PROJECT NAME : HOSPITAL HEALTH CARE DATA FOR ANALYSIS
TEAM ID : PNT2022TMID16345

```
import os
for dirname, _, filenames in os.walk('input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
import seaborn as sns
from warnings import filterwarnings
filterwarnings('ignore')
pd.options.display.max_columns = None
pd.options.display.max_rows = None
pd.options.display.float_format = '{:.6f}'.format
from sklearn.model_selection import train_test_split
import statsmodels
import statsmodels.api as sm
from sklearn.preprocessing import StandardScaler
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from sklearn.metrics import cohen_kappa_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn import tree
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import
AdaBoostClassifier,GradientBoostingClassifier
from catboost import CatBoostClassifier
from sklearn.feature_selection import RFE
plt.rcParams['figure.figsize'] = [15,8]
```

```
train = pd.read_csv("D:/HealthCare/train_data.csv")
test = pd.read_csv("D:/HealthCare/test_data.csv")
```

```
train.head()
```

```
   case_id  Hospital_code Hospital_type_code  City_Code_Hospital  \
0        1              8                  c                   3
1        2              2                  c                   5
2        3             10                  e                   1
3        4             26                  b                   2
4        5             26                  b                   2


  Hospital_region_code Available Extra Rooms in Hospital
Department \
0                    Z                                  3
```

```
radiotherapy
1                    Z                                  2
radiotherapy
2                    X                                  2
anesthesia
3                    Y                                  2
radiotherapy
4                    Y                                  2
radiotherapy

  Ward_Type Ward_Facility_Code Bed Grade patientid
City_Code_Patient \
0         R                  F   2.000000      31397
7.000000
1         S                  F   2.000000      31397
7.000000
2         S                  E   2.000000      31397
7.000000
3         R                  D   2.000000      31397
7.000000
4         S                  D   2.000000      31397
7.000000

  Type of Admission Severity of Illness  Visitors with Patient    Age
\
0         Emergency            Extreme                        2  51-60

1            Trauma            Extreme                        2  51-60

2            Trauma            Extreme                        2  51-60

3            Trauma            Extreme                        2  51-60

4            Trauma            Extreme                        2  51-60


   Admission_Deposit   Stay
0        4911.000000   0-10
1        5954.000000  41-50
2        4745.000000  31-40
3        7272.000000  41-50
4        5558.000000  41-50
```

```
print(train.shape)
print(test.shape)
```

```
(318438, 18)
(137057, 17)
```

```
train.dtypes
```

```
case_id                                int64
Hospital_code                          int64
Hospital_type_code                    object
City_Code_Hospital                     int64
Hospital_region_code                  object
Available  Extra  Rooms  in  Hospital  int64
Department                            object
Ward_Type                             object
Ward_Facility_Code                    object
Bed Grade                            float64
patientid                              int64
City_Code_Patient                    float64
Type of Admission                     object
Severity of Illness                   object
Visitors with Patient                  int64
Age                                   object
Admission_Deposit                    float64
Stay                                  object
dtype: object
```

```
train.nunique()
```

```
case_id                              318438
Hospital_code                            32
Hospital_type_code                        7
City_Code_Hospital                       11
Hospital_region_code                      3
Available Extra Rooms in Hospital        18
Department                                5
Ward_Type                                 6
Ward_Facility_Code                        6
Bed Grade                                 4
patientid                             92017
City_Code_Patient                        37
Type of Admission                         3
Severity of Illness                       3
Visitors with Patient                    28
Age                                      10
Admission_Deposit                      7300
Stay                                     11
dtype: int64
```

```
train.duplicated().sum()
```

```
0
```

```
train['Hospital_code'] = train['Hospital_code'].astype(object)
train['City_Code_Hospital'] =
train['City_Code_Hospital'].astype(object)
train['Available Extra Rooms in Hospital'] = train['Available Extra
Rooms in Hospital'].astype(object)
```

```
train['Bed Grade'] = train['Bed Grade'].astype(object)
train['City_Code_Patient'] = train['City_Code_Patient'].astype(object)
```

```
train.dtypes
```

```
case_id                                int64
Hospital_code                         object
Hospital_type_code                    object
City_Code_Hospital                    object
Hospital_region_code                  object
Available  Extra  Rooms  in  Hospital object
Department                            object
Ward_Type                             object
Ward_Facility_Code                    object
Bed Grade                             object
patientid                              int64
City_Code_Patient                     object
Type of Admission                     object
Severity of Illness                   object
Visitors with Patient                  int64
Age                                   object
Admission_Deposit                    float64
Stay                                  object
dtype: object
```

```
test['Hospital_code'] = test['Hospital_code'].astype(object)
test['City_Code_Hospital'] = test['City_Code_Hospital'].astype(object)
test['Available Extra Rooms in Hospital'] = test['Available Extra
Rooms in Hospital'].astype(object)
test['Bed Grade'] = test['Bed Grade'].astype(object)
test['City_Code_Patient'] = test['City_Code_Patient'].astype(object)
```

```
test.dtypes
```

```
case_id                                int64
Hospital_code                         object
Hospital_type_code                    object
City_Code_Hospital                    object
Hospital_region_code                  object
Available  Extra  Rooms  in  Hospital object
Department                            object
Ward_Type                             object
Ward_Facility_Code                    object
Bed Grade                             object
patientid                              int64
City_Code_Patient                     object
Type of Admission                     object
Severity of Illness                   object
Visitors with Patient                  int64
Age                                   object
```

```
Admission_Deposit                    float64
dtype: object
```

```
train.drop(['case_id', 'patientid'], axis=1, inplace=True)
```

```
test.drop(['case_id', 'patientid'], axis=1, inplace=True)
```

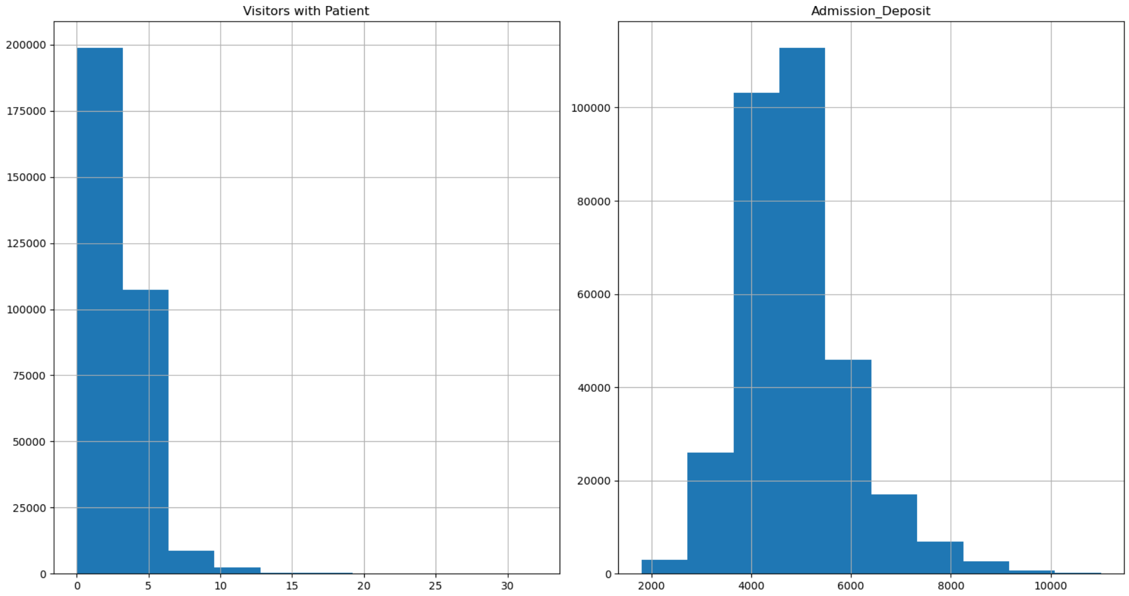
```
train['Stay'] .replace ('More than 100 Days', '100+', inplace=True)
```

```
plt.figure(figsize=(15,8))
```

```
train.drop('Stay', axis = 1).hist()
plt.tight_layout()
plt.show()
```

```
print('Skewness:')
train.drop('Stay', axis = 1).skew()
```

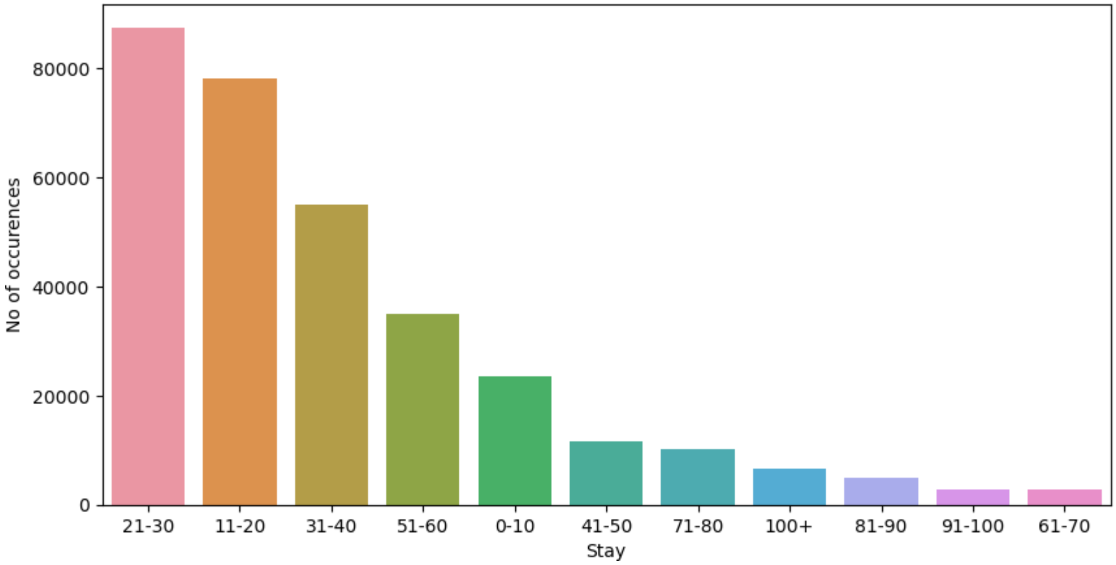
```
<Figure size 1500x800 with 0 Axes>
```

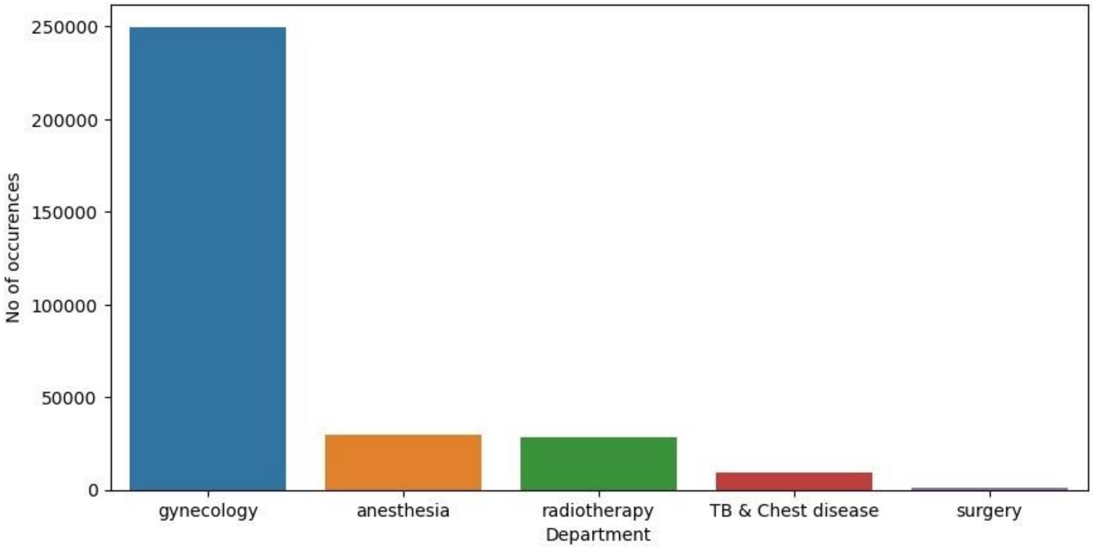


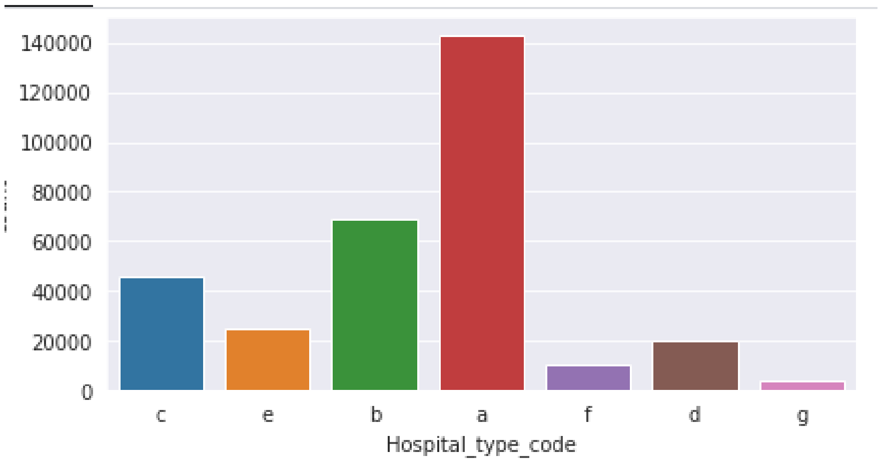
```
Skewness:

Hospital_code                       -0.280783
City_Code_Hospital                   0.538809
Available Extra Rooms in Hospital    0.971930
Bed Grade                            0.051754
City_Code_Patient                    1.581736
Visitors with Patient                3.137125
Admission_Deposit                    0.931454
dtype: float64
```

