In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import statistics
```

In [2]:
```
train=pd.read_csv('C:/Users/Ranjan kumar/Downloads/healthcare/train_data.csv')
```

In [3]:
```
train
```

Out[3]:

	case_id	Hospital_code	Hospital_type_code	City_Code_Hospital	Hospital_region_code	Available Extra Rooms in Hospital	Department	Ward_Type	Ward_Facility_Code	Bed Grade	I
0	1	8	c	3	Z	3	radiotherapy	R	F	2.0	
1	2	2	c	5	Z	2	radiotherapy	S	F	2.0	
2	3	10	e	1	X	2	anesthesia	S	E	2.0	
3	4	26	b	2	Y	2	radiotherapy	R	D	2.0	
4	5	26	b	2	Y	2	radiotherapy	S	D	2.0	
...	...	...	...	...	...	...	...	...	...	...	
318433	318434	6	a	6	X	3	radiotherapy	Q	F	4.0	

In [10]:
```
train.isnull().sum()
```

Out[10]:
```
case_id                             0
Hospital_code                       0
Hospital_type_code                  0
City_Code_Hospital                  0
Hospital_region_code                0
Available Extra Rooms in Hospital   0
Department                          0
Ward_Type                           0
Ward_Facility_Code                  0
Bed Grade                           0
patientid                           0
City_Code_Patient                   0
Type of Admission                   0
Severity of Illness                 0
Visitors with Patient               0
Age                                 0
Admission_Deposit                   0
Stay                                0
dtype: int64
```

In [11]:
```
train.drop(['case_id', 'patientid'], axis=1, inplace=True)
```

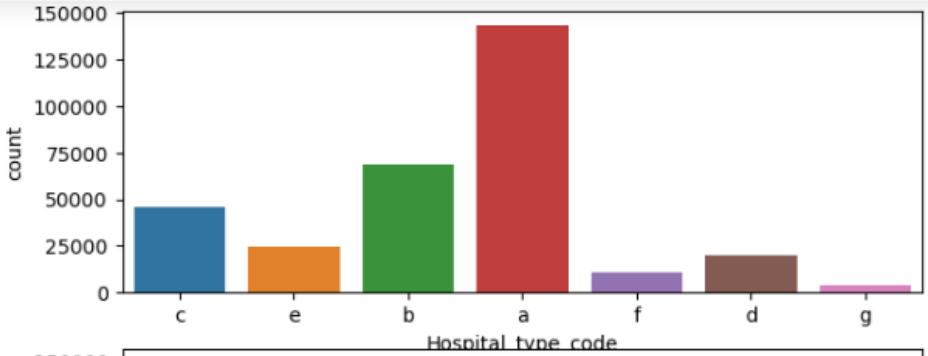
In [12]:
```
category=[]
numer=[]

for col in train.columns:
    if train[col].dtypes=='object':
        category.append(col)

for col in train.columns:
    if train[col].dtypes!='object':
        numer.append(col)

print(category)
```

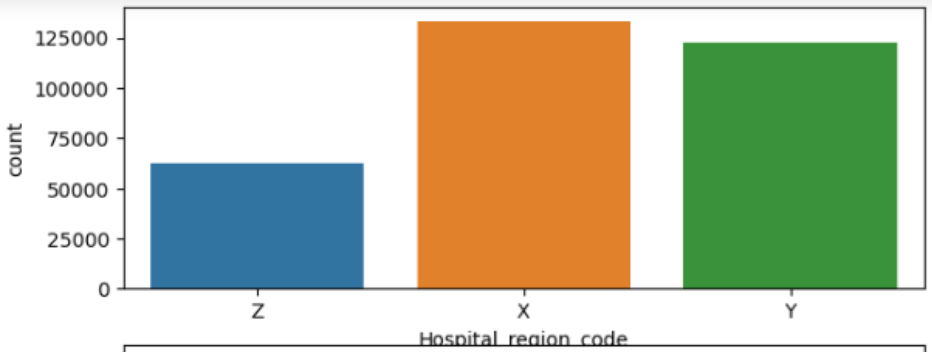
In [12]:
```
category=[]
numer=[]

for col in train.columns:
    if train[col].dtypes=='object':
        category.append(col)

for col in train.columns:
    if train[col].dtypes!='object':
        numer.append(col)

print(category)
print(numer)
```

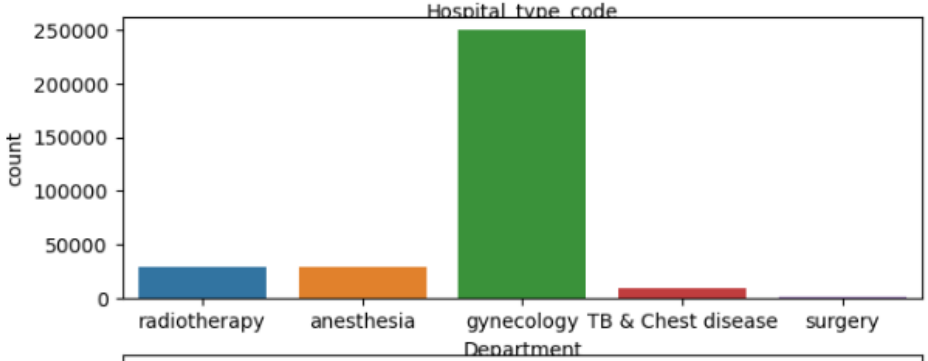
```
['Hospital_type_code', 'Hospital_region_code', 'Department', 'Ward_Type', 'Ward_Facility_Code', 'Type of Admission', 'Severity
of Illness', 'Age', 'Stay']
['Hospital_code', 'City_Code_Hospital', 'Available Extra Rooms in Hospital', 'Bed Grade', 'City_Code_Patient', 'Visitors with P
atient', 'Admission_Deposit']
```

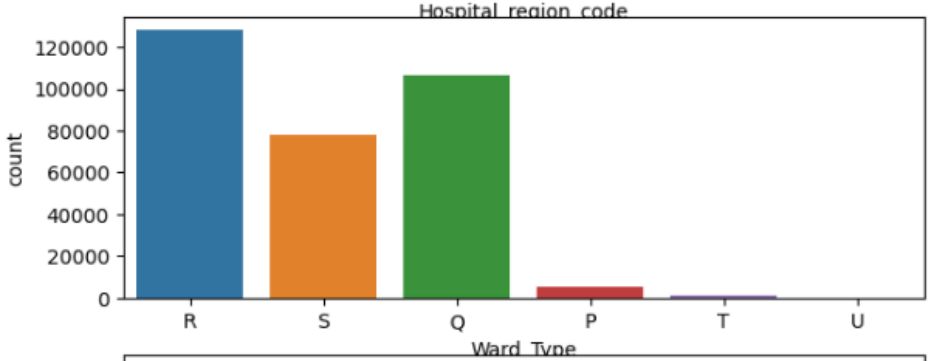
In [13]:
```
i=1
plt.figure(figsize=(16,18))
for col in category:
    plt.subplot(6,2,i)
    sns.countplot(train[col])
    i=i+1
plt.show()
```

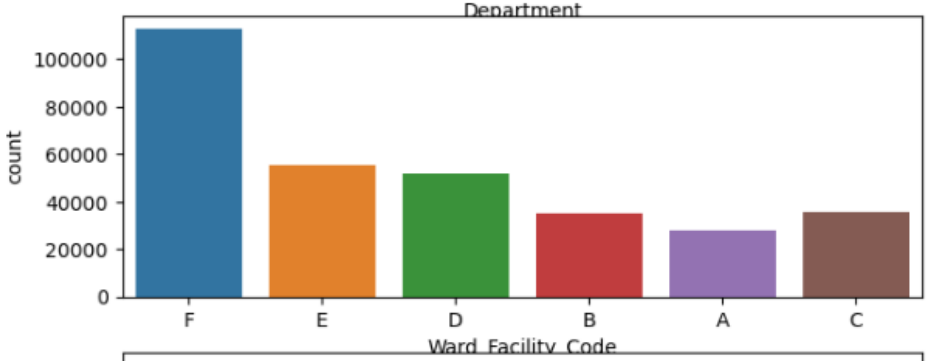
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword a
rg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit ke
yword will result in an error or misinterpretation.
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword a
rg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit ke
yword will result in an error or misinterpretation.
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword a
rg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit ke
```

