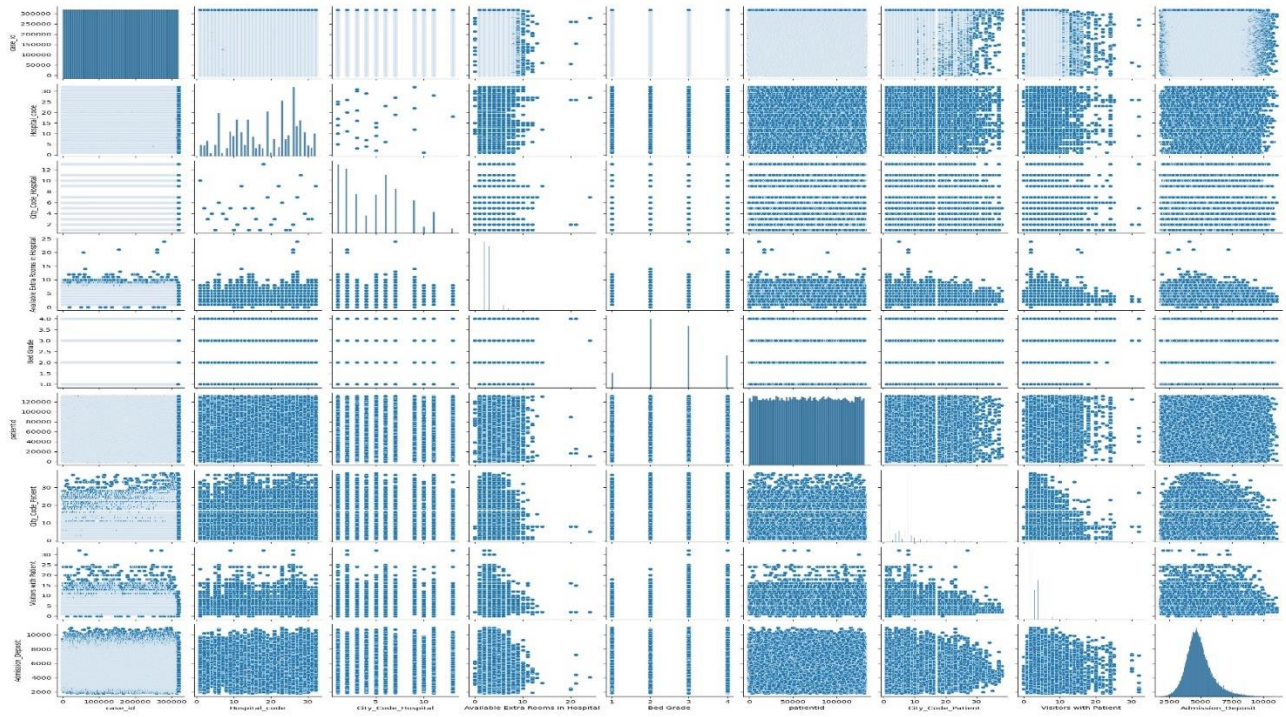
Or

```
From matplotlib import pyplot

From pandas.tools.plotting import scatter_matrix

scatter_matrix(df)

pyplot.show()
```



From figure

distribution of Admission-Deposit feature is close to normal distribution.

Bed Grad, Visitors with Patient. The number of visitors was greater in the case of bed grade= 3 or 4 compared to bed grade= 1 or 2.