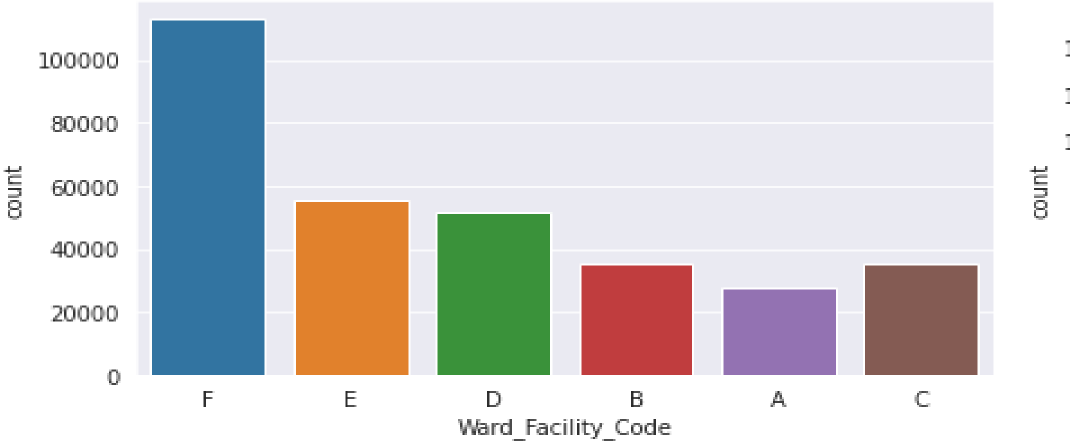
```
train['Stay'] .replace('More than 100 Days', '>100', inplace=True)
```

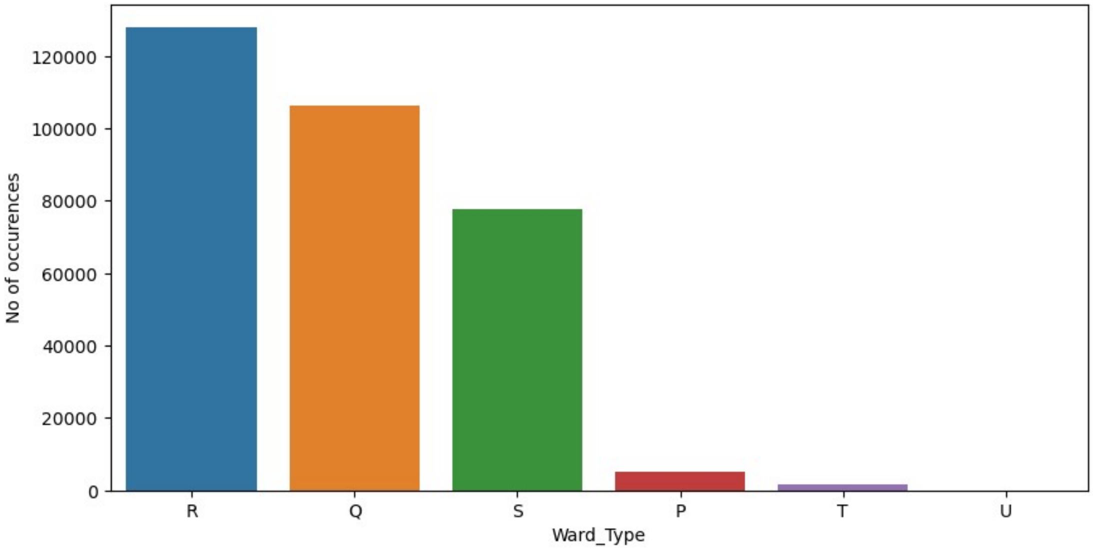
```
for i in [ 'Stay', 'Department', 'Available Extra Rooms in Hospital',
'Ward_Type' ,'Ward_Facility_Code', 'Age',
          'Type of Admission', 'Severity of Illness', 'Bed Grade',
'Hospital_region_code', 'Hospital_type_code' ,
          'City_Code_Hospital', 'Hospital_code', 'City_Code_Patient',
'Visitors with Patient']:
            count = train[i].value_counts()

            plt.figure(figsize=(10,5))

            sns.barplot(x=count.index.values,
y=count.values,data=train)

            plt.xlabel(i)
            plt.ylabel('No of occurences')
```