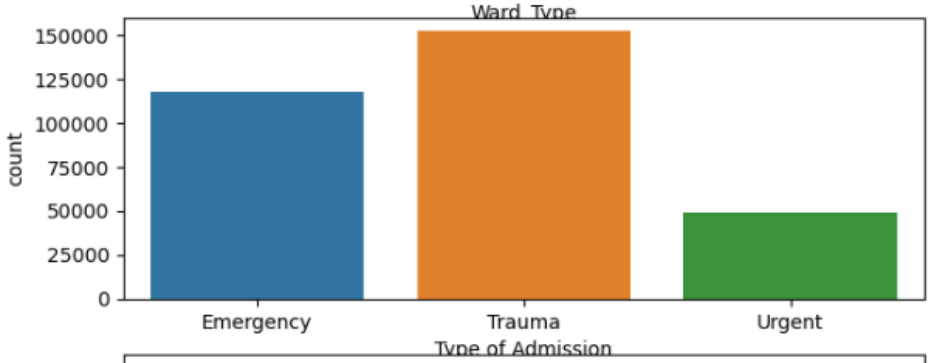

150000
125000
100000
75000
50000
25000
0
count
c
e
b
a
f
d
g
Hospital_type_code

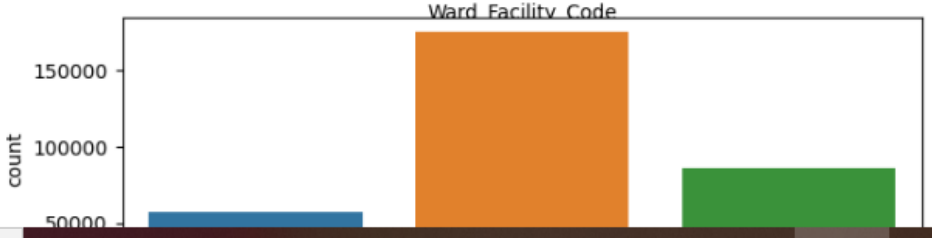

125000
100000
75000
50000
25000
0
count
Z
X
Y
Hospital_region_code

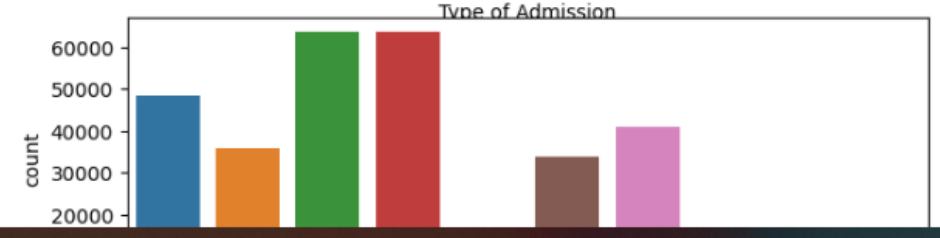

250000
200000
150000
100000
50000
0
count
radiotherapy
anesthesia
gynecology
TB & Chest disease
surgery
Department

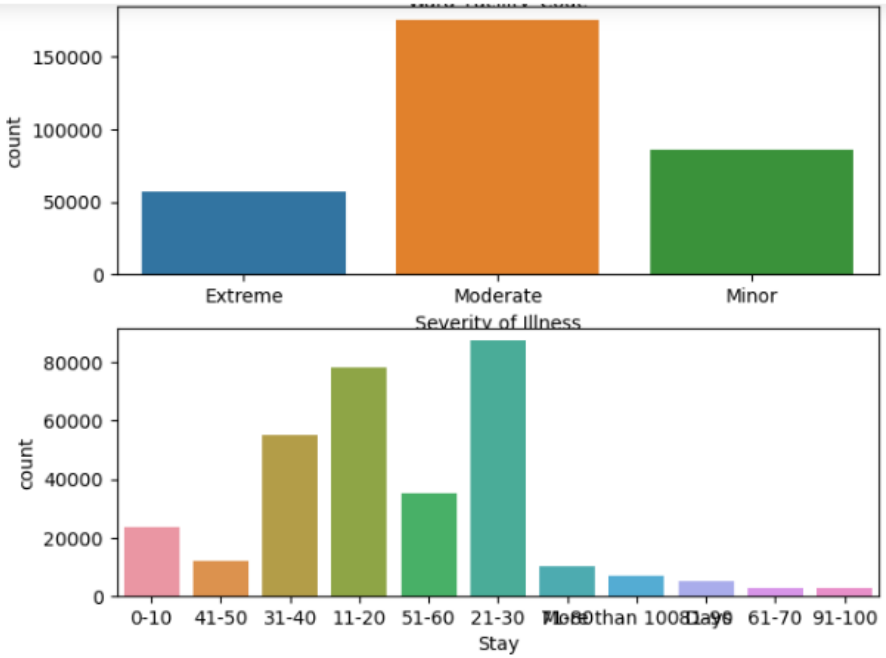

120000
100000
80000
60000
40000
20000
0
count
R
S
Q
P
T
U
Ward_Type

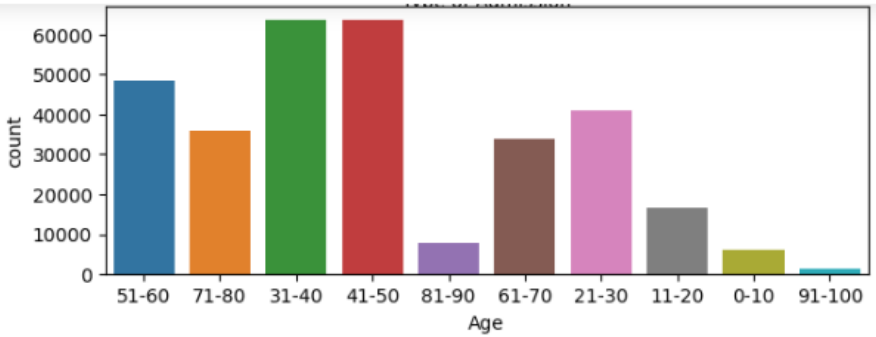

100000
80000
60000
40000
20000
0
count
F
E
D
B
A
C
Ward_Facility_Code


150000
125000
100000
75000
50000
25000
0
count
Emergency
Trauma
Urgent
Type of Admission