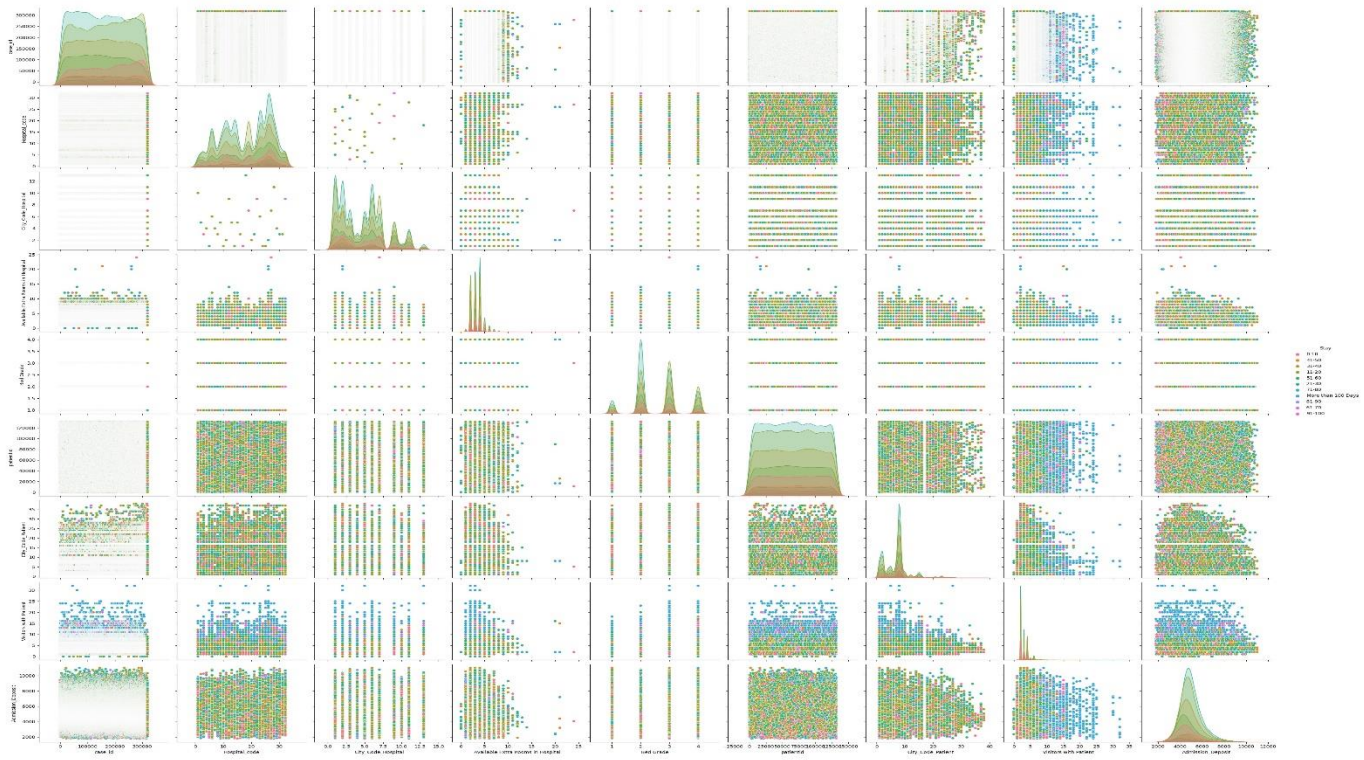
Bed Grad, Available Extra Rooms in Hospital. The number of Available Extra Rooms in Hospital was greater in the case of bed grade= 2 & 4 compared to bed grade= 1 or 2.

most features contain outliers.

```
# Seaborn visualization library

import seaborn as sns
```

```
sns.pairplot(df, hue='Stay', height=3, aspect=1.3)
```

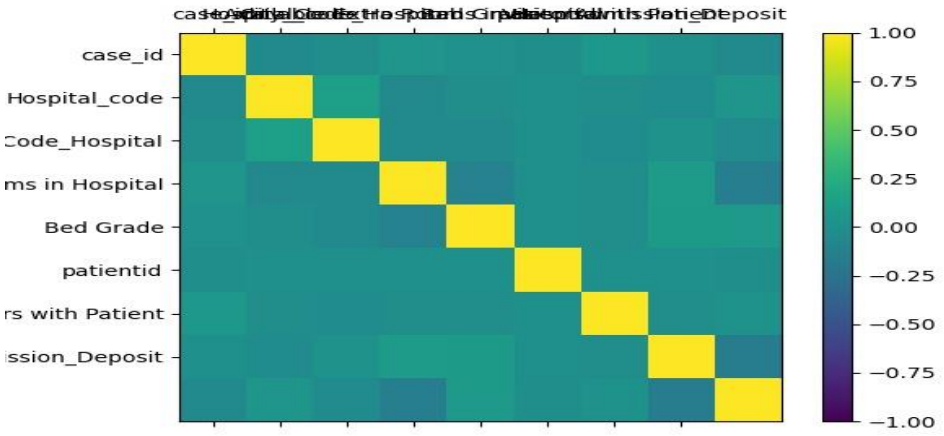


Correlation Matrix Plot uses to show which variable is having a high or low correlation in respect to another variable.

```
sns.heatmap(df, cmap='viridis', linecolor='k', linewidths=2, annot=True)
```