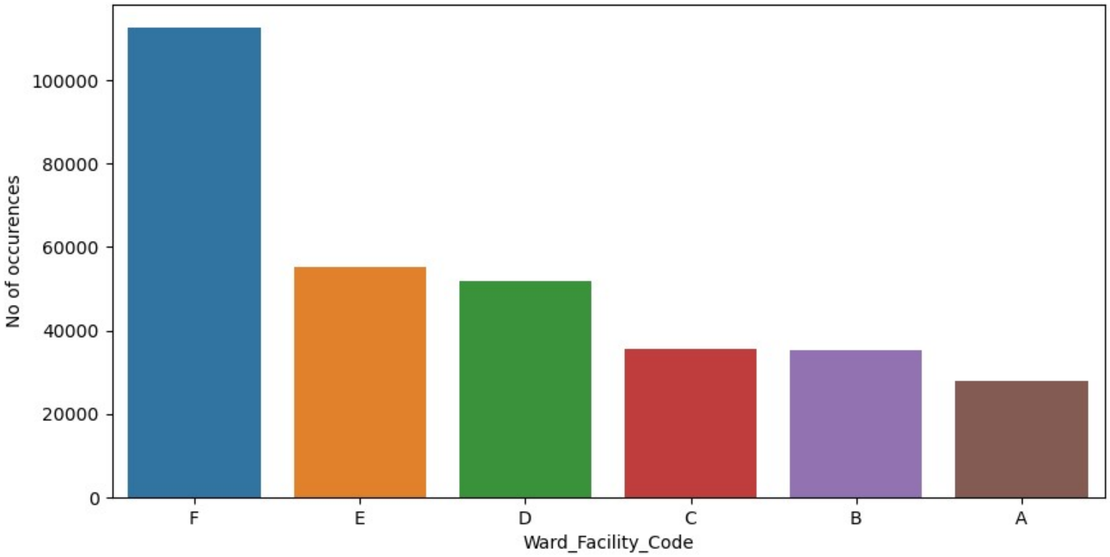


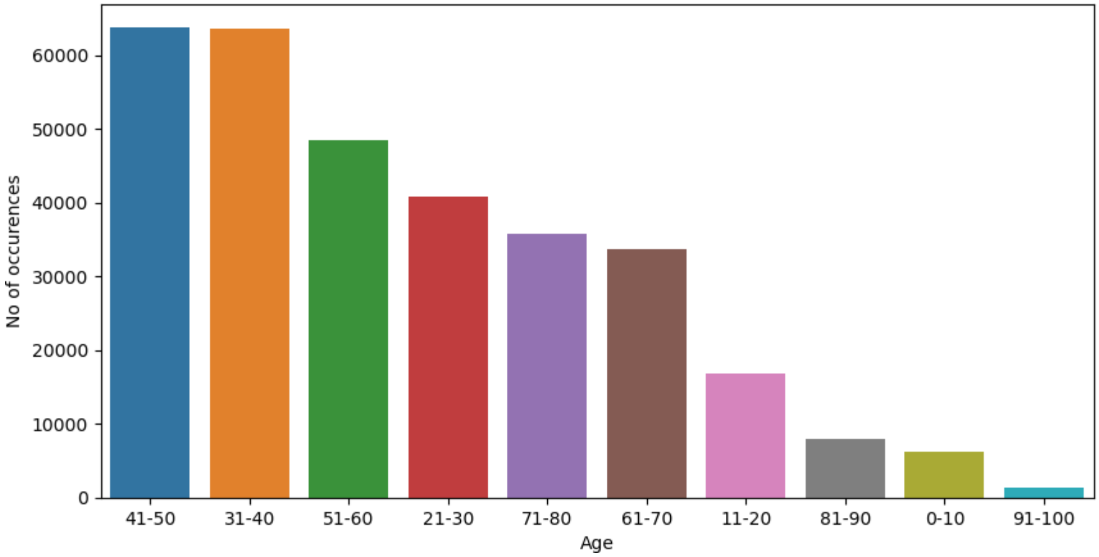


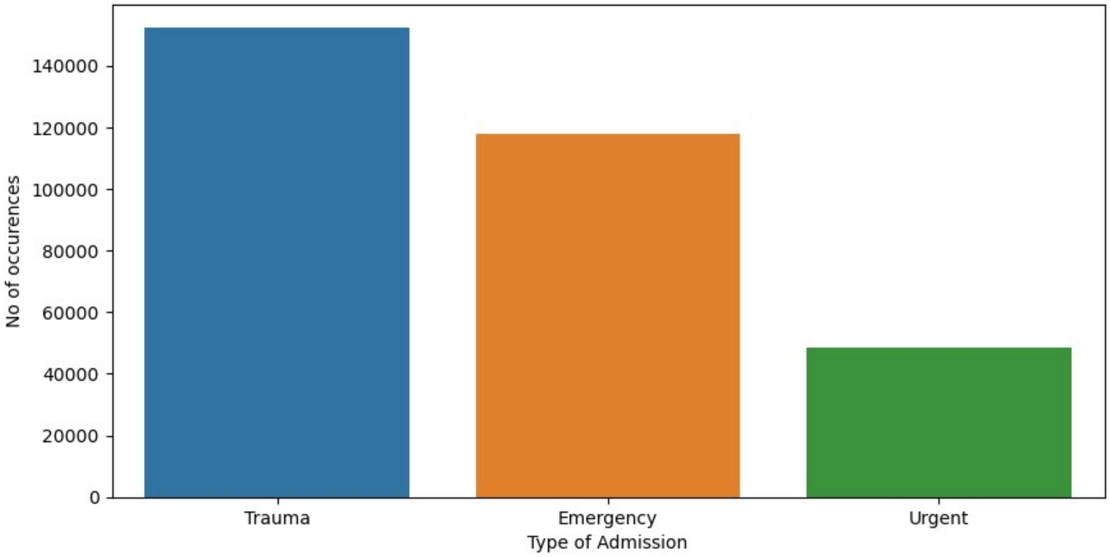

140000
120000
100000
80000
60000
40000
20000
0
c
e
b
a
f
d
g
Hospital_type_code

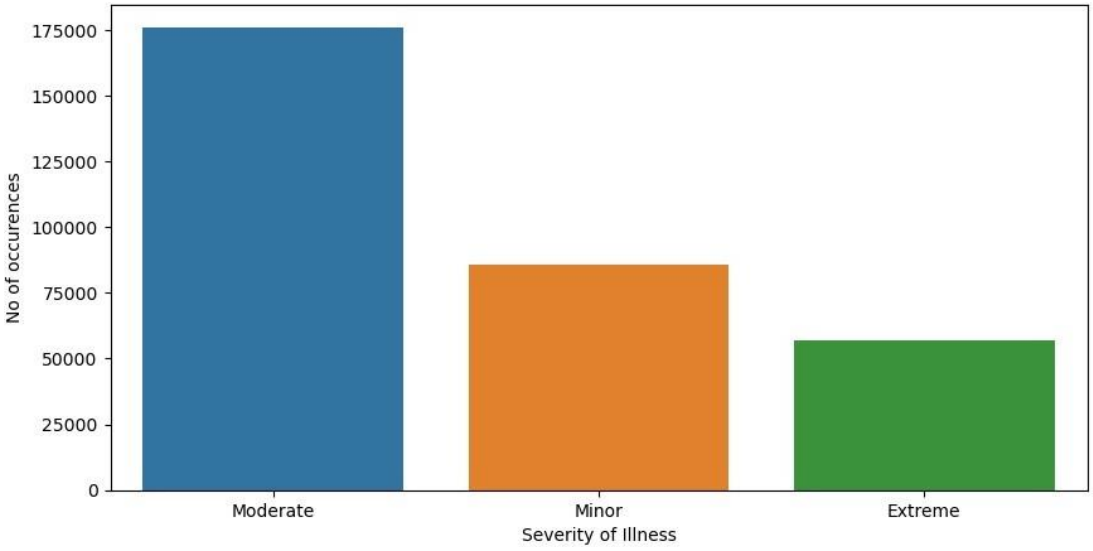

100000
80000
60000
40000
20000
0
count
F
E
D
B
A
C
Ward_Facility_Code
count

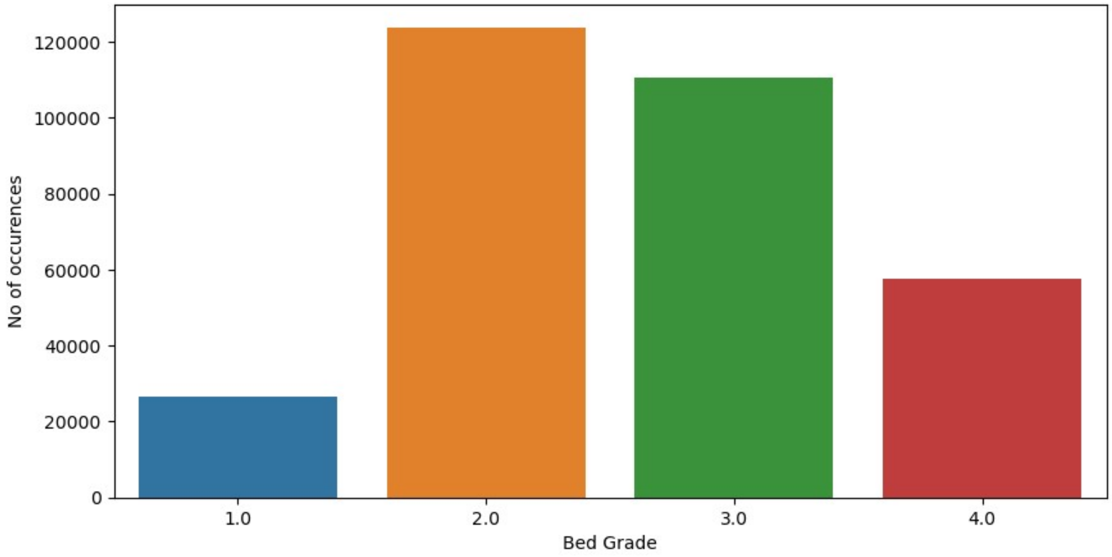

120000
100000
80000
60000
40000
20000
0
No of occurences
R
Q
S
P
T
U
Ward_Type


No of occurences
100000
80000
60000
40000
20000
0
F
E
D
C
B
A
Ward_Facility_Code

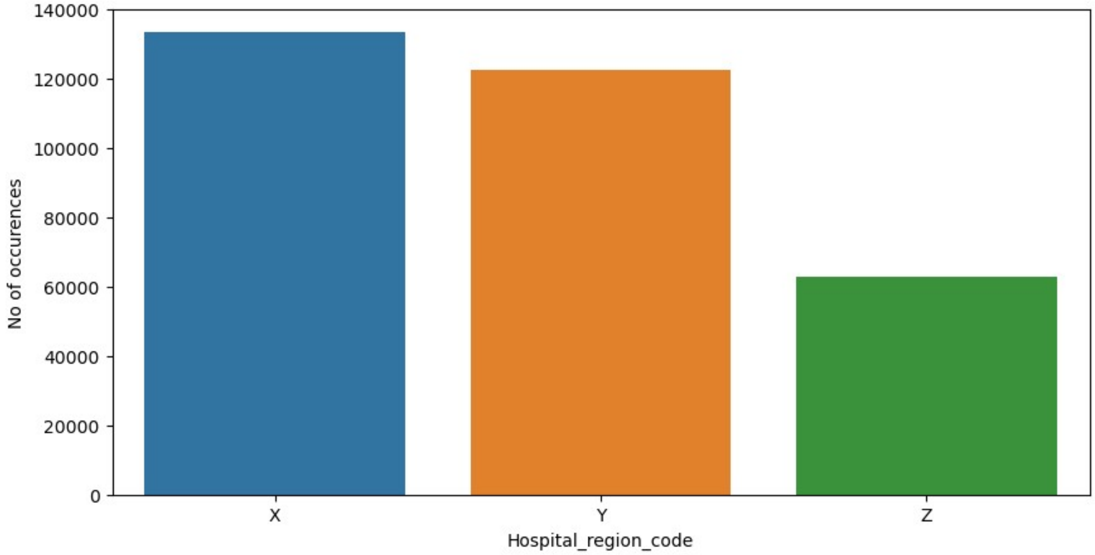

No of occurences
60000
50000
40000
30000
20000
10000
0
41-50
31-40
51-60
21-30
71-80
61-70
11-20
81-90
0-10
91-100
Age


No of occurences
0
20000
40000
60000
80000
100000
120000
140000
Trauma
Emergency
Urgent
Type of Admission

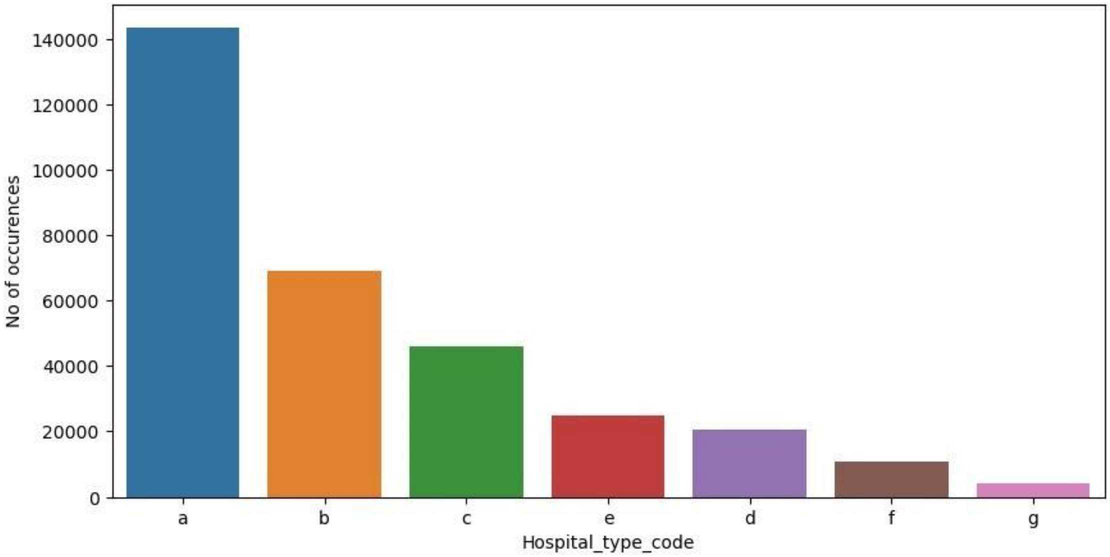

No of occurences
0
25000
50000
75000
100000
125000
150000
175000
Moderate
Minor
Extreme
Severity of Illness

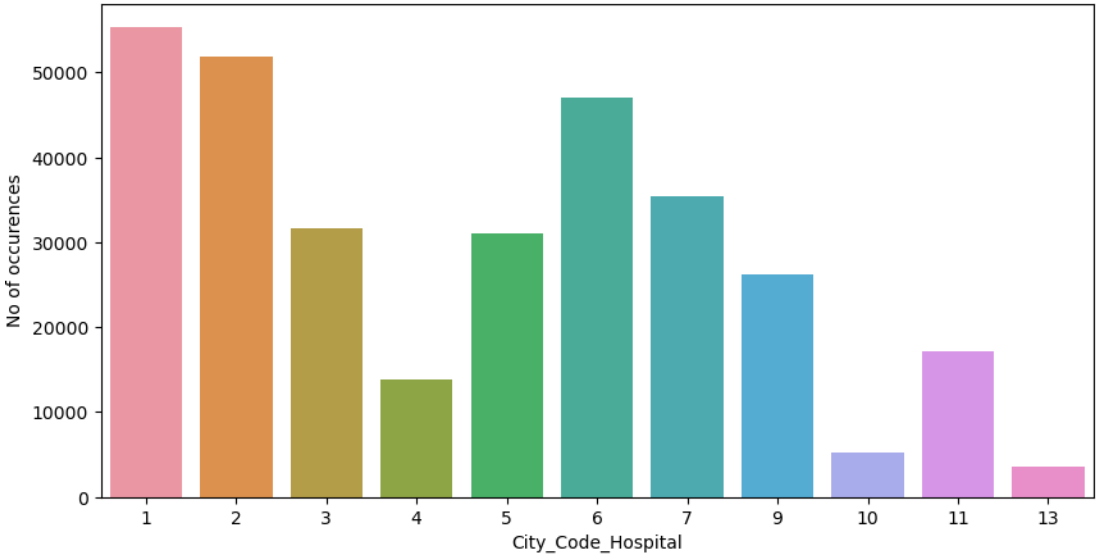


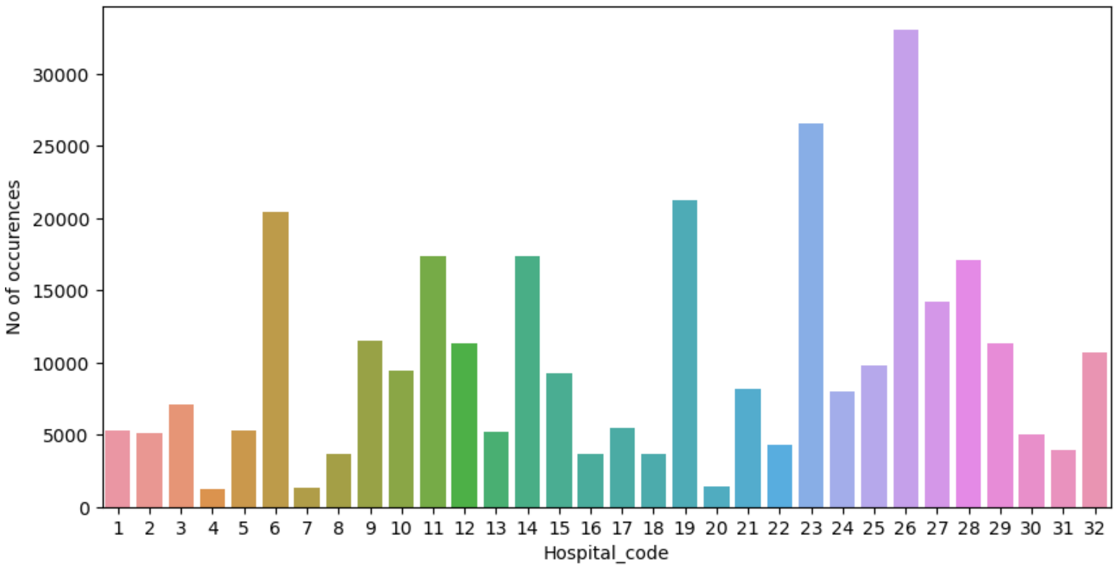
No of occurences
0
20000
40000
60000
80000
100000
120000
1.0
2.0
3.0
4.0
Bed Grade