150000
100000
50000
count


60000
50000
40000
30000
20000
count





In [14]:
```
train['Stay'].value_counts()
```

Out[14]:
```
21-30                 87491
11-20                 78139
31-40                 55159
51-60                 35018
0-10                  23604
41-50                 11743
71-80                 10254
More than 100 Days     6683
81-90                  4838
91-100                 2765
61-70                  2744
Name: Stay, dtype: int64
```

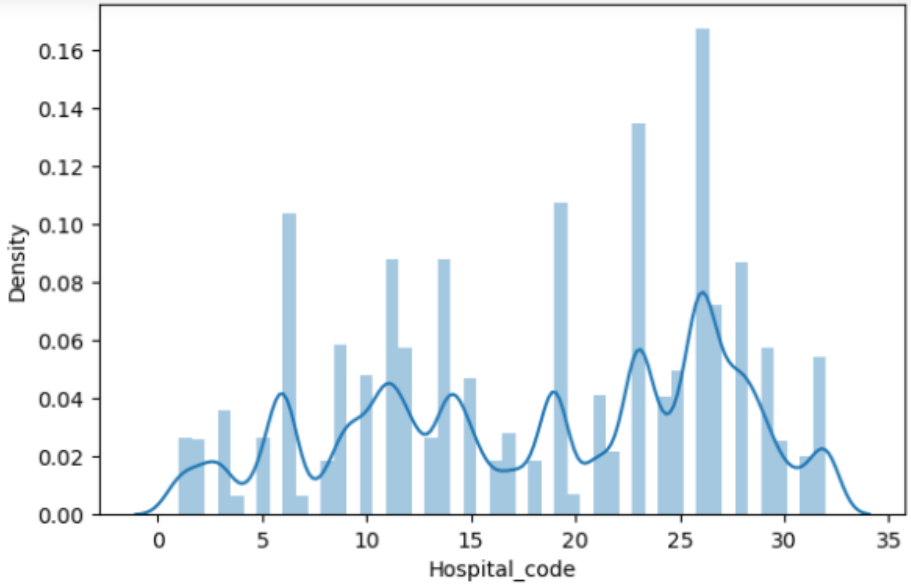
In [15]:
```
train['Stay'].replace('More than 100 Days', '>100', inplace=True)
```

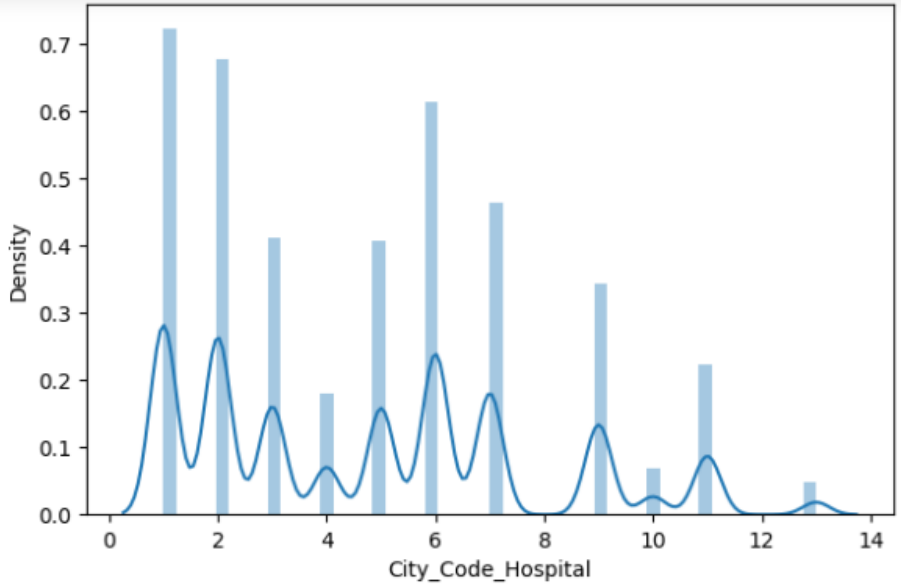
In [16]:
```
train['Stay'].value_counts()
```

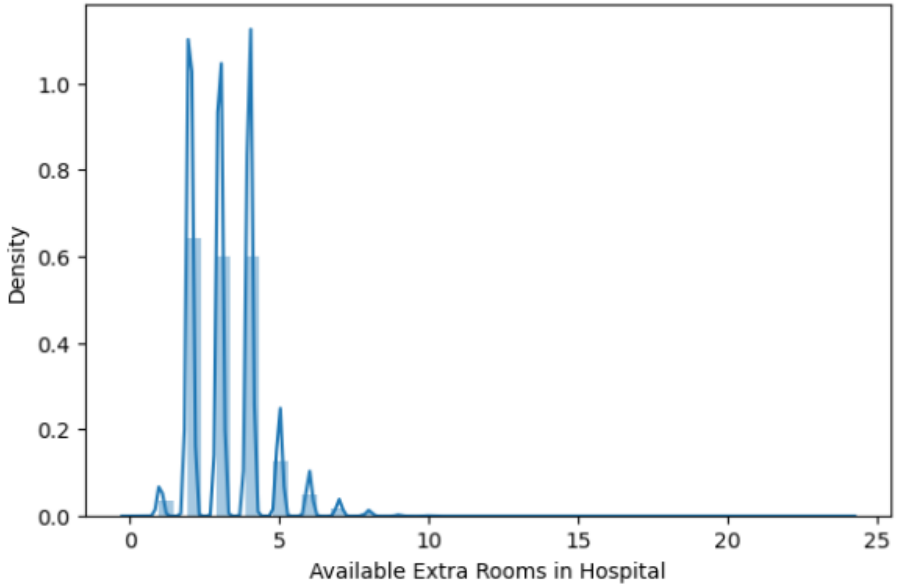
Out[16]:
```
21-30     87491
11-20     78139
31-40     55159
51-60     35018
0-10      23604
41-50     11743
71-80     10254
>100       6683
81-90      4838
91-100     2765
61-70      2744
Name: Stay, dtype: int64
```