Or

```
names = ['case_id', 'Hospital_code', 'City_Code_Hospital',
'Available Extra Rooms in Hospital','Bed Grade', 'patientid',
'Visitors with Patient', 'Admission_Deposit']
correlations = df.corr()
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(correlations, vmin=-1, vmax=1)
fig.colorbar(cax)
ticks = np.arange(0,8,1)
ax.set_xticks(ticks)
ax.set_yticks(ticks)
ax.set_xticklabels(names)
ax.set_yticklabels(names)
plt.savefig('Correlation_Matrix .jpg') #to save figure
plt.show()
```



From `Correlation_Matrix` figure:

For read.

https://towardsdatascience.com/visualizing-data-with-pair-plots-in-python-f228cf529166

4-Data Preprocessing

For doing data preprocessing in Healthcare Analytics problem. I used feature transformations**.**

Columns- used [hospital_code, city_code_hospital, Available extra rooms in hospital, Department, Ward Type, Bed Grade, Type of Admission, Severity of Illness, Visitors with Patient, Age, Admission Deposit, Stay].

Columns_drop ['case_id', 'Hospital_type_code', 'Hospital_region_code', 'Ward_Facility_Code', 'patientid', 'City_Code_Patient'].

Feature Transformations

- ✓ Data Cleaning
- ✓ Work with Missing Data
- ✓ Work with Categorical Data
- ✓ Detect and Handle Outliers
- ✓ Deal with Imbalanced Classes
- ✓ Feature Scaling

Data Cleaning:

Data cleaning or cleansing is the process of detecting and correcting (or removing) corrupt or inaccurate data values in dataset.

For cleaning Healthcare Analytics dataset, frist when load data and read by Pandas DataFram, convert values which equal ['N/A', 'no', '?'] to NaN value. Second convert two columns ['Bed Grade', 'Admission_Deposit'] from float64 to int type.

Healthcare Analytics dataset not contain NaN values to handle.

```
df = pd.read_csv('train.csv', sep=',', na_values=['N/A', 'no',
'?'])
## Converting float64 to int type
df['Bed Grade'] = df['Bed Grade'].astype(np.int64)
df['Admission_Deposit'] = df['Admission_Deposit'].astype(np.int64)
```

Work with Missing Data:

To detect and handle missing data in Healthcare Analytics dataset [rows = 318437, columns= 18].

to **check missing values** in Pandas DataFrame, we use a function isnull() and notnull() . Both function help in **checking** whether a **value** is NaN or not.

```
df.isnull().sum()
```

Columns missing data ['City_Code_Patient', 'Bed Grade'], number of missing data in 'City_Code_Patient' column = 4532, number of missing data in 'Bed Grade' = 113.

missing data is less, so drop rows which contain missing data.

```
## Drop Missing Data
df = df.dropna(axis=0)  # drop rows from a data set containing missing values
```

Work with Categorical Data:

Work with Categorical Data, columns [Department, Ward_Type, Type of Admission, Severity of Illness, Age, Stay]

```
df = pd.get_dummies(df, columns=['Department', 'Ward_Type', 'Type of Admission',
'Severity of Illness', 'Age'], drop_first=True)
print(df.columns)
df.info()                    # (total 29 columns), 313793 entries (rows)
le = preprocessing.LabelEncoder()       # to convert Y from categorical to label
encoder
df['Stay'] = le.fit_transform(df['Stay'])
df['Stay'] = df['Stay'].astype(int)
```

Detect and Handle Outliers:

To check outliers, columns_used = ['Hospital_code','City_Code_Hospital', 'Available Extra Rooms in Hospital', 'Bed Grade', 'Visitors with Patient', 'Admission_Deposit']

```
for col in columns:     # to show outliers
    sns.boxplot(x=col, data=df)
    sns.stripplot(x=col, data=df, color="#474646")
    plt.show()
```

```
from datasist.structdata import detect_outliers
outliers_indices = detect_outliers(df, 0, columns)
print(len(outliers_indices))
# handle outliers
df.drop(outliers_indices, inplace=True)
```

Deal with Imbalanced Classes:

To check Imbalanced classes, column='Stay'.

```
df['Stay'].value_counts()
```

Class '0' = 22516, Class '1' = 74138, Class '2' = 82496, Class '3' = 50933, Class '4' = 10842, Class '5' = 31069, Class '6' = 2380, Class '7' = 8343, Class '8' = 3573, Class '9' = 2111, Class '10' = 3215.

```
from sklearn.model_selection import train_test_split

x = df.drop('Stay', axis=1)

y = df['Stay']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2,
random_state=22)

from imblearn.over_sampling import SMOTE

smote = SMOTE(random_state=22)

# make smote in only training set

X_train, Y_train = smote.fit_sample(x_train, y_train)
```

Feature Scaling:

```
from sklearn.preprocessing import RobustScaler, MinMaxScaler, StandardScaler
scaling = StandardScaler()
X_train = scaling.fit_transform(X_train)
X_test = scaling.transform(x_test)
```

5-Train models

The machine learns patterns from data in such a way that the learned representation successfully maps the original dimension to the suggested class without any interference from a human.

Machine learning Algorithms for Classification Problem.

classification tasks have discrete categories, unlike regressions tasks.

- ✓ Logistic Regression
- ✓ K Nearest Neighbors (KNN)
- ✓ Support Vector machine (SVC)
- ✓ Decision Trees
- ✓ Random Forest
- ✓ XGBoost

Logistic Regression:

Logistic regression is a ML algorithm which is used for the classification problems, that uses a more complex cost function, this cost function can be defined as the 'Sigmoid function' or also known as the 'logistic function' instead of a linear function.

LR works best on binary classification problems, although it can be applied to do multi-class classification problems through the one-vs-rest scheme in which for each class a binary classification problem of data belonging or not to that class is done, or changing the loss function to cross- entropy loss.

Logistic regression can be divided into following types: Binary or Binomial, Multinomial and Ordinal.

In this task, I build a logistic regression model for doing a multi-class 'Multinomial' classification.

```
From   sklearn.linear_model  import LogisticRegression
Model = LogisticRegression(solver='lbfgs')
Model.fit(x_train, y_train)
Y_predict = model.predict(x_test)
```

in case the 'multi_class' option is set to 'multinomial'. (the 'multinomial' option is supported by the 'lbfgs', 'sag' and 'newton-cg' solvers.)

For read:

https://www.geeksforgeeks.org/understanding-logistic-regression/

https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_classification_algorithms_logistic_regression.htm

https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148

K Nearest Neighbors (KNN):

K nearest neighbors is a ML algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). it can be used for both classification and regression predictive problems.

To choose the right value for K, use 'Elbow Method' for choosing the best K value.

```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=5)
model.fit(x_train, y_train)
Y_predict = model.predict(x_test)
```

For read:

https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761

https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/

Support Vector machine (SVC):

For read:

https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm

Decision Tress:

Decision tree is a flowchart like tree structure. the two main entities of a tree are decision nodes, branches represent an outcome of the test and Leaf nodes represent class labels or class distribution. it can be used for both classification and regression tasks.

```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
model.fit(x_train, Y_train)
Y_predict = model.predict(x_test)
```

For read:

https://www.datacamp.com/community/tutorials/decision-tree-classification-python

https://towardsdatascience.com/decision-tree-classification-de64fc4d5aac

Random Forest:

For read:

https://www.datacamp.com/community/tutorials/random-forests-classifier-python
https://www.geeksforgeeks.org/random-forest-regression-in-python/?ref=lbp

XGBoost:

For read:
https://machinelearningmastery.com/develop-first-xgboost-model-python-scikit-learn/
https://www.datacamp.com/community/tutorials/xgboost-in-python

6-Evaluating Model Performance

All results in file "result.txt"

7-Link of Web Application deployed on Heroku

https://healtcare-system.herokuapp.com/