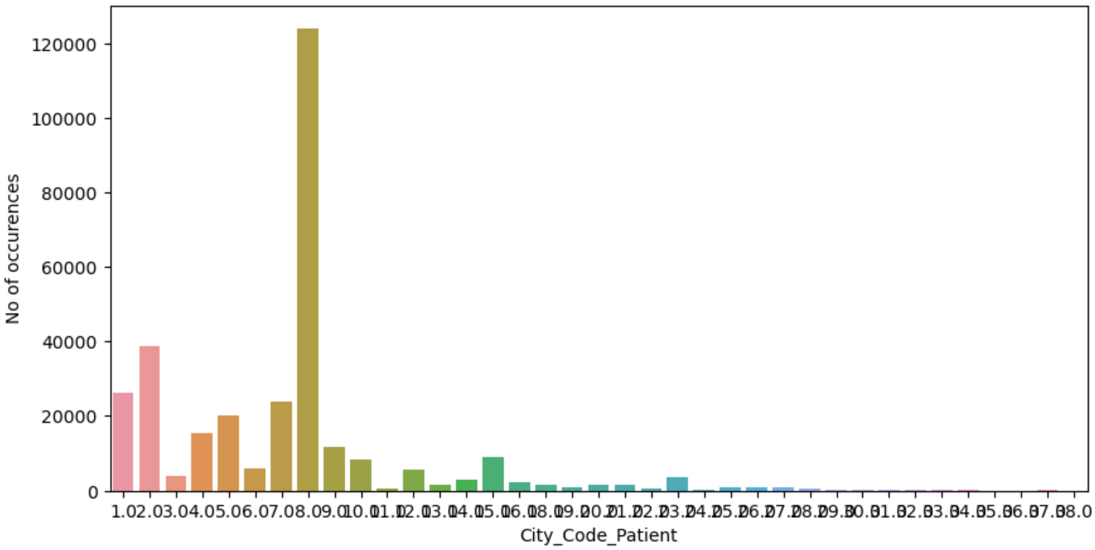


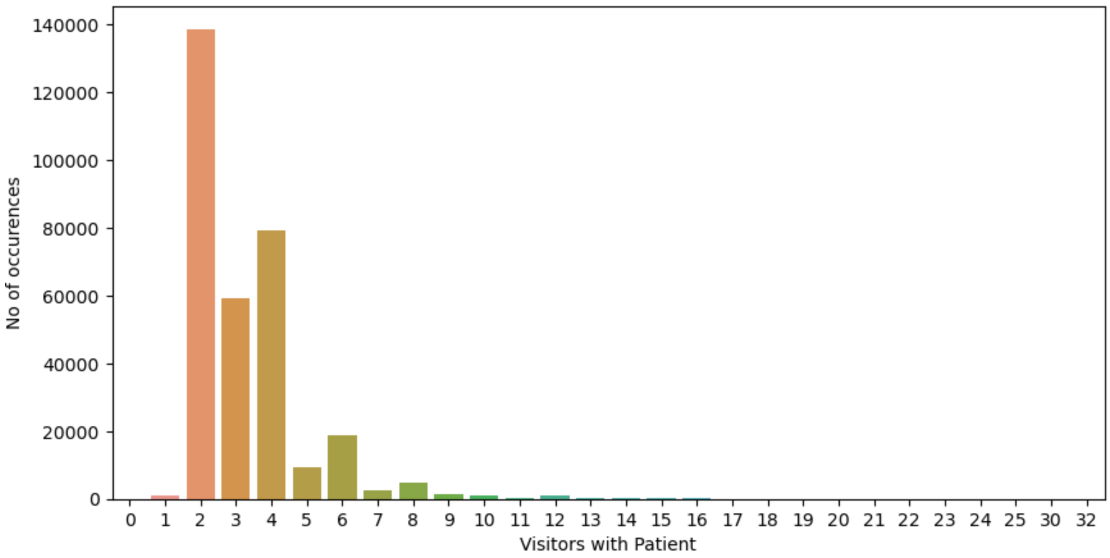
No of occurences
0
20000
40000
60000
80000
100000
120000
140000
X
Y
Z
Hospital_region_code


140000
120000
100000
80000
60000
40000
20000
0
No of occurences
a
b
c
e
d
f
g
Hospital_type_code

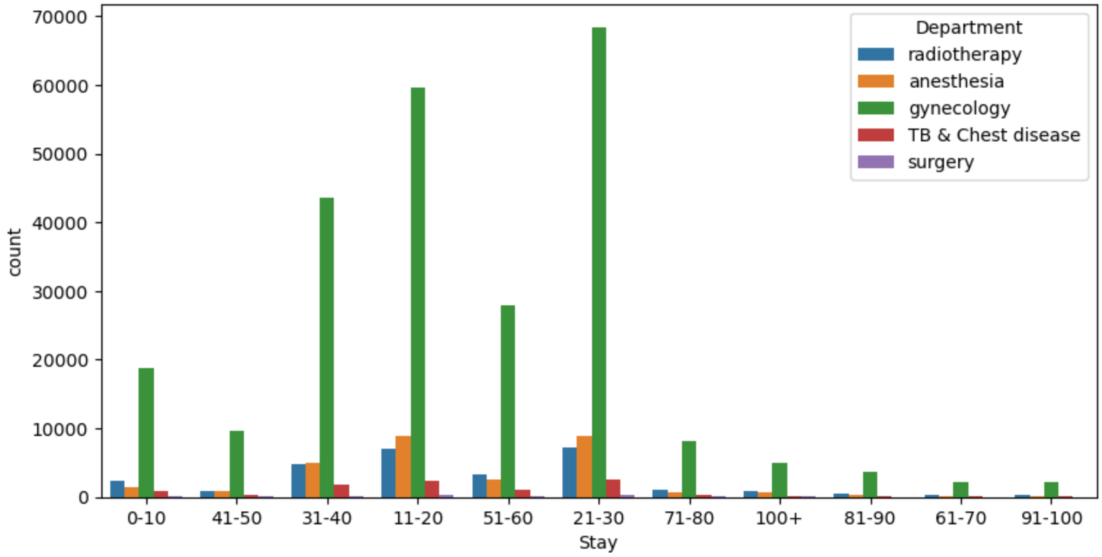

50000
40000
30000
20000
10000
0
No of occurences
1
2
3
4
5
6
7
9
10
11
13
City_Code_Hospital

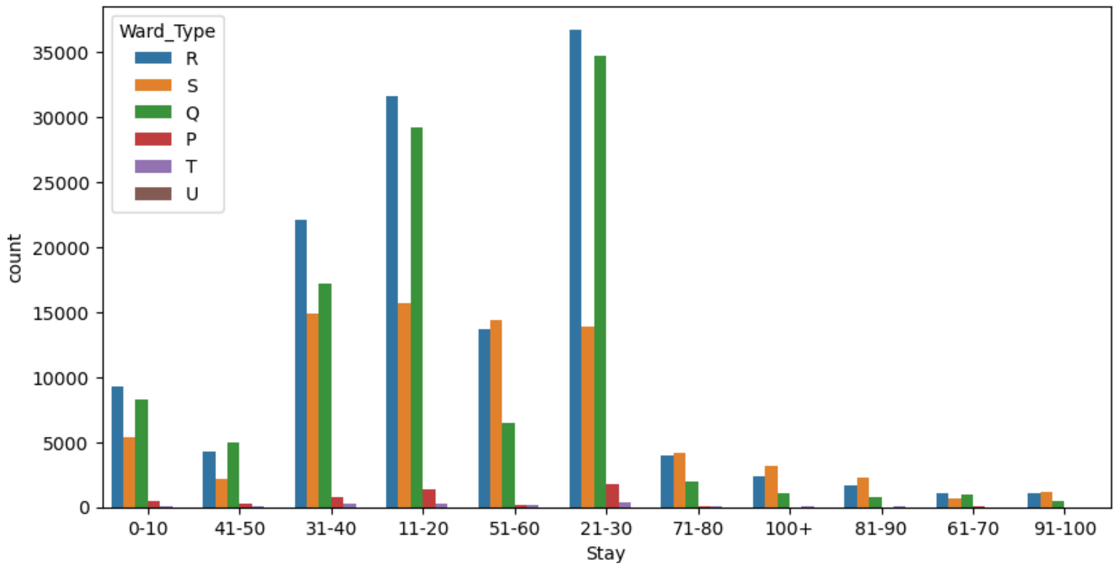

No of occurences
0
5000
10000
15000
20000
25000
30000
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
Hospital_code

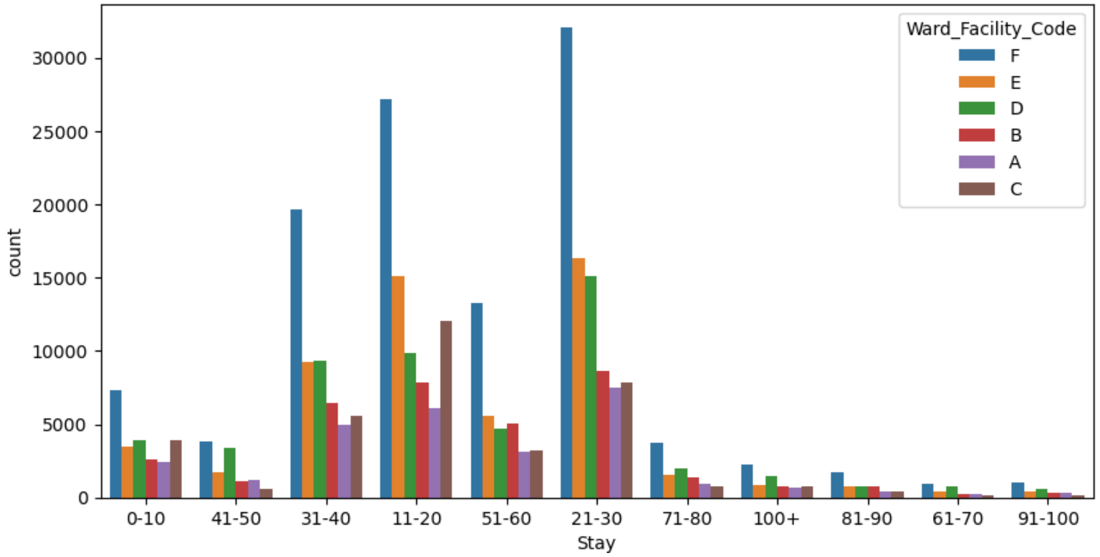

No of occurences
0
20000
40000
60000
80000
100000
120000
1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 13.0 14.0 15.0 16.0 18.0 19.0 20.0 21.0 22.0 23.0 24.0 25.0 26.0 27.0 28.0 29.0 30.0 31.0 32.0 33.0 34.0 35.0 36.0 37.0 38.0
City_Code_Patient

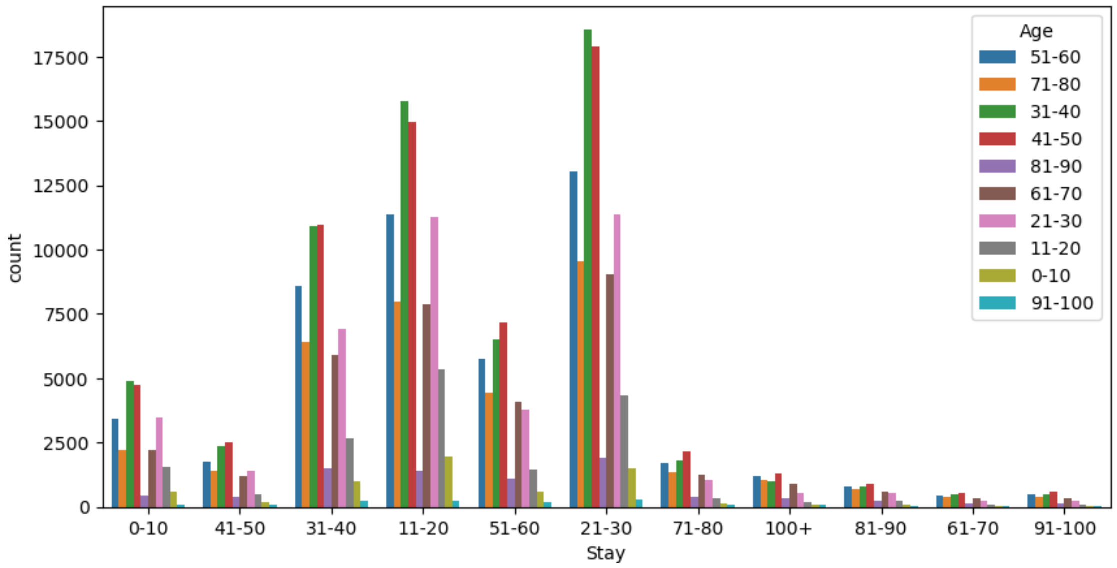


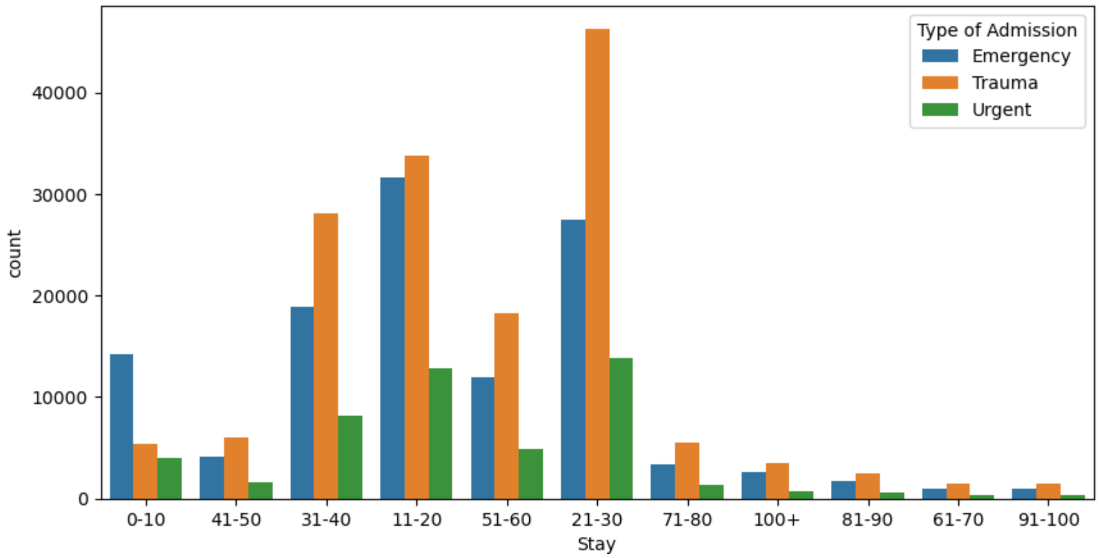
```
for i in [ 'Department', 'Ward_Type' ,'Ward_Facility_Code', 'Age',
'Type of Admission', 'Severity of Illness',
          'Bed Grade', 'Hospital_region_code', 'Hospital_type_code' ]:

            plt.figure(figsize=(10,5))

            sns.countplot(x='Stay',hue=i,data=train)
```