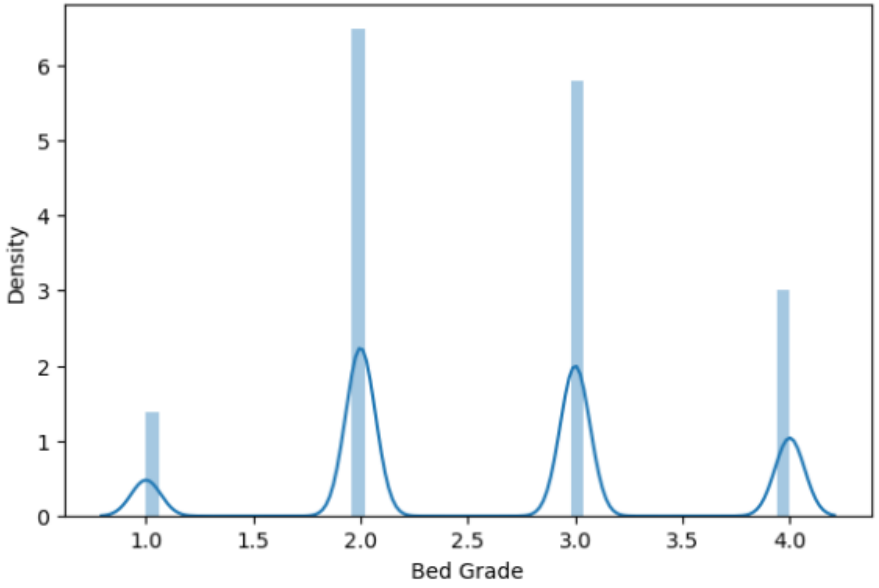
In [17]:
```
i=1
plt.figure(figsize=(15,20))
for col in numer:
    plt.subplot(4,2,i)
    sns.distplot(train[col])
    i=i+1

plt.show()
```

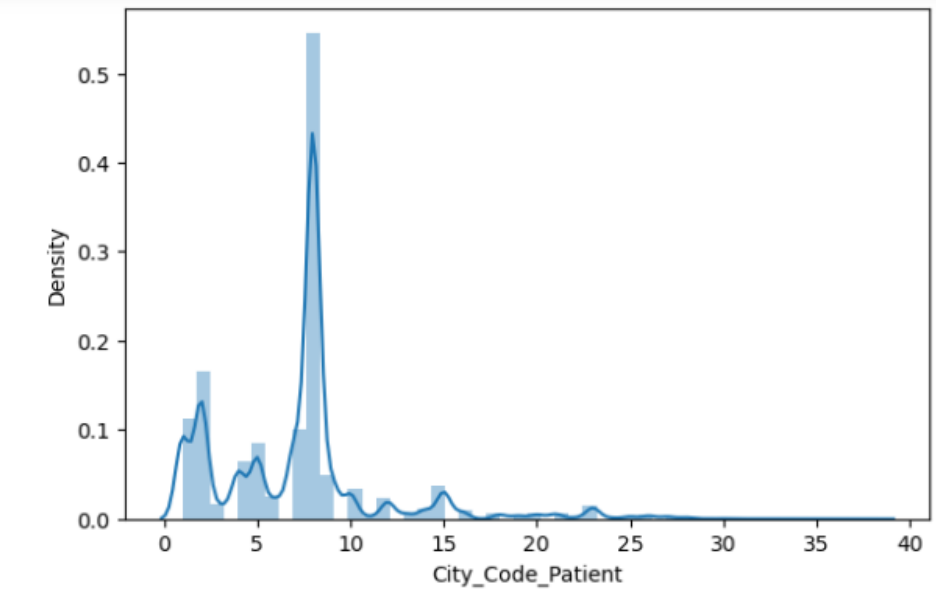
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function an
d will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar fle
xibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function an
d will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar fle
xibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function an
d will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar fle
```

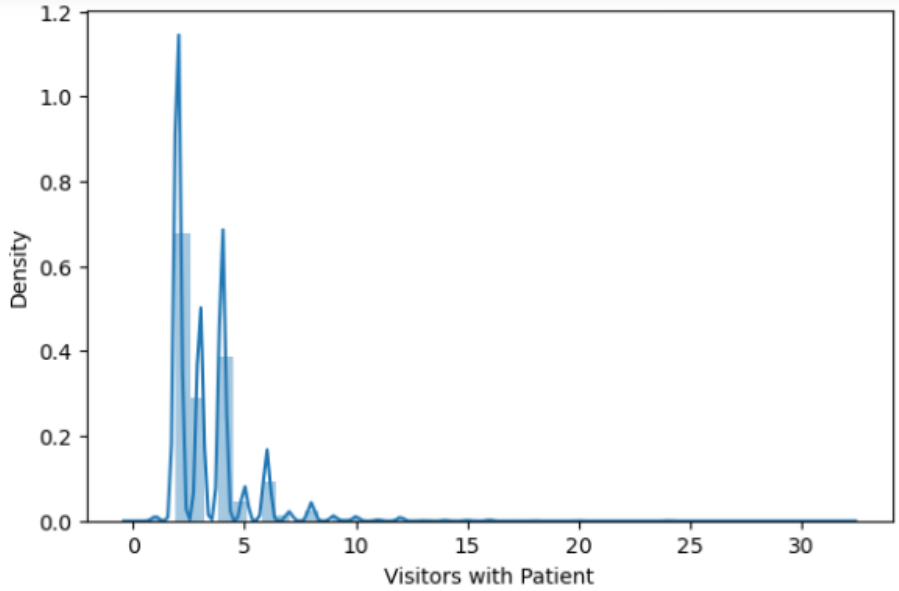

Density
Hospital_code

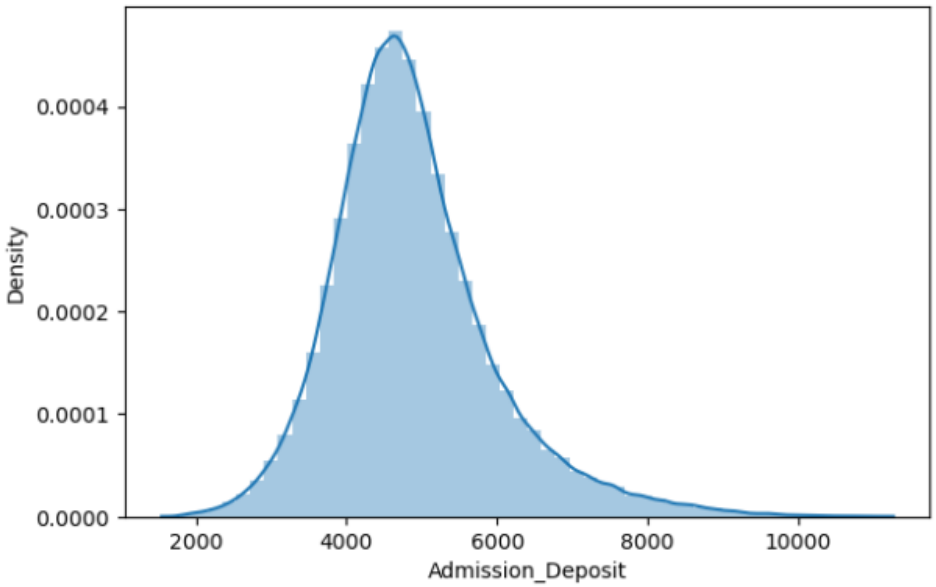

Density
City_Code_Hospital


Density
Available Extra Rooms in Hospital


Density
Bed Grade


Density
City_Code_Patient
0.0
0.1
0.2
0.3
0.4
0.5
0
5
10
15
20
25
30
35
40


Density
Visitors with Patient
0.0
0.2
0.4
0.6
0.8
1.0
1.2
0
5
10
15
20
25
30


Density
Admission_Deposit
0.0000
0.0001
0.0002
0.0003
0.0004
2000
4000
6000
8000
10000

In [19]:
```
df_beds=train[['Hospital_code','Available Extra Rooms in Hospital','Hospital_region_code']]
df_beds['Hospital_region_code']=df_beds['Hospital_region_code'].map({'X':1,'Y':2,'Z':3})
df_beds_grouped=df_beds.groupby('Hospital_code')[['Available Extra Rooms in Hospital','Hospital_region_code']].median().reset_inc
```

```
C:\Users\Ranjan kumar\AppData\Local\Temp\ipykernel_9944\3549231326.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_beds['Hospital_region_code']=df_beds['Hospital_region_code'].map({'X':1,'Y':2,'Z':3})
```

In [20]:
```
df_beds_grouped['Hospital_region_code'].map({1:'X',2:'Y',3:'Z'})
```

Out[20]:
```
0     Y
1     Z
2     Z
3     X
4     X
5     X
6     X
7     Z
8     Z
9     X
10    Y
11    Y
12    Z
13    X
14    Z
15    Z
16    X
17    Y
18    Y
19    Y
20    Z
```

jupyter Health Last Checkpoint: 16 hours ago (autosaved)

Logout

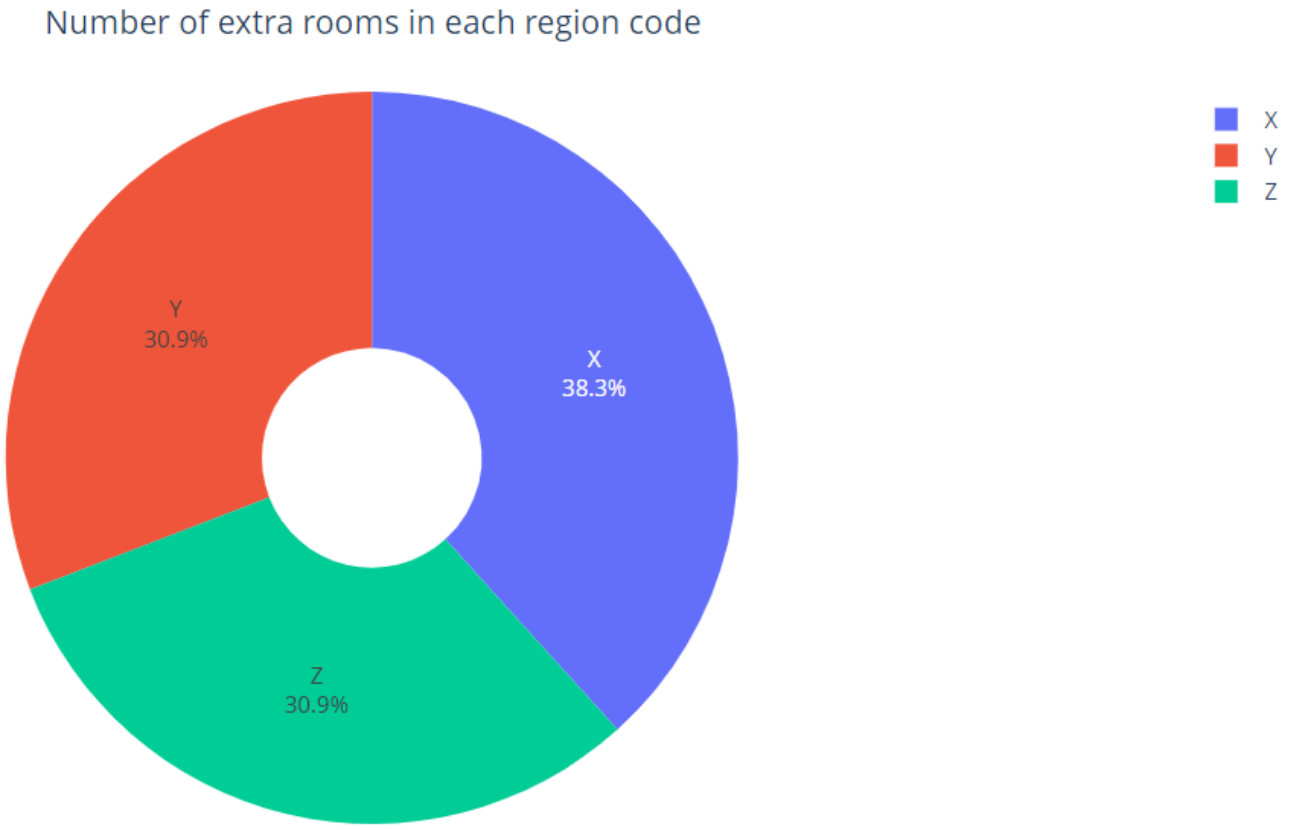
File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3 (ipykernel)

Run Code

In [22]:
```
import plotly.express as px
```

In [23]:
```
df_beds_1=df_beds_grouped.groupby('Hospital_region_code')['Available Extra Rooms in Hospital'].sum().reset_index()
fig=px.pie(df_beds_1,values='Available Extra Rooms in Hospital',names='Hospital_region_code',hole=0.3)
fig.update_layout(title='Number of extra rooms in each region code',title_x=0.5)