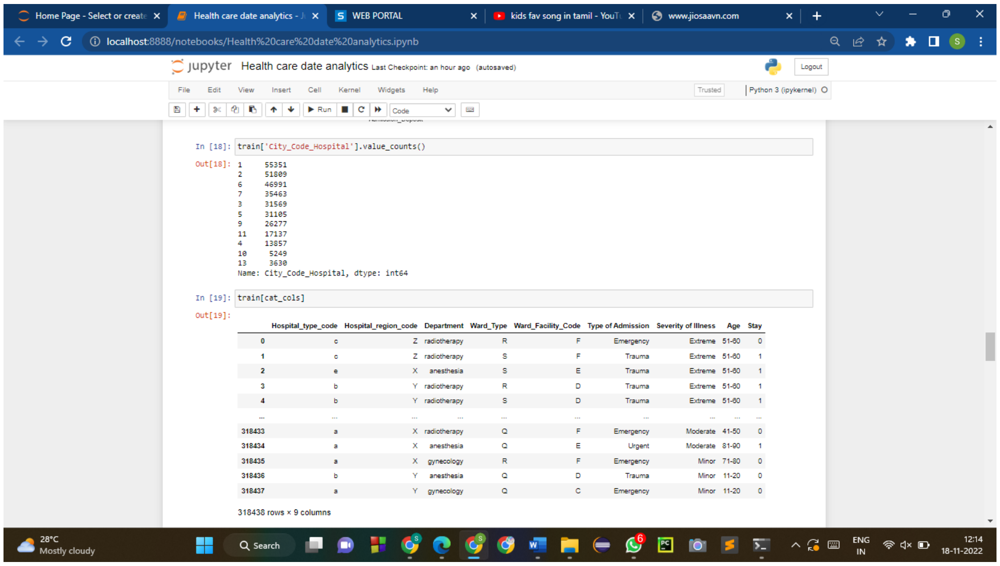



Ward_Type
R
S
Q
P
T
U
count
0
5000
10000
15000
20000
25000
30000
35000
0-10
41-50
31-40
11-20
51-60
21-30
71-80
100+
81-90
61-70
91-100
Stay


Ward_Facility_Code
F
E
D
B
A
C
count
0
5000
10000
15000
20000
25000
30000
0-10
41-50
31-40
11-20
51-60
21-30
71-80
100+
81-90
61-70
91-100
Stay


Age
51-60
71-80
31-40
41-50
81-90
61-70
21-30
11-20
0-10
91-100
count
17500
15000
12500
10000
7500
5000
2500
0
0-10
41-50
31-40
11-20
51-60
21-30
71-80
100+
81-90
61-70
91-100
Stay


Type of Admission
Emergency
Trauma
Urgent
count
40000
30000
20000
10000
0
0-10
41-50
31-40
11-20
51-60
21-30
71-80
100+
81-90
61-70
91-100
Stay



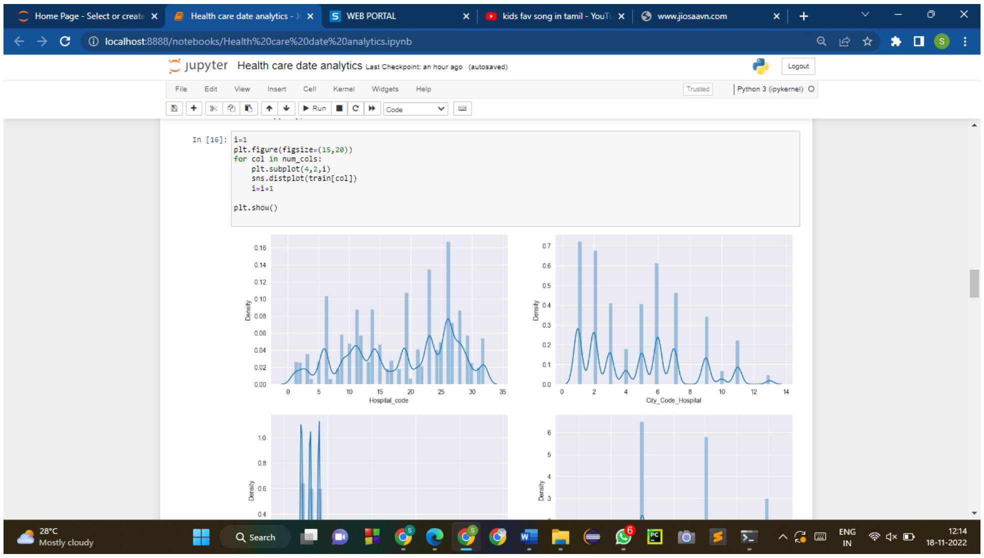
Home Page - Select or create
Health care date analytics - J
WEB PORTAL
kids fav song in tamil - YouT
www.jiosaavn.com
localhost:8888/notebooks/Health%20care%20date%20analytics.ipynb
jupyter Health care date analytics Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
28°C
Mostly cloudy
Search
ENG
IN
12:14
18-11-2022

```
In [18]: train['City_Code_Hospital'].value_counts()
Out[18]: 1     55351
         2     51809
         6     46991
         7     35463
         3     31569
         5     31105
         9     26277
         11    17137
         4     13857
         10     5249
         13     3630
         Name: City_Code_Hospital, dtype: int64
```

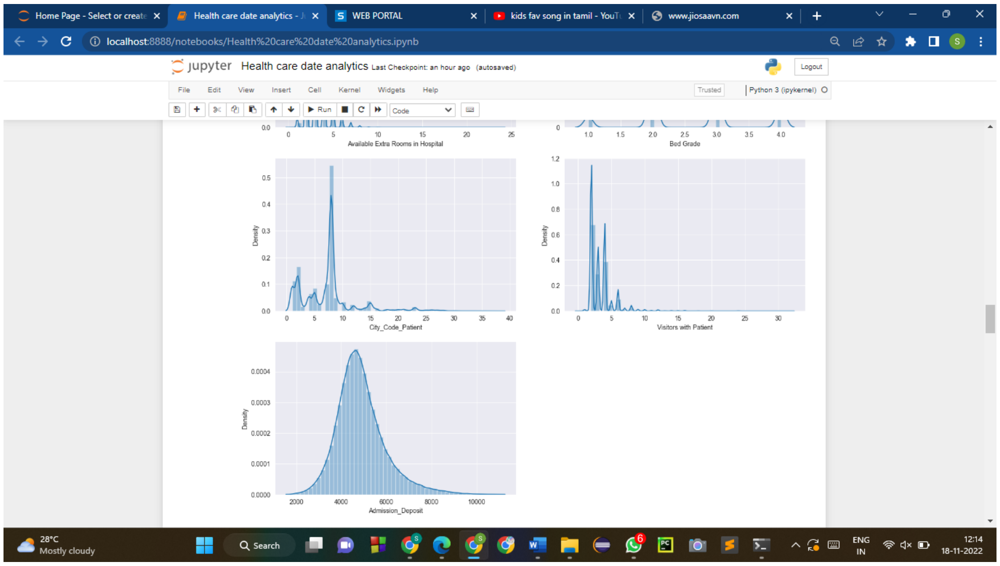
```
In [19]: train[cat_cols]
Out[19]:
```

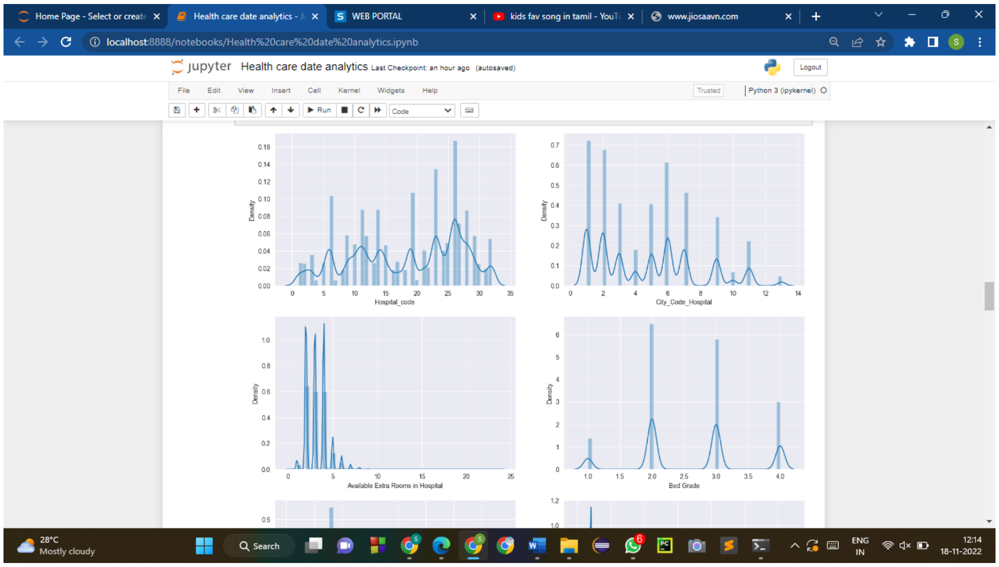
	Hospital_type_code	Hospital_region_code	Department	Ward_Type	Ward_Facility_Code	Type of Admission	Severity of Illness	Age	Stay
0	c	Z	radiotherapy	R	F	Emergency	Extreme	51-60	0
1	c	Z	radiotherapy	S	F	Trauma	Extreme	51-60	1
2	e	X	anesthesia	S	E	Trauma	Extreme	51-60	1
3	b	Y	radiotherapy	R	D	Trauma	Extreme	51-60	1
4	b	Y	radiotherapy	S	D	Trauma	Extreme	51-60	1
...	...	...	...	...	...	...	...	...	...
318433	a	X	radiotherapy	Q	F	Emergency	Moderate	41-50	0
318434	a	X	anesthesia	Q	E	Urgent	Moderate	81-90	1
318435	a	X	gynecology	R	F	Emergency	Minor	71-80	0
318436	b	Y	anesthesia	Q	D	Trauma	Minor	11-20	0
318437	a	Y	gynecology	Q	C	Emergency	Minor	11-20	0

318438 rows × 9 columns



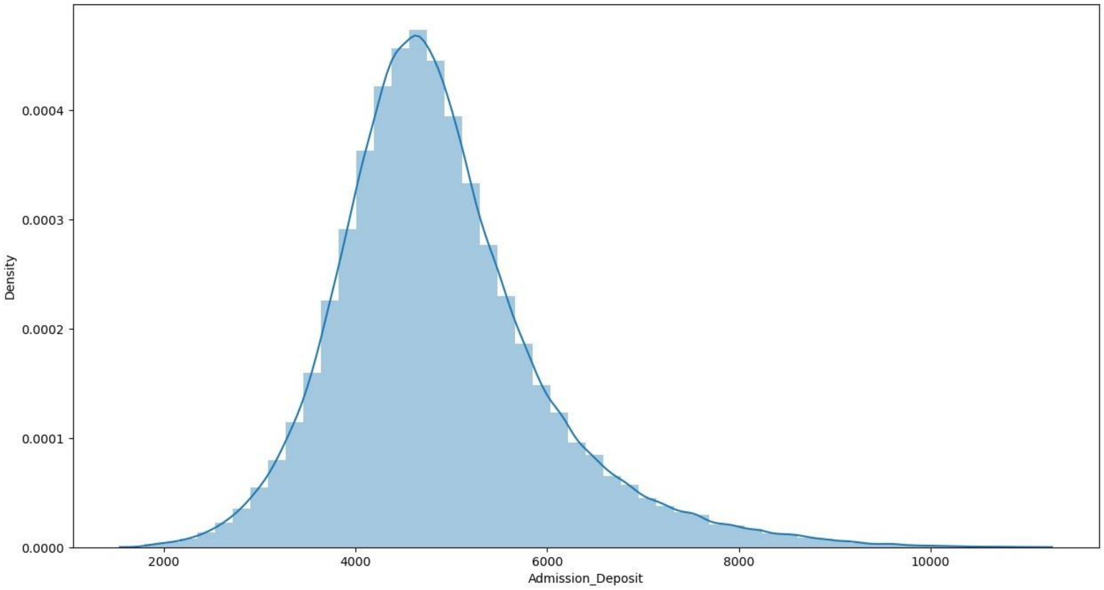
```
In [16]: i=1
         plt.figure(figsize=(15,20))
         for col in num_cols:
             plt.subplot(4,2,i)
             sns.distplot(train[col])
             i=i+1

         plt.show()
```


Home Page - Select or create
Health care date analytics
WEB PORTAL
kids fav song in tamil - YouTube
www.jiosaavn.com
localhost:8888/notebooks/Health%20care%20date%20analytics.ipynb
jupyter Health care date analytics Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
Available Extra Rooms in Hospital
Bed Grade
Density
City_Code_Patient
Visitors with Patient
Admission_Deposit
28°C
Mostly cloudy
Search
ENG
IN
12:14
18-11-2022


Home Page - Select or create
Health care date analytics
WEB PORTAL
kids fav song in tamil - YouTube
www.jiosaavn.com
localhost:8888/notebooks/Health%20care%20date%20analytics.ipynb
jupyter Health care date analytics Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
Density
Hospital_code
City_Code_Hospital
Available Extra Rooms in Hospital
Bed Grade
28°C
Mostly cloudy
Search
ENG
IN
12:14
18-11-2022

```
sns.distplot( train['Admission_Deposit'])
```

```
<AxesSubplot:xlabel='Admission_Deposit', ylabel='Density'>
```



```
Total = train.isnull().sum().sort_values(ascending=False)

Percent =
(train.isnull().sum()*100/train.isnull().count()).sort_values(ascendin
g=False)

missing_data = pd.concat([Total, Percent], axis = 1, keys = ['Total',
'Percentage of Missing Values'])
missing_data
```

```
                                   Total  Percentage of Missing Values
City_Code_Patient                   4532                      1.423197
Bed Grade                            113                      0.035486
Hospital_code                          0                      0.000000
Hospital_type_code                     0                      0.000000
City_Code_Hospital                     0                      0.000000
Hospital_region_code                   0                      0.000000
Available Extra Rooms in Hospital      0                      0.000000
Department                             0                      0.000000
Ward_Type                              0                      0.000000
Ward_Facility_Code                     0                      0.000000
Type of Admission                      0                      0.000000
Severity of Illness                    0                      0.000000
Visitors with Patient                  0                      0.000000
Age                                    0                      0.000000
Admission_Deposit                      0                      0.000000
Stay                                   0                      0.000000
```

```
Total = train.isnull().sum().sort_values(ascending=False)

Percent =
(train.isnull().sum()*100/train.isnull().count()).sort_values(ascendin
g=False)

missing_data = pd.concat([Total, Percent], axis = 1, keys = ['Total',
'Percentage of Missing Values'])
missing_data
```

```
                                   Total  Percentage of Missing Values
City_Code_Patient                   4532                      1.423197
Bed Grade                            113                      0.035486
Hospital_code                          0                      0.000000
Hospital_type_code                     0                      0.000000
City_Code_Hospital                     0                      0.000000
Hospital_region_code                   0                      0.000000
Available Extra Rooms in Hospital      0                      0.000000
Department                             0                      0.000000
Ward_Type                              0                      0.000000
Ward_Facility_Code                     0                      0.000000
Type of Admission                      0                      0.000000
Severity of Illness                    0                      0.000000
Visitors with Patient                  0                      0.000000
Age                                    0                      0.000000
Admission_Deposit                      0                      0.000000
Stay                                   0                      0.000000
```

```
Total = test.isnull().sum().sort_values(ascending=False)

Percent =
(test.isnull().sum()*100/test.isnull().count()).sort_values(ascending=
False)

missing_data = pd.concat([Total, Percent], axis = 1, keys = ['Total',
'Percentage of Missing Values'])
missing_data
```

```
                                   Total  Percentage of Missing Values
City_Code_Patient                   2157                      1.573798
Bed Grade                             35                      0.025537
Hospital_code                          0                      0.000000
Hospital_type_code                     0                      0.000000
City_Code_Hospital                     0                      0.000000
Hospital_region_code                   0                      0.000000
Available Extra Rooms in Hospital      0                      0.000000
Department                             0                      0.000000
Ward_Type                              0                      0.000000
Ward_Facility_Code                     0                      0.000000
Type of Admission                      0                      0.000000
Severity of Illness                    0                      0.000000
Visitors with Patient                  0                      0.000000
Age                                    0                      0.000000
Admission_Deposit                      0                      0.000000
```

```
train.dropna(subset = ['Bed Grade', 'City_Code_Patient'], inplace =
True)
```

```
test['Bed Grade'] = test['Bed Grade'].fillna(test['Bed Grade'].mode()
[0], inplace = True)
test['City_Code_Patient'] =
test['City_Code_Patient'].fillna(test['City_Code_Patient'].mode()[0],
inplace = True)
```

```
df_num_train = train.select_dtypes([np.number])
df_num_train.head()
```

```
   Visitors with Patient  Admission_Deposit
0                      2        4911.000000
1                      2        5954.000000
2                      2        4745.000000
3                      2        7272.000000
4                      2        5558.000000
```

```
df_cat_train = train.select_dtypes([np.object])
df_cat_train.head()
```

```
  Hospital_code Hospital_type_code City_Code_Hospital
Hospital_region_code \
0             8                  c                  3
Z
1             2                  c                  5
Z
2            10                  e                  1
X
3            26                  b                  2
Y
4            26                  b                  2
Y

  Available Extra Rooms in Hospital    Department Ward_Type  \
0                                  3  radiotherapy         R
1                                  2  radiotherapy         S
2                                  2    anesthesia         S
3                                  2  radiotherapy         R
4                                  2  radiotherapy         S

  Ward_Facility_Code Bed Grade City_Code_Patient Type of Admission  \
0                  F  2.000000          7.000000         Emergency
1                  F  2.000000          7.000000            Trauma
2                  E  2.000000          7.000000            Trauma
3                  D  2.000000          7.000000            Trauma
4                  D  2.000000          7.000000            Trauma

  Severity of Illness    Age   Stay
0             Extreme  51-60   0-10
```

```
1             Extreme  51-60  41-50
2             Extreme  51-60  31-40
3             Extreme  51-60  41-50
4             Extreme  51-60  41-50
```

```
df_num_test = test.select_dtypes([np.number])
df_num_test.head()
```

```
   Visitors with Patient  Admission_Deposit
0                      2        3095.000000
1                      4        4018.000000
2                      3        4492.000000
3                      3        4173.000000
4                      4        4161.000000
```

```
df_cat_test = test.select_dtypes([np.object])
df_cat_test.head()
```

```
  Hospital_code Hospital_type_code City_Code_Hospital
Hospital_region_code \
0            21                  c                  3
Z
1            29                  a                  4
X
2            26                  b                  2
Y
3             6                  a                  6
X
4            28                  b                 11
X

  Available Extra Rooms in Hospital  Department Ward_Type
Ward_Facility_Code  \
0                                 3  gynecology         S
A
1                                 2  gynecology         S
F
2                                 3  gynecology         Q
D
3                                 3  gynecology         Q
F
4                                 2  gynecology         R
F

  Bed Grade City_Code_Patient Type of Admission Severity of Illness
Age
0      None              None         Emergency            Moderate
71-80
1      None              None            Trauma            Moderate
71-80
2      None              None         Emergency            Moderate
```

```
71-80
3      None              None            Trauma            Moderate
71-80
4      None              None            Trauma            Moderate
71-80

admission_encode = {'Trauma': 1, 'Urgent': 2, 'Emergency' : 3 }
severity_encode   = {'Minor': 1, 'Moderate': 2, 'Extreme': 3 }


df_cat_train['Type of Admission'] = df_cat_train['Type of
Admission'].map (admission_encode)
df_cat_train['Severity of Illness'] = df_cat_train['Severity of
Illness'].map (severity_encode)

df_cat_test['Type of Admission'] = df_cat_test['Type of
Admission'].map (admission_encode)
df_cat_test['Severity of Illness'] = df_cat_test['Severity of
Illness'].map (severity_encode)

df_cat_train['Stay']= df_cat_train['Stay'].replace({'0-10':1, '11-
20':2, '21-30':3, '31-40':4, '41-50':5, '51-60':6,'61-70':7,
                                      '71-80':8,'81-90':9,'91-
100':10,'100+':11})

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

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

df_cat_train['Stay']=df_cat_train['Stay'].astype(int)

from sklearn.preprocessing import LabelEncoder
LE=LabelEncoder()

df_cat_train['Hospital_code']=LE.fit_transform(df_cat_train['Hospital_
code'])
df_cat_train['Hospital_type_code']=LE.fit_transform(df_cat_train['Hosp
ital_type_code'])
df_cat_train['City_Code_Hospital']=LE.fit_transform(df_cat_train['City
_Code_Hospital'])
df_cat_train['Hospital_region_code']=LE.fit_transform(df_cat_train['Ho
spital_region_code'])
df_cat_train['Department']=LE.fit_transform(df_cat_train['Department']
)
```

```
df_cat_train['Ward_Type']=LE.fit_transform(df_cat_train['Ward_Type'])
df_cat_train['Ward_Facility_Code']=LE.fit_transform(df_cat_train['Ward_Facility_Code'])
df_cat_train['City_Code_Patient']=LE.fit_transform(df_cat_train['City_Code_Patient'])
df_cat_train['Bed Grade']=LE.fit_transform(df_cat_train['Bed Grade'])
```

```
df_cat_train.head()
```

```
   Hospital_code  Hospital_type_code  City_Code_Hospital  \
0              7                   2                   2   
1              1                   2                   4   
2              9                   4                   0   
3             25                   1                   1   
4             25                   1                   1   

   Hospital_region_code Available Extra Rooms in Hospital  Department  
\
0                     2                                 3           3   

1                     2                                 2           3   

2                     0                                 2           1   

3                     1                                 2           3   

4                     1                                 2           3   


   Ward_Type  Ward_Facility_Code  Bed Grade  City_Code_Patient  \
0          2                   5          1                  6   
1          3                   5          1                  6   
2          3                   4          1                  6   
3          2                   3          1                  6   
4          3                   3          1                  6   

   Type of Admission  Severity of Illness  Age  Stay  
0                  3                    3    6     1  
1                  1                    3    6     5  
2                  1                    3    6     4  
3                  1                    3    6     5  
4                  1                    3    6     5  
```

```
from sklearn.preprocessing import LabelEncoder
LE=LabelEncoder()

df_cat_test['Hospital_code']=LE.fit_transform(df_cat_test['Hospital_code'])
df_cat_test['Hospital_type_code']=LE.fit_transform(df_cat_test['Hospital_type_code'])
```

```
df_cat_test['City_Code_Hospital']=LE.fit_transform(df_cat_test['City_C
ode_Hospital'])
df_cat_test['Hospital_region_code']=LE.fit_transform(df_cat_test['Hosp
ital_region_code'])
df_cat_test['Department']=LE.fit_transform(df_cat_test['Department'])
df_cat_test['Ward_Type']=LE.fit_transform(df_cat_test['Ward_Type'])
df_cat_test['Ward_Facility_Code']=LE.fit_transform(df_cat_test['Ward_F
acility_Code'])
df_cat_test['City_Code_Patient']=LE.fit_transform(df_cat_test['City_Co
de_Patient'])
df_cat_test['Bed Grade']=LE.fit_transform(df_cat_test['Bed Grade'])

df_cat_test.head()
```

```
   Hospital_code  Hospital_type_code  City_Code_Hospital  \
0             20                   2                   2
1             28                   0                   3
2             25                   1                   1
3              5                   0                   5
4             27                   1                   9

   Hospital_region_code Available Extra Rooms in Hospital  Department  
\
0                     2                                 3           2

1                     0                                 2           2

2                     1                                 3           2

3                     0                                 3           2

4                     0                                 2           2


   Ward_Type  Ward_Facility_Code  Bed Grade  City_Code_Patient  \
0          3                   0          0                  0
1          3                   5          0                  0
2          1                   3          0                  0
3          1                   5          0                  0
4          2                   5          0                  0

   Type of Admission  Severity of Illness  Age
0                  3                    2    8
1                  1                    2    8
2                  3                    2    8
3                  1                    2    8
4                  1                    2    8
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
```

```
num_scaled = sc.fit_transform(df_num_train)

df_num_scaled = pd.DataFrame(num_scaled, columns =
df_num_train.columns)


num_scaled_test = sc.fit_transform(df_num_test)

df_num_scaled_test = pd.DataFrame(num_scaled_test, columns =
df_num_test.columns)

df_num_scaled.shape
```

```
(313793, 2)
```

```
df_cat_train = df_cat_train.reset_index(drop=True)
df_num_scaled = df_num_scaled.reset_index(drop=True)
df_cat_test = df_cat_test.reset_index(drop=True)
df_num_scaled_test = df_num_scaled_test.reset_index(drop=True)

df_cat_train.shape
```

```
(313793, 14)
```

```
df_full = pd.concat([df_num_scaled, df_cat_train],axis=1)
df_full_test =  pd.concat([df_num_scaled_test, df_cat_test],axis=1)

df_full.shape
```

```
(313793, 16)
```