fig.update_traces(textinfo='percent+label')
```



jupyter Health Last Checkpoint: 16 hours ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

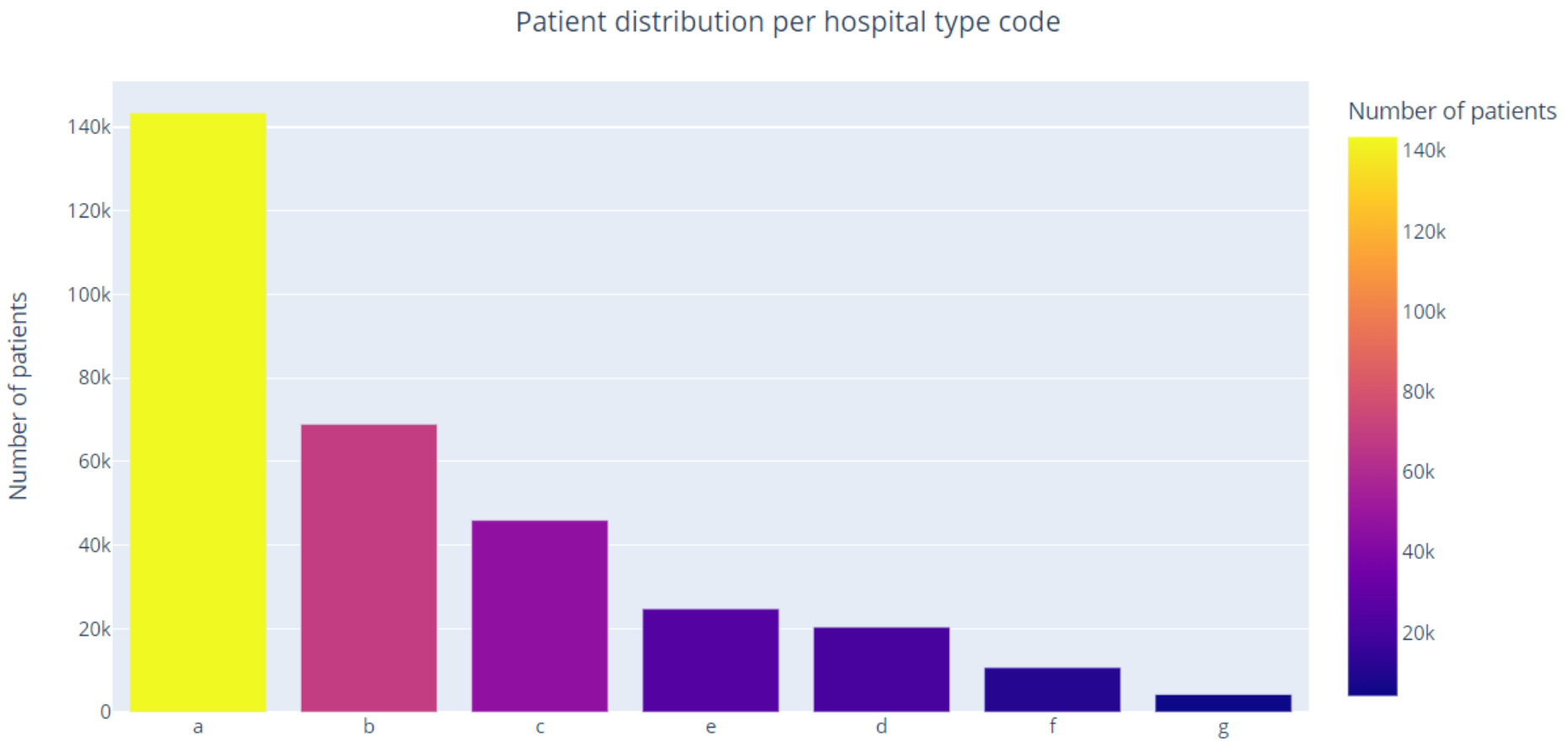
Trusted Python 3 (ipykernel)

Run Code

In [24]:
```
train['train_flag']=1
```

In [25]:
```
train_index=train.groupby('Hospital_type_code')['train_flag'].sum().reset_index().sort_values(by='train_flag',ascending=False)
```

In [26]:
```
n_index,x='Hospital_type_code',y='train_flag',color='train_flag',labels={'Hospital_type_code':'Hospital Type Code','train_flag':'
ut(title='Patient distribution per hospital type code',title_x=0.5)
```



jupyter **Health** Last Checkpoint: 16 hours ago (autosaved)

Logout

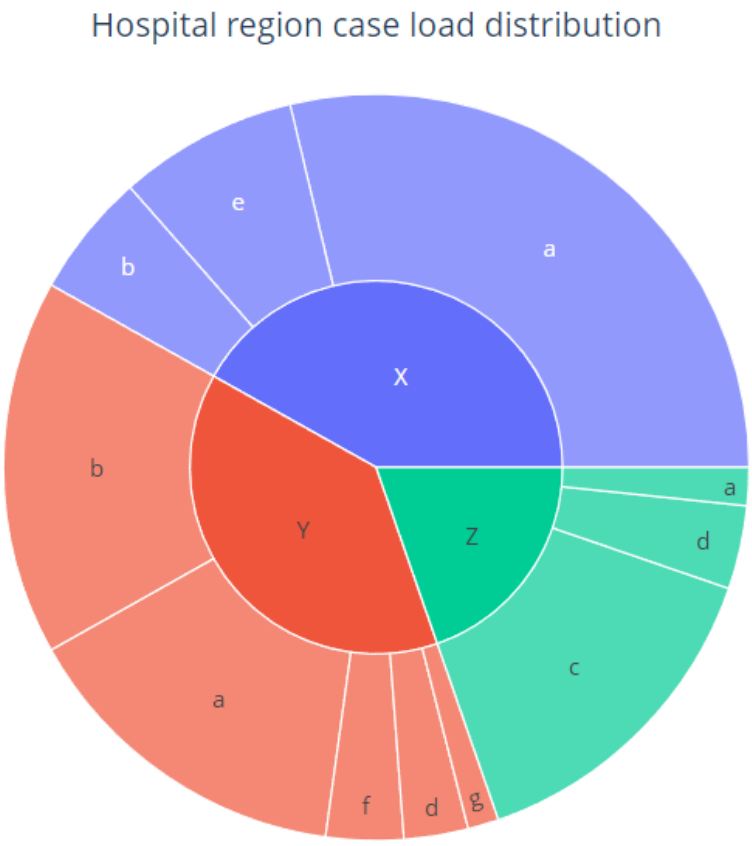
File Edit View Insert Cell Kernel Widgets Help

Trusted

Python 3 (ipykernel)

Run Code

In [27]:
```
fig3=px.sunburst(train,path=['Hospital_region_code','Hospital_type_code'])
fig3.update_layout(title='Hospital region case load distribution',title_x=0.5)
fig3.show()
```