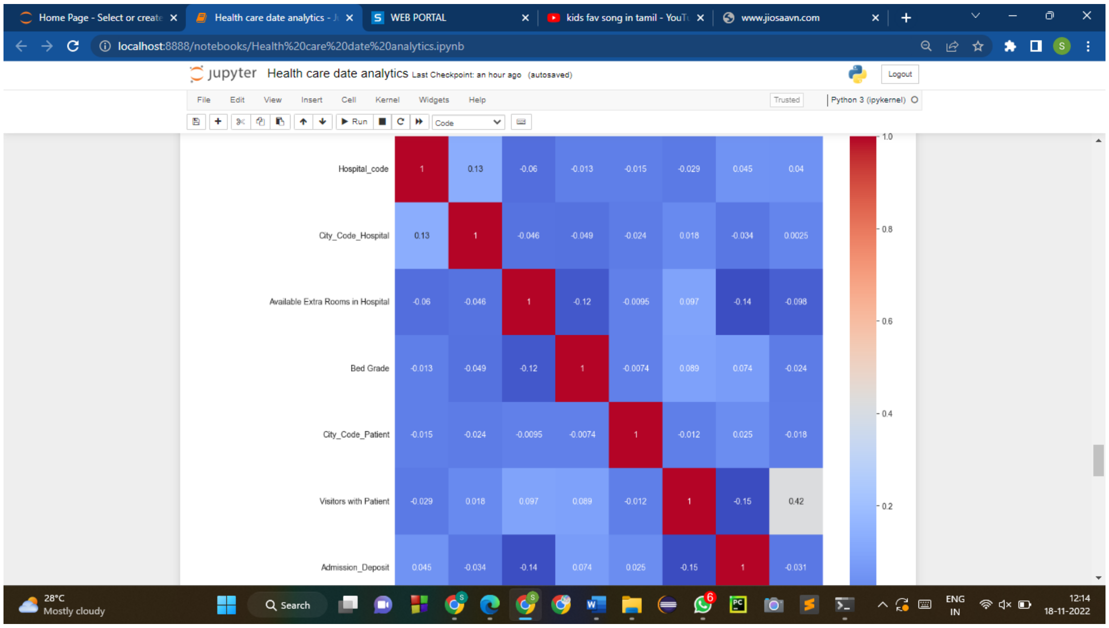
```
df_full.head()
```

```
   Visitors with Patient  Admission_Deposit  Hospital_code  \
0              -0.727035           0.026796              7
1              -0.727035           0.986987              1
2              -0.727035          -0.126025              9
3              -0.727035           2.200344             25
4              -0.727035           0.622427             25

   Hospital_type_code  City_Code_Hospital  Hospital_region_code  \
0                   2                   2                     2
1                   2                   4                     2
2                   4                   0                     0
3                   1                   1                     1
4                   1                   1                     1

  Available Extra Rooms in Hospital  Department  Ward_Type  \
0                                 3           3          2
1                                 2           3          3
2                                 2           1          3
```

```
3                                  2           3          2
4                                  2           3          2

   Ward_Facility_Code  Bed Grade  City_Code_Patient  Type of Admission  \
0                   5          1                  6                  3
1                   5          1                  6                  1
2                   4          1                  6                  1
3                   3          1                  6                  1
4                   3          1                  6                  1

   Severity of Illness  Age  Stay
0                    3    6     1
1                    3    6     5
2                    3    6     4
3                    3    6     5
4                    3    6     5
```

```python
sns.heatmap(df_full.corr(), annot = True)
```

```
<AxesSubplot:>
```