```
91-100     2765
61-70      2744
Name: Stay, dtype: int64
```

In [29]:
```
from sklearn.preprocessing import LabelEncoder
encoder= LabelEncoder()
for col in category:
    train[col]= encoder.fit_transform(train[col])
```

In [30]:
```
train[category]
```

Out[30]:

	Hospital_type_code	Hospital_region_code	Department	Ward_Type	Ward_Facility_Code	Type of Admission	Severity of Illness	Age	Stay
0	2	2	3	2	5	0	0	5	0
1	2	2	3	3	5	1	0	5	4
2	4	0	1	3	4	1	0	5	3
3	1	1	3	2	3	1	0	5	4
4	1	1	3	3	3	1	0	5	4
...	...	...	...	...	...	...	...	...	...
318433	0	0	3	1	5	0	2	4	1
318434	0	0	1	1	4	2	2	8	3
318435	0	0	2	2	5	0	1	7	1
318436	1	1	1	1	3	1	1	1	1
318437	0	1	2	1	2	0	1	1	0

318438 rows × 9 columns

In [31]:
```
train['City_Code_Hospital'].value_counts()
```

Out[31]:
```
1     55351
2     51809
```

In [32]:
```
from sklearn.preprocessing import StandardScaler

scaler= StandardScaler()

train[numer]= scaler.fit_transform(train[numer].values)
```

In [33]:
```
train
```

Out[33]:

	Hospital_code	Hospital_type_code	City_Code_Hospital	Hospital_region_code	Available Extra Rooms in Hospital	Department	Ward_Type	Ward_Facility_Code	Bed Grade	City_C
0	-1.195176	2	-0.571055	2	-0.169177	3	2	5	-0.716535	
1	-1.890124	2	0.073580	2	-1.025217	3	3	5	-0.716535	
2	-0.963527	4	-1.215691	0	-1.025217	1	3	4	-0.716535	
3	0.889668	1	-0.893373	1	-1.025217	3	2	3	-0.716535	
4	0.889668	1	-0.893373	1	-1.025217	3	3	3	-0.716535	
...	...	...	...	...	...	...	...	...	...	
318433	-1.426825	0	0.395897	0	-0.169177	3	1	5	1.574234	
318434	0.658018	0	-1.215691	0	-1.025217	1	1	4	1.574234	
318435	-1.311001	0	-0.248738	0	-0.169177	2	2	5	1.574234	
318436	-0.847702	1	-0.893373	1	-0.169177	1	1	3	0.428849	
318437	0.078895	0	0.718215	1	1.542903	2	1	2	-0.716535	

318438 rows × 17 columns

In [34]:
```
from sklearn.model_selection import train_test_split
y= train['Stay']
```

In [34]:
```
from sklearn.model_selection import train_test_split
y= train['Stay']
X= train.drop('Stay', axis=1)
```

In [35]:
```
X
```

Out[35]:

	Hospital_code	Hospital_type_code	City_Code_Hospital	Hospital_region_code	Available Extra Rooms in Hospital	Department	Ward_Type	Ward_Facility_Code	Bed Grade	City_Code_P
0	-1.195176	2	-0.571055	2	-0.169177	3	2	5	-0.716535	-0.0
1	-1.890124	2	0.073580	2	-1.025217	3	3	5	-0.716535	-0.0
2	-0.963527	4	-1.215691	0	-1.025217	1	3	4	-0.716535	-0.0
3	0.889668	1	-0.893373	1	-1.025217	3	2	3	-0.716535	-0.0
4	0.889668	1	-0.893373	1	-1.025217	3	3	3	-0.716535	-0.0
...	...	...	...	...	...	...	...	...	...	
133	-1.426825	0	0.395897	0	-0.169177	3	1	5	1.574234	3.3
134	0.658018	0	-1.215691	0	-1.025217	1	1	4	1.574234	0.1
135	-1.311001	0	-0.248738	0	-0.169177	2	2	5	1.574234	0.5
136	-0.847702	1	-0.893373	1	-0.169177	1	1	3	0.428849	0.1
137	0.078895	0	0.718215	1	1.542903	2	1	2	-0.716535	0.1

38 rows × 16 columns

In [ ]:
```
X_train, X_test, y_train,y_test= train_test_split(X,y,test_size= 0.2, stratify=y, random_state=42)
```

34	0.658018	0	-1.215691	0	-1.025217	1	1	4	1.574234	0.1
35	-1.311001	0	-0.248738	0	-0.169177	2	2	5	1.574234	0.5
36	-0.847702	1	-0.893373	1	-0.169177	1	1	3	0.428849	0.1
37	0.078895	0	0.718215	1	1.542903	2	1	2	-0.716535	0.1

38 rows × 16 columns

In [ ]:
```
X_train, X_test, y_train,y_test= train_test_split(X,y,test_size= 0.2, stratify=y, random_state=42)
```

In [ ]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
```

In [40]:
```
reg = LogisticRegression()
reg.fit(X_train,y_train)

y_pred=reg.predict(X_test)
accuracy_score(y_test,y_pred)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning:

lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
```

Out[40]:
```
0.3780932043713101
```

In [42]:
```
print (classification_report(y_test,y_pred))
```

```
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
```

Out[40]: 0.3780932043713101

In [42]:
```
print (classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.44      0.03      0.06      4721
           1       0.37      0.40      0.38     15628
           2       0.40      0.68      0.50     17498
           3       0.33      0.18      0.23     11032
           4       0.00      0.00      0.00      2349
           5       0.35      0.49      0.41      7004
           6       0.00      0.00      0.00       549
           7       0.00      0.00      0.00      2051
           8       0.00      0.00      0.00       967
           9       0.00      0.00      0.00       553
          10       0.54      0.29      0.38      1336

    accuracy                           0.38     63688
   macro avg       0.22      0.19      0.18     63688
weighted avg       0.34      0.38      0.33     63688
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning:

Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter t
o control this behavior.

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning:
```

In [43]:
```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import confusion_matrix, classification_report, make_scorer, accuracy_score
from sklearn.model_selection import cross_val_score, train_test_split
```

In [44]:
```
ada_classifier = AdaBoostClassifier(n_estimators = 19, random_state = 10)
ada_classifier.fit(X_train, y_train)
pred_ada = ada_classifier.predict(X_test)
accuracy_score(y_test,pred_ada)
```

Out[44]: 0.3595182765984173

In [45]:
```
print (classification_report(y_test, pred_ada))
```

```
              precision    recall  f1-score   support

           0       0.23      0.04      0.07      4721
           1       0.33      0.32      0.33     15628
           2       0.39      0.68      0.49     17498
           3       0.32      0.19      0.24     11032
           4       0.00      0.00      0.00      2349
           5       0.35      0.47      0.40      7004
           6       0.00      0.00      0.00       549
           7       0.00      0.00      0.00      2051
           8       0.16      0.02      0.04       967
           9       0.00      0.00      0.00       553
          10       0.47      0.30      0.37      1336

    accuracy                           0.36     63688
   macro avg       0.20      0.18      0.18     63688
weighted avg       0.31      0.36      0.31     63688
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning:

Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter t
o control this behavior.
```