Home Page - Select or create
Health care date analytics - J
WEB PORTAL
kids fav song in tamil - YouT
www.jiosaavn.com
localhost:8888/notebooks/Health%20care%20date%20analytics.ipynb
jupyter
Health care date analytics Last Checkpoint: an hour ago (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3 (ipykernel)
Run
Code
Hospital_code 1 0.13 -0.06 -0.013 -0.015 -0.029 0.045 0.04
City_Code_Hospital 0.13 1 -0.046 -0.049 -0.024 0.018 -0.034 0.0025
Available Extra Rooms in Hospital -0.06 -0.046 1 -0.12 -0.0095 0.097 -0.14 -0.098
Bed Grade -0.013 -0.049 -0.12 1 -0.0074 0.089 0.074 -0.024
City_Code_Patient -0.015 -0.024 -0.0095 -0.0074 1 -0.012 0.025 -0.018
Visitors with Patient -0.029 0.018 0.097 0.089 -0.012 1 -0.15 0.42
Admission_Deposit 0.045 -0.034 -0.14 0.074 0.025 -0.15 1 -0.031
1.0
0.8
0.6
0.4
0.2
28°C
Mostly cloudy
Search
ENG
IN
12:14
18-11-2022

```
X = df_full.drop('Stay',axis=1)
y = df_full['Stay']
```

```
X = sm.add_constant(X)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state
= 10, test_size = 0.3)
```

```
print('X_train', X_train.shape)
print('y_train', y_train.shape)

print('X_test', X_test.shape)
print('y_test', y_test.shape)
```

```
X_train (219655, 16)
y_train (219655,)
X_test (94138, 16)
y_test (94138,)
```

```
from sklearn.model_selection import KFold,cross_val_score
kfold=KFold(n_splits=10, shuffle=True, random_state=10)
```

```
LR = LogisticRegression()
```

```
LR.fit(X_train,y_train)
```

```
y_pred_LR=LR.predict(X_test)
accuracy_score(y_test,y_pred_LR)*100
```

```
37.94217000573626
```

```
print(classification_report(y_test,y_pred_LR))
```

```
              precision    recall  f1-score   support

           1       0.48      0.02      0.03      6901
           2       0.37      0.45      0.41     23205
           3       0.40      0.64      0.49     25792
           4       0.32      0.19      0.24     16289
           5       0.00      0.00      0.00      3439
           6       0.37      0.46      0.41     10470
           7       0.00      0.00      0.00       822
           8       0.00      0.00      0.00      3093
           9       0.11      0.00      0.01      1412
          10       0.00      0.00      0.00       782
          11       0.49      0.32      0.39      1933

    accuracy                           0.38     94138
   macro avg       0.23      0.19      0.18     94138
weighted avg       0.34      0.38      0.33     94138
```

```
decision_tree_classification = DecisionTreeClassifier(criterion =
'entropy', random_state = 10)

decision_tree = decision_tree_classification.fit(X_train, y_train)

y_pred_DT=decision_tree.predict(X_test)
accuracy_score(y_test,y_pred_DT)*100
```

```
29.678769466102956
```

```
print(classification_report(y_test,y_pred_DT))
```

```
              precision    recall  f1-score   support

           1       0.19      0.19      0.19      6901
           2       0.34      0.34      0.34     23205
           3       0.39      0.38      0.38     25792
           4       0.25      0.25      0.25     16289
           5       0.06      0.07      0.07      3439
           6       0.31      0.30      0.31     10470
           7       0.03      0.04      0.03       822
           8       0.15      0.15      0.15      3093
           9       0.22      0.23      0.22      1412
          10       0.08      0.10      0.09       782
          11       0.32      0.35      0.33      1933

    accuracy                           0.30     94138
   macro avg       0.21      0.22      0.21     94138
weighted avg       0.30      0.30      0.30     94138
```

```
dt_tuned = DecisionTreeClassifier(criterion = 'gini', max_depth=11,
random_state = 10)

decision_tree_tuned = dt_tuned.fit(X_train, y_train)

y_pred_DT_tuned = decision_tree_tuned.predict(X_test)
accuracy_score(y_test,y_pred_DT_tuned)*100
```

```
40.82198474579872
```

```
print(classification_report(y_test,y_pred_DT_tuned))
```

```
              precision    recall  f1-score   support

           1       0.36      0.13      0.19      6901
           2       0.41      0.46      0.44     23205
           3       0.41      0.67      0.51     25792
           4       0.39      0.23      0.29     16289
           5       0.07      0.00      0.00      3439
           6       0.41      0.45      0.43     10470
           7       0.00      0.00      0.00       822
           8       0.27      0.03      0.05      3093
           9       0.35      0.21      0.27      1412
          10       0.20      0.03      0.05       782
          11       0.52      0.36      0.43      1933

    accuracy                           0.41     94138
   macro avg       0.31      0.23      0.24     94138
weighted avg       0.38      0.41      0.37     94138
```

```
rf_classification = RandomForestClassifier(random_state = 10)
```

```
rf_model = rf_classification.fit(X_train, y_train)
```

```
y_pred_RF = rf_model.predict(X_test)
accuracy_score(y_test,y_pred_RF)*100
```

```
38.315026875438186
```

```
print(classification_report(y_test,y_pred_RF))
```

```
              precision    recall  f1-score   support

           1       0.30      0.19      0.23      6901
           2       0.39      0.44      0.42     23205
           3       0.41      0.54      0.47     25792
           4       0.33      0.27      0.29     16289
           5       0.09      0.02      0.04      3439
           6       0.40      0.44      0.42     10470
           7       0.10      0.02      0.03       822
           8       0.28      0.10      0.15      3093
           9       0.37      0.22      0.28      1412
          10       0.24      0.06      0.09       782
          11       0.52      0.44      0.48      1933

    accuracy                           0.38     94138
   macro avg       0.31      0.25      0.26     94138
weighted avg       0.36      0.38      0.37     94138
```

```
rf_classification_tuned = RandomForestClassifier(criterion = 'gini',
n_estimators = 47, random_state = 10)
```

```
rf_model_tuned = rf_classification_tuned.fit(X_train, y_train)
```

```
y_pred_RF_tuned = rf_model_tuned.predict(X_test)
accuracy_score(y_test,y_pred_RF_tuned)*100
```

```
37.94323227602031
```

```
print(classification_report(y_test,y_pred_RF_tuned))
```

```
              precision    recall  f1-score   support

           1       0.29      0.19      0.23      6901
           2       0.38      0.44      0.41     23205
           3       0.41      0.53      0.46     25792
           4       0.32      0.27      0.30     16289
           5       0.09      0.02      0.04      3439
           6       0.39      0.43      0.41     10470
           7       0.10      0.02      0.03       822
           8       0.28      0.11      0.15      3093
           9       0.36      0.22      0.27      1412
          10       0.22      0.05      0.08       782
          11       0.52      0.43      0.47      1933

    accuracy                           0.38     94138
   macro avg       0.31      0.25      0.26     94138
weighted avg       0.36      0.38      0.36     94138
```

```
dt_tuned = DecisionTreeClassifier(criterion = 'gini', max_depth=11,
random_state = 10)
ada_model_DT = AdaBoostClassifier(base_estimator=dt_tuned,
random_state = 10)
```

```
ada_model_DT.fit(X_train, y_train)
```

```
AdaBoostClassifier(base_estimator=DecisionTreeClassifier(max_depth=11,

random_state=10),
                   random_state=10)
```

```
y_pred_ada_model_DT = ada_model_DT.predict(X_test)
accuracy_score(y_test,y_pred_ada_model_DT)*100
```

```
29.91990482058255
```

```
print(classification_report(y_test,y_pred_ada_model_DT))
```

```
              precision    recall  f1-score   support

           1       0.19      0.16      0.17      6901
           2       0.33      0.34      0.33     23205
           3       0.38      0.39      0.39     25792
           4       0.22      0.27      0.24     16289
           5       0.06      0.04      0.05      3439
           6       0.31      0.36      0.33     10470
           7       0.00      0.00      0.00       822
           8       0.14      0.10      0.11      3093
           9       0.22      0.10      0.13      1412
          10       0.10      0.01      0.02       782
          11       0.45      0.20      0.27      1933
```

```
    accuracy                           0.30     94138
   macro avg       0.22      0.18      0.19     94138
weighted avg       0.29      0.30      0.29     94138
```

```
ada_model_DT_tuned = AdaBoostClassifier(base_estimator=dt_tuned,
n_estimators = 1, random_state = 10)
```

```
ada_model_DT_tuned.fit(X_train, y_train)
```

```
AdaBoostClassifier(base_estimator=DecisionTreeClassifier(max_depth=11,

random_state=10),
                   n_estimators=1, random_state=10)
```

```
y_pred_ada_model_DT_tuned = ada_model_DT.predict(X_test)
accuracy_score(y_test,y_pred_ada_model_DT_tuned)*100
```

```
29.91990482058255
```

```
print(classification_report(y_test,y_pred_ada_model_DT_tuned))
```

```
              precision    recall  f1-score   support

           1       0.19      0.16      0.17      6901
           2       0.33      0.34      0.33     23205
           3       0.38      0.39      0.39     25792
           4       0.22      0.27      0.24     16289
           5       0.06      0.04      0.05      3439
           6       0.31      0.36      0.33     10470
           7       0.00      0.00      0.00       822
           8       0.14      0.10      0.11      3093
           9       0.22      0.10      0.13      1412
          10       0.10      0.01      0.02       782
          11       0.45      0.20      0.27      1933

    accuracy                           0.30     94138
   macro avg       0.22      0.18      0.19     94138
weighted avg       0.29      0.30      0.29     94138
```

```
rf_classification_tuned = RandomForestClassifier(criterion = 'gini',
n_estimators = 47, random_state = 10)
```

```
ada_model_rf =
AdaBoostClassifier(base_estimator=rf_classification_tuned,
n_estimators=1, random_state = 10)
```

```
ada_model_rf.fit(X_train, y_train)
```

```
AdaBoostClassifier(base_estimator=RandomForestClassifier(n_estimators=
47,

random_state=10),
                   n_estimators=1, random_state=10)
```

```
y_pred_ada_model_RF = ada_model_rf.predict(X_test)
accuracy_score(y_test,y_pred_ada_model_RF)*100
```

```
38.081327412946955
```

```
print(classification_report(y_test,y_pred_ada_model_RF))
```

```
              precision    recall  f1-score   support

           1       0.29      0.20      0.23      6901
           2       0.39      0.44      0.41     23205
           3       0.41      0.53      0.46     25792
           4       0.32      0.27      0.30     16289
           5       0.10      0.03      0.04      3439
           6       0.40      0.43      0.41     10470
           7       0.12      0.02      0.03       822
           8       0.26      0.10      0.15      3093
           9       0.39      0.24      0.29      1412
          10       0.23      0.05      0.09       782
          11       0.51      0.44      0.47      1933

    accuracy                           0.38     94138
   macro avg       0.31      0.25      0.26     94138
weighted avg       0.36      0.38      0.36     94138
```

```
ada_model_rf_tuned =
AdaBoostClassifier(base_estimator=rf_classification_tuned,
n_estimators = 4, random_state = 10)
```

```
ada_model_rf_tuned.fit(X_train, y_train)
```

```
AdaBoostClassifier(base_estimator=RandomForestClassifier(n_estimators=
47,

random_state=10),
                   n_estimators=4, random_state=10)
```

```
from sklearn.ensemble import GradientBoostingClassifier
GB=GradientBoostingClassifier(random_state=10)
GB.fit(X_train, y_train)
```

```
GradientBoostingClassifier(random_state=10)
```

```
y_pred_GB = GB.predict(X_test)
accuracy_score(y_test,y_pred_GB)*100
```

```
41.5475153498056
```

```
print(classification_report(y_test,y_pred_GB))
```

```
              precision    recall  f1-score   support

           1       0.41      0.12      0.19      6901
           2       0.42      0.51      0.46     23205
           3       0.42      0.66      0.52     25792
           4       0.41      0.17      0.24     16289
           5       0.14      0.00      0.00      3439
           6       0.39      0.53      0.45     10470
           7       0.00      0.00      0.00       822
           8       0.30      0.01      0.02      3093
           9       0.31      0.20      0.24      1412
          10       0.17      0.01      0.01       782
          11       0.52      0.40      0.45      1933

    accuracy                           0.42     94138
   macro avg       0.32      0.24      0.23     94138
weighted avg       0.39      0.42      0.37     94138
```

```
GB_tuned=GradientBoostingClassifier(n_estimators=29, random_state=10)
GB_tuned.fit(X_train, y_train)
```

```
GradientBoostingClassifier(n_estimators=29, random_state=10)
```

```
from sklearn.naive_bayes import GaussianNB
NB = GaussianNB()
NB.fit(X_train,y_train)
```

```
GaussianNB()
```

```
y_pred_NB = NB.predict(X_test)
accuracy_score(y_test,y_pred_NB)*100
```

```
36.37850814761308
```

```
print(classification_report(y_test,y_pred_NB))
```

```
              precision    recall  f1-score   support

           1       0.30      0.09      0.14      6901
           2       0.36      0.41      0.39     23205
           3       0.39      0.65      0.49     25792
           4       0.32      0.15      0.21     16289
           5       0.08      0.01      0.01      3439
           6       0.33      0.38      0.36     10470
           7       0.04      0.00      0.00       822
           8       0.10      0.01      0.02      3093
           9       0.12      0.02      0.04      1412
```

```
          10       0.50      0.00      0.00       782
          11       0.42      0.37      0.39      1933

    accuracy                           0.36     94138
   macro avg       0.27      0.19      0.18     94138
weighted avg       0.33      0.36      0.32     94138
```

```
from sklearn.neighbors import KNeighborsClassifier
```

```
knn=KNeighborsClassifier(n_neighbors=565,weights='distance')
knn.fit(X_train,y_train)
```

```
KNeighborsClassifier(n_neighbors=565, weights='distance')
```

```
y_pred_KNN = NB.predict(X_test)
accuracy_score(y_test,y_pred_KNN)*100
```

```
36.37850814761308
```

```
print(classification_report(y_test,y_pred_NB))
```

```
              precision    recall  f1-score   support

           1       0.30      0.09      0.14      6901
           2       0.36      0.41      0.39     23205
           3       0.39      0.65      0.49     25792
           4       0.32      0.15      0.21     16289
           5       0.08      0.01      0.01      3439
           6       0.33      0.38      0.36     10470
           7       0.04      0.00      0.00       822
           8       0.10      0.01      0.02      3093
           9       0.12      0.02      0.04      1412
          10       0.50      0.00      0.00       782
          11       0.42      0.37      0.39      1933

    accuracy                           0.36     94138
   macro avg       0.27      0.19      0.18     94138
weighted avg       0.33      0.36      0.32     94138
```

```
from catboost import CatBoostClassifier
```

```
cb =
CatBoostClassifier(random_state=10,use_best_model=True,iterations=1000
)
cb.fit(X_train,y_train,use_best_model=True,verbose=100,eval_set=(X_tes
t,y_test))
```

```
Learning rate set to 0.120271
0:    learn: 2.1972797 test: 2.1978440  best: 2.1978440 (0)   total:
```

```
532ms remaining: 8m 50s
100:  learn: 1.5115382 test: 1.5242638  best: 1.5242638 (100) total:
37.5s remaining: 5m 34s
200:  learn: 1.4818453 test: 1.5070444  best: 1.5070444 (200) total: 1m
17s   remaining: 5m 9s
300:  learn: 1.4638516 test: 1.5016071  best: 1.5016071 (300) total: 1m
57s   remaining: 4m 32s
400:  learn: 1.4484799 test: 1.4983697  best: 1.4983697 (400) total: 2m
35s   remaining: 3m 52s
500:  learn: 1.4345747 test: 1.4970449  best: 1.4970449 (500) total: 3m
14s   remaining: 3m 13s
600:  learn: 1.4237650 test: 1.4965192  best: 1.4964737 (579) total: 3m
56s   remaining: 2m 37s
700:  learn: 1.4123374 test: 1.4963260  best: 1.4961744 (652) total: 4m
34s   remaining: 1m 57s
800:  learn: 1.4018164 test: 1.4964416  best: 1.4961744 (652) total: 5m
13s   remaining: 1m 17s
900:  learn: 1.3915056 test: 1.4967396  best: 1.4961744 (652) total: 5m
52s   remaining: 38.8s
999:  learn: 1.3815565 test: 1.4971529  best: 1.4961744 (652) total: 6m
37s   remaining: 0us

bestTest = 1.496174357
bestIteration = 652

Shrink model to first 653 iterations.
```

```
<catboost.core.CatBoostClassifier at 0x1f1be732bb0>
```

```
cb_pred = cb.predict(X_test)
accuracy_score(y_test,cb_pred)*100
```

```
42.54180033567741
```

```
print(classification_report(y_test,cb_pred))
```

```
              precision    recall  f1-score   support

           1       0.41      0.16      0.23      6901
           2       0.43      0.51      0.47     23205
           3       0.43      0.66      0.52     25792
           4       0.41      0.24      0.30     16289
           5       0.24      0.00      0.01      3439
           6       0.41      0.48      0.44     10470
           7       0.12      0.00      0.00       822
           8       0.42      0.03      0.05      3093
           9       0.36      0.21      0.27      1412
          10       0.29      0.01      0.02       782
          11       0.52      0.43      0.47      1933

    accuracy                           0.43     94138
```

```
   macro avg       0.37      0.25      0.25     94138
weighted avg       0.41      0.43      0.39     94138
```

```
cb_pred_train= cb.predict(X_train)
accuracy_score(y_train,cb_pred_train)*100
```

```
45.57692745441715
```

```
print(classification_report(y_train,cb_pred_train))
```

```
              precision    recall  f1-score   support

           1       0.46      0.19      0.27     16349
           2       0.45      0.54      0.49     53890
           3       0.45      0.68      0.54     60524
           4       0.47      0.26      0.34     38023
           5       0.63      0.01      0.02      8102
           6       0.45      0.53      0.48     23993
           7       0.79      0.01      0.02      1876
           8       0.71      0.06      0.10      7003
           9       0.55      0.31      0.40      3349
          10       0.84      0.07      0.12      1931
          11       0.62      0.51      0.56      4615

    accuracy                           0.46    219655
   macro avg       0.58      0.29      0.30    219655
weighted avg       0.48      0.46      0.42    219655
```

```
ls  =  df_full_test.columns.tolist()

in_data =  df_full_test[ls]

out_data =  cb.predict (in_data)

test = pd.read_csv("D:/HealthCare/test_data.csv")
submit = pd.DataFrame()

submit ['case_id']  =  test['case_id']
submit ['Stay']     =  out_data

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

submit ['Stay']     = submit ['Stay'].map(stay_decode)

submit.head(15)
```

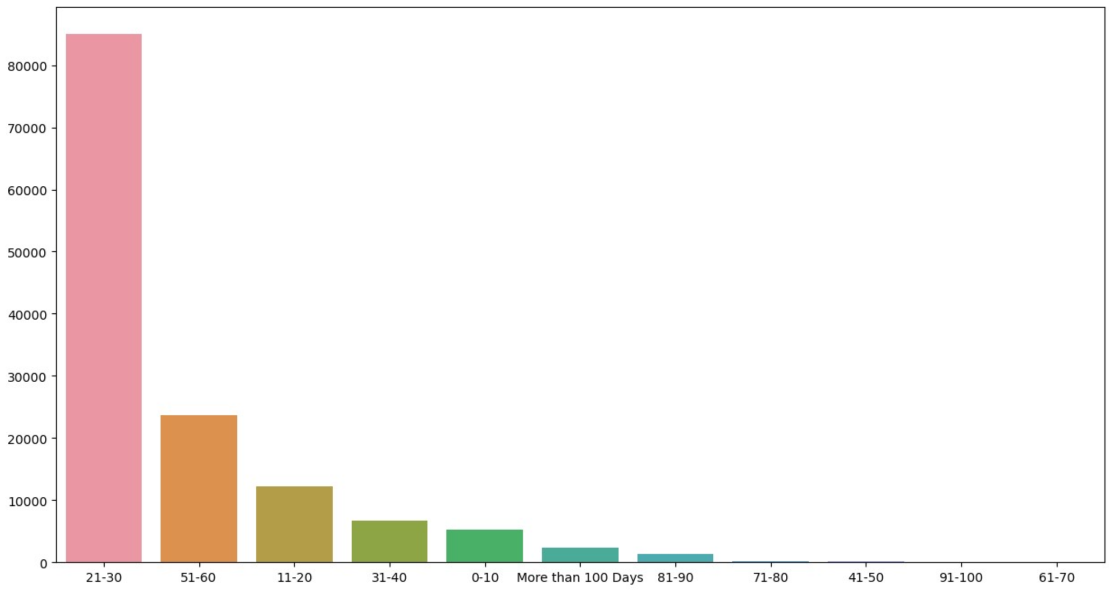
```
    case_id   Stay
0    318439   0-10
1    318440  51-60
2    318441  21-30
3    318442  21-30
4    318443  51-60
5    318444  21-30
6    318445  21-30
7    318446  21-30
8    318447  21-30
9    318448  21-30
10   318449  21-30
11   318450  51-60
12   318451  21-30
13   318452  21-30
14   318453  31-40
```

```
count = submit ['Stay'].value_counts()
count
```

```
21-30                 85076
51-60                 23685
11-20                 12285
31-40                  6693
0-10                   5242
More than 100 Days     2407
81-90                  1329
71-80                   174
41-50                   127
91-100                   32
61-70                     7
Name: Stay, dtype: int64
```

```
sns.barplot(x=count.index.values, y=count.values,data=submit)
```

```
<AxesSubplot:>
```



80000
70000
60000
50000
40000
30000
20000
10000
0
21-30
51-60
11-20
31-40
0-10
More than 100 Days
81-90
71-80
41-50
91-100
61-70