In [46]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn import tree
from sklearn.model_selection import GridSearchCV
```

In [47]:
```
deci= DecisionTreeClassifier(criterion = 'gini', max_depth=10, random_state = 42)

decision_tree = deci.fit(X_train, y_train)
```

In [48]:
```
y_preddd = decision_tree.predict(X_test)
accuracy_score(y_test,y_preddd)
```

Out[48]:
```
0.4109879412134154
```

In [49]:
```
print(accuracy_score(y_test,y_preddd))
```

```
0.4109879412134154
```

In [50]:
```
print (classification_report(y_test,y_preddd ))
```

```
              precision    recall  f1-score   support

           0       0.35      0.13      0.19      4721
           1       0.42      0.46      0.44     15628
           2       0.41      0.67      0.51     17498
           3       0.41      0.25      0.31     11032
           4       0.16      0.00      0.01      2349
           5       0.41      0.45      0.43      7004
           6       0.00      0.00      0.00       549
           7       0.31      0.02      0.04      2051
           8       0.35      0.20      0.25       967
           9       0.18      0.01      0.02       553
          10       0.54      0.39      0.45      1336

    accuracy                           0.41     63688
   macro avg       0.32      0.23      0.24     63688
weighted avg       0.39      0.41      0.38     63688
```

In [54]:
```
Random = RandomForestClassifier(criterion = 'entropy', n_estimators = 61, random_state = 142)

random_model =Random.fit(X_train, y_train)
```

In [55]:
```
y_predddd = random_model.predict(X_test)
accuracy_score(y_test,y_predddd)
```

Out[55]: 0.38200288908428587

In [56]:
```
print (classification_report(y_test,y_predddd ))
```

```
              precision    recall  f1-score   support

           0       0.28      0.19      0.22      4721
           1       0.39      0.44      0.41     15628
           2       0.41      0.54      0.47     17498
           3       0.34      0.27      0.30     11032
           4       0.08      0.02      0.03      2349
           5       0.39      0.44      0.41      7004
           6       0.09      0.02      0.03       549
           7       0.27      0.10      0.15      2051
           8       0.37      0.22      0.27       967
           9       0.27      0.05      0.09       553
          10       0.54      0.46      0.49      1336

    accuracy                           0.38     63688
   macro avg       0.31      0.25      0.26     63688
weighted avg       0.36      0.38      0.37     63688
```

In [57]:
```
import xgboost as xgb
```

In [58]:
```
xg=xgb.XGBClassifier(max_depth=11, objective='multi:softmax', n_estimators=59,random_state=42)
xg.fit(X_train,y_train)
xgbpred=xgbcl.predict(X_test)
```

jupyter Health Last Checkpoint: 16 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted

Run Code

In [54]:
```
Random = RandomForestClassifier(criterion = 'entropy', n_estimators = 61, random_state = 142)

random_model =Random.fit(X_train, y_train)
```

In [55]:
```
y_predddd = random_model.predict(X_test)
accuracy_score(y_test,y_predddd)
```

Out[55]: 0.38200288908428587

In [56]:
```
print (classification_report(y_test,y_predddd ))
```

	precision	recall	f1-score	support
0	0.28	0.19	0.22	4721
1	0.39	0.44	0.41	15628
2	0.41	0.54	0.47	17498
3	0.34	0.27	0.30	11032
4	0.08	0.02	0.03	2349
5	0.39	0.44	0.41	7004
6	0.09	0.02	0.03	549
7	0.27	0.10	0.15	2051
8	0.37	0.22	0.27	967
9	0.27	0.05	0.09	553
10	0.54	0.46	0.49	1336
accuracy			0.38	63688
macro avg	0.31	0.25	0.26	63688
weighted avg	0.36	0.38	0.37	63688

In [57]:
```
import xgboost as xgb
```

In [58]:
```
xg=xgb.XGBClassifier(max_depth=11, objective='multi:softmax', n_estimators=59,random_state=42)
xg.fit(X_train,y_train)
xgbpred=xgbcl.predict(X_test)
```

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import statistics
```

In [2]:
```
train=pd.read_csv('C:/Users/Ranjan kumar/Downloads/healthcare/train_data.csv')
```

In [3]:
```
train.head(5)
```

Out[3]:

	case_id	Hospital_code	Hospital_type_code	City_Code_Hospital	Hospital_region_code	Available Extra Rooms in Hospital	Department	Ward_Type	Ward_Facility_Code	Bed Grade	patien
0	1	8	c	3	Z	3	radiotherapy	R	F	2.0	313
1	2	2	c	5	Z	2	radiotherapy	S	F	2.0	313
2	3	10	e	1	X	2	anesthesia	S	E	2.0	313
3	4	26	b	2	Y	2	radiotherapy	R	D	2.0	313
4	5	26	b	2	Y	2	radiotherapy	S	D	2.0	313

In [4]:
```
test=pd.read_csv('C:/Users/Ranjan kumar/Downloads/healthcare/test_data.csv')
```

In [5]:
```
train.describe()
```

Out[5]:

e	1	X	2	anesthesia	S	E	2.0	31397	7.0	Trauma	Extreme
b	2	Y	2	radiotherapy	R	D	2.0	31397	7.0	Trauma	Extreme
b	2	Y	2	radiotherapy	S	D	2.0	31397	7.0	Trauma	Extreme

In [4]:
```
test=pd.read_csv('C:/Users/Ranjan kumar/Downloads/healthcare/test_data.csv')
```

In [5]:
```
test.head(5)
```

Out[5]:

	case_id	Hospital_code	Hospital_type_code	City_Code_Hospital	Hospital_region_code	Available Extra Rooms in Hospital	Department	Ward_Type	Ward_Facility_Code	Bed Grade	patient
0	318439	21	c	3	Z	3	gynecology	S	A	2.0	170(
1	318440	29	a	4	X	2	gynecology	S	F	2.0	170(
2	318441	26	b	2	Y	3	gynecology	Q	D	4.0	170(
3	318442	6	a	6	X	3	gynecology	Q	F	2.0	170(
4	318443	28	b	11	X	2	gynecology	R	F	2.0	170(

```
weighted avg       0.39      0.41      0.38     63688
```

In [*]:
```
Random = RandomForestClassifier(criterion = 'entropy', n_estimators = 61, random_state = 142)

random_model =Random.fit(X_train, y_train)
```

In [*]:
```
y_predddd = random_model.predict(X_test)
accuracy_score(y_test,y_predddd)
```

In [*]:
```
print (classification_report(y_test,y_predddd ))
```

In [*]:
```
import xgboost as xgb
```

In [*]:
```
xg=xgb.XGBClassifier(max_depth=11, objective='multi:softmax', n_estimators=59,random_state=42)
xgbcl=xg.fit(X_train,y_train)
xgbpred=xgbcl.predict(X_test)
```

In [*]:
```
accuracy_score(y_test,xgbpred)
```

In [*]:
```
print (classification_report(y_test,xgbpred ))
```

In [ ]:

In [ ]: