1. Importing Libraries

```
import pandas as pd
import numpy as np
import seaborn as sns

from sklearn.preprocessing import LabelEncoder

import warnings
warnings.filterwarnings('ignore')

import matplotlib.pyplot as plt
import seaborn as sns

pd.set_option("display.max_rows", None, "display.max_columns", None)
```

2. Importing datasets

```
train = pd.read_csv("SPARCS_194_2016_SelectedHospitals.csv", parse_dates=['Discharge Year'])
```

User Defined Functions

```
def quality_report(df):
    """
    Description: Displays quality of data in terms of missing values,
    unique numbers, datatypes etc.

    Arguments: Dataframe
    """
    dtypes = df.dtypes
    nuniq = df.T.apply(lambda x: x.nunique(), axis=1)
    total = df.isnull().sum().sort_values(ascending = False)
    percent = (df.isnull().sum()*100/df.isnull().count()).sort_values(ascending = False)
    quality_df  = pd.concat([total, percent, nuniq, dtypes], axis=1, keys=['Total NaN', 'Percent of NaN','Nunique…
    display(quality_df)


def object_count_plot(df):
    """Description : Plot countplot for all categorical features present in the dataframe passed Argument : Dat…
    for var in df.columns:
        if df[var].dtype == 'object':
            print(df[var].value_counts())
            plt.figure(figsize=(12,5))
            g = sns.countplot(x=var,data=df)
            g.set_xticklabels(g.get_xticklabels(), rotation=90, ha="right")
            plt.tight_layout()
            plt.show()


def numeric_distribution_plot(df):
    """Description : Gives distribution plot for all the numeric featuresin the dataframe passedArgument : Data…
    for col in df.columns:
        if df[col].dtype != 'object':
            print(df[col].describe())
            plt.figure(figsize=(12,5))
            plt.title("Distribution of "+col)
            ax = sns.distplot(df[col].dropna())
            plt.show()
            value_counts = df[col].value_counts().rename_axis("unique_"+col).to_frame('counts').sort_values('co…
            print(value_counts.head(5),'\n',value_counts.tail(5))
```

Visualising the data set

```
train.head(2)
```

	Facility ID	Facility Name	Age Group	Zip Code - 3 digits	Gender	Race	Ethnicity	Length of Stay	Admit Day of Week	Type of Admission	Patient Disposition	Discharge Year	Discharge Day of Week	CCS Diagnosis Code
0	66	Olean General Hospital	70 or Older	147	M	White	Not Span/Hispanic	2	None	Emergency	Home or Self Care	2016-01-01	None	108
1	66	Olean General Hospital	50 to 69	147	F	White	Not Span/Hispanic	5	None	Emergency	Inpatient Rehabilitation Facility	2016-01-01	None	108

3. Quick EDA

```
quality_report(train)
train.shape
```

	Total NaN	Percent of NaN	Nunique	Dtype
Zip Code - 3 digits	5	0.105285	47	object
Facility ID	0	0.000000	8	int64
CCS Procedure Description	0	0.000000	75	object
Total Charges	0	0.000000	4744	float64
Emergency Department Indicator	0	0.000000	2	object
Abortion Edit Indicator	0	0.000000	1	object
Birth Weight	0	0.000000	2	int64
Source of Payment 1	0	0.000000	9	object
APR Medical Surgical Description	0	0.000000	1	object
APR Risk of Mortality	0	0.000000	4	object
APR Severity of Illness Description	0	0.000000	4	object
APR Severity of Illness Code	0	0.000000	4	int64
APR MDC Description	0	0.000000	1	object
APR MDC Code	0	0.000000	1	int64
APR DRG Description	0	0.000000	1	object
APR DRG Code	0	0.000000	1	int64
CCS Procedure Code	0	0.000000	75	int64
Facility Name	0	0.000000	8	object
CCS Diagnosis Description	0	0.000000	4	object
CCS Diagnosis Code	0	0.000000	4	int64
Discharge Day of Week	0	0.000000	1	object
Discharge Year	0	0.000000	1	datetime64[ns]
Patient Disposition	0	0.000000	17	object
Type of Admission	0	0.000000	3	object
Admit Day of Week	0	0.000000	1	object
Length of Stay	0	0.000000	50	object
Ethnicity	0	0.000000	4	object
Race	0	0.000000	4	object
Gender	0	0.000000	2	object
Age Group	0	0.000000	5	object
Total Costs	0	0.000000	4744	float64

```
(4749, 31)
```

```
# there are 5 missing values in the zip code field.
#dropping these
train.dropna(axis = 0,inplace= True)
train.shape
```

```
(4744, 31)
```

```
#dropping redundant columns i.e columns that give little new information
train.drop(['Facility Name','APR MDC Code',
            'APR DRG Description','APR DRG Description',
            'Discharge Day of Week', 'Discharge Year',
            'Admit Day of Week','APR Medical Surgical Description'
            ,'Abortion Edit Indicator','Birth Weight'],axis =1, inplace = True)

train.shape
```

```
(4744, 22)
```

```
#the attributes reduced to 22 after dropping low information columns
```

```
train.columns
```

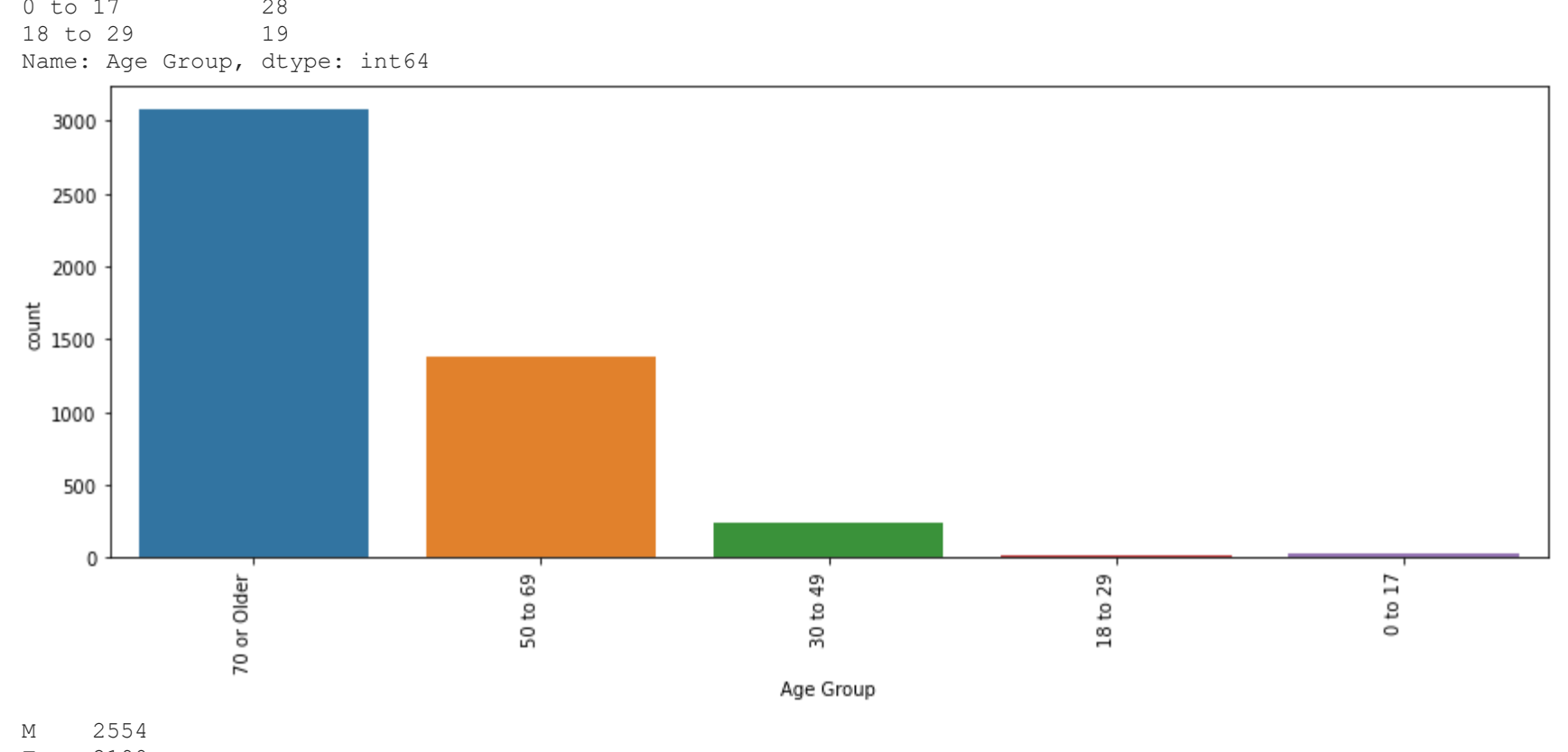
```
Index(['Facility ID', 'Age Group', 'Zip Code - 3 digits', 'Gender', 'Race',
       'Ethnicity', 'Length of Stay', 'Type of Admission',
       'Patient Disposition', 'CCS Diagnosis Code',
       'CCS Diagnosis Description', 'CCS Procedure Code',
       'CCS Procedure Description', 'APR DRG Code', 'APR MDC Description',
       'APR Severity of Illness Code', 'APR Severity of Illness Description',
       'APR Risk of Mortality', 'Source of Payment 1',
       'Emergency Department Indicator', 'Total Charges', 'Total Costs'],
      dtype='object')
```

Univariate Analysis

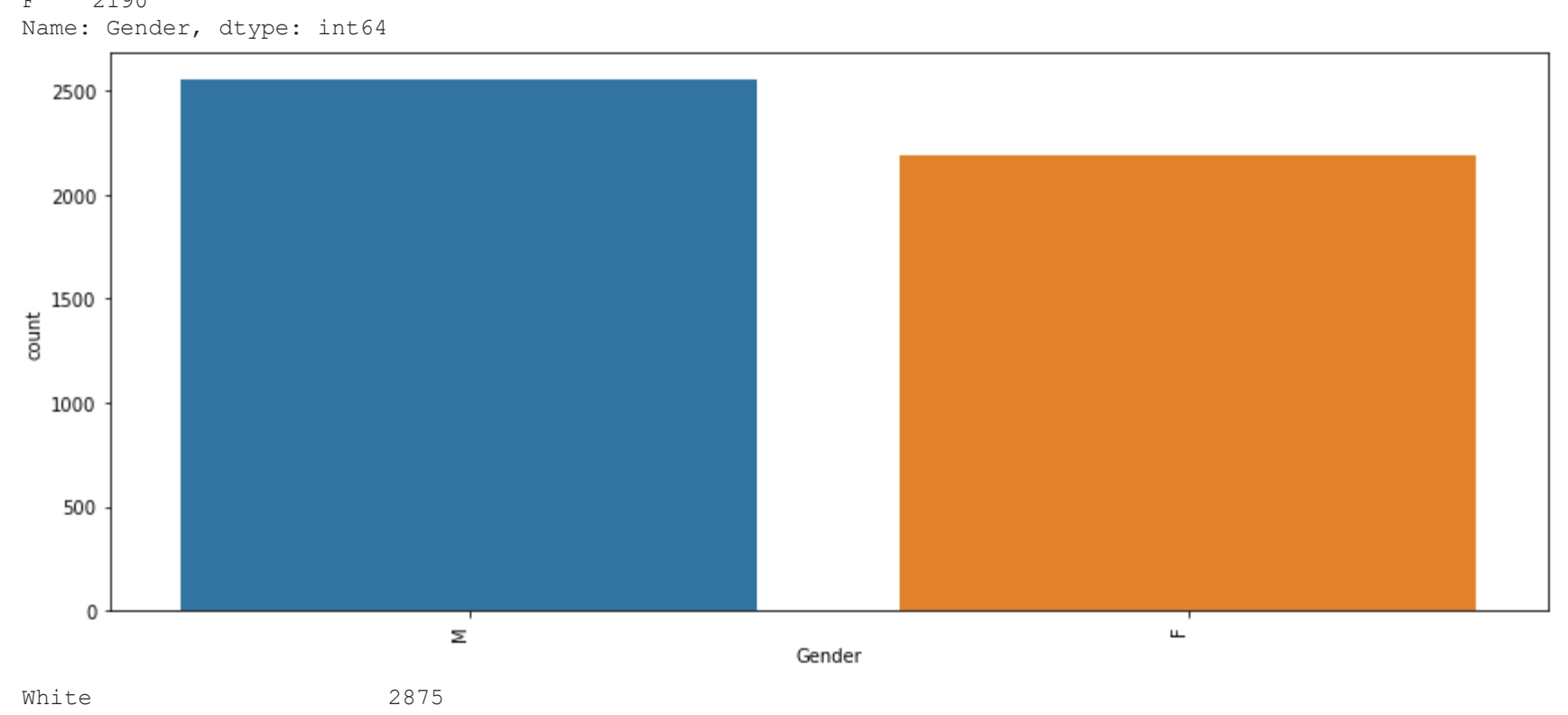
Categorical data

```
#using the user defined function to generate plots for the object variables
object_count_plot(train.drop(['Emergency Department Indicator','APR Risk of Mortality','APR Severity of Illness…
```

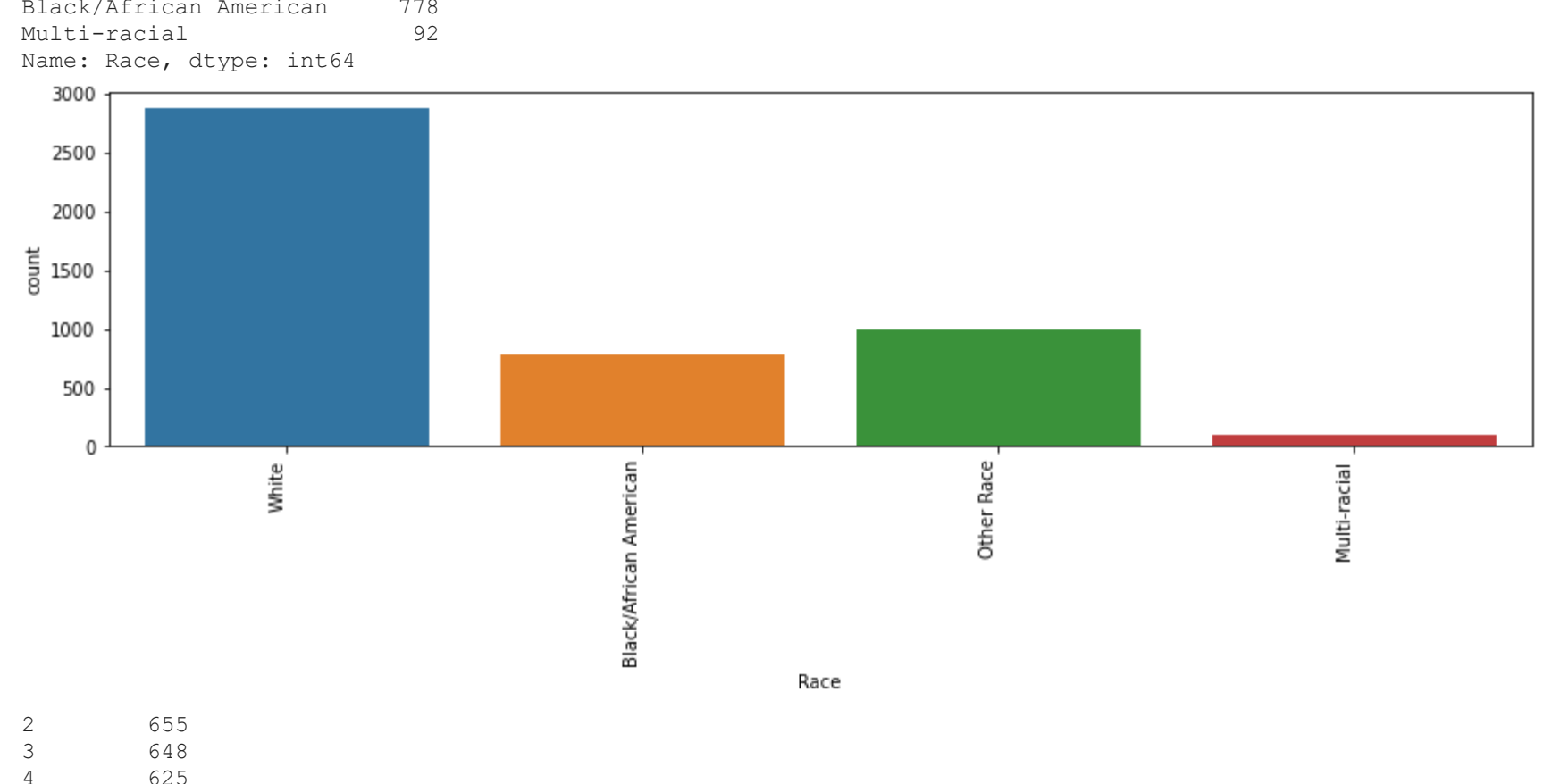
```
70 or Older    3082
50 to 69       1378
30 to 49        237
0 to 17          28
18 to 29         19
Name: Age Group, dtype: int64
```



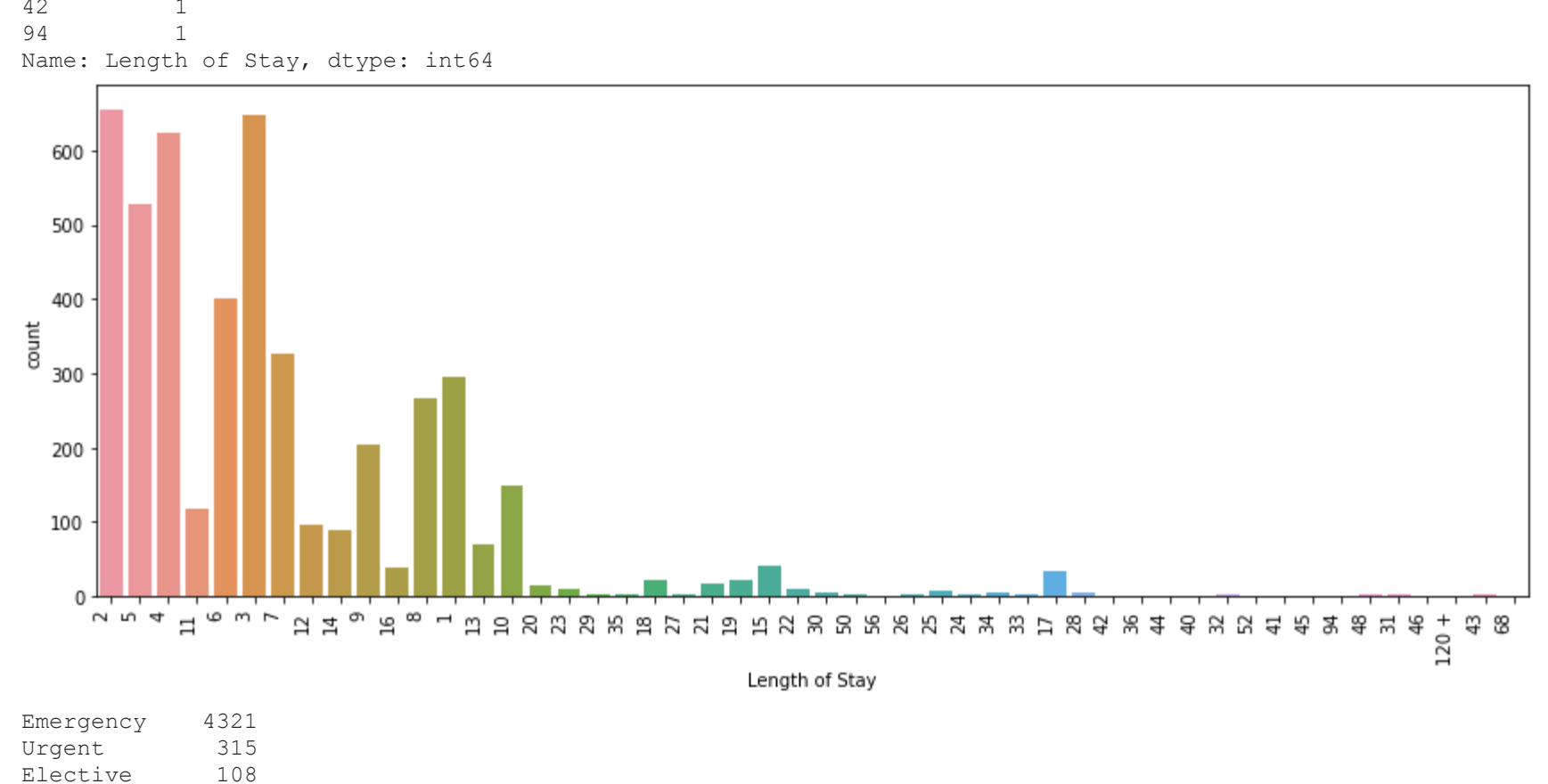
```
M    2554
F    2190
Name: Gender, dtype: int64
```



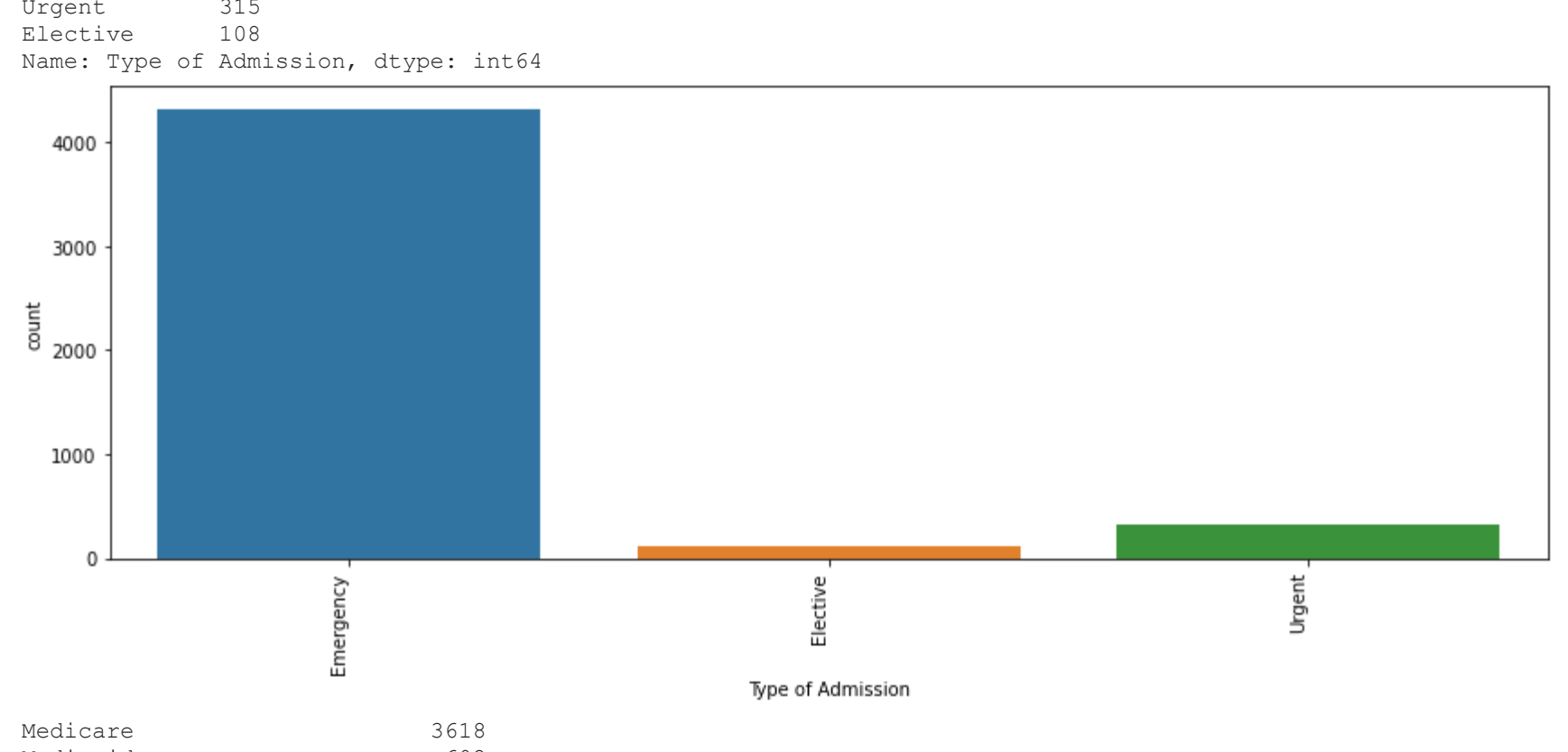
```
White                     2875
Other Race                 999
Black/African American     778
Multi-racial                92
Name: Race, dtype: int64
```



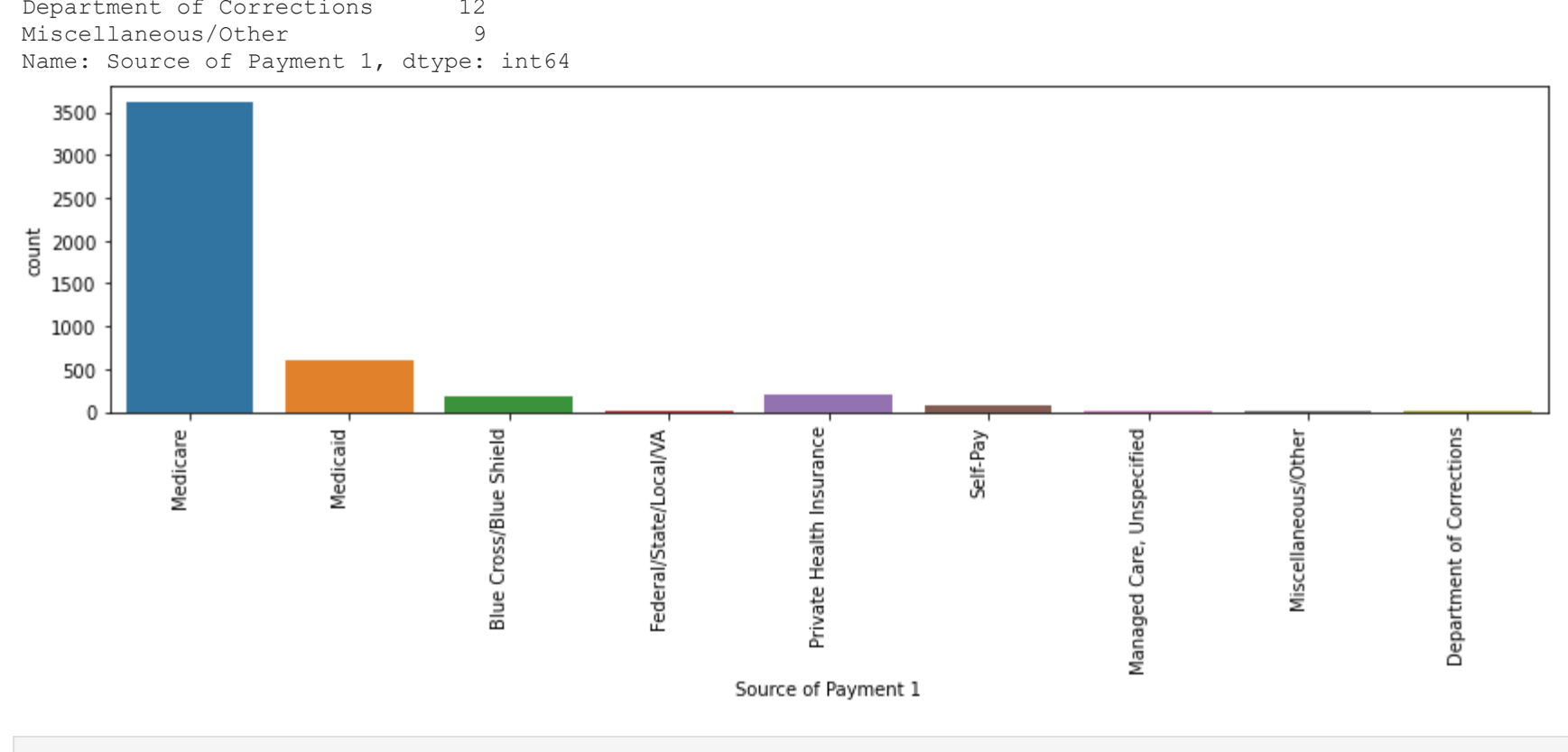
```
2       655
3       648
4       625
5       527
6       402
7       326
1       296
8       267
9       204
10      148
11      117
12       97
14       88
13       71
15       42
16       38
17       33
19       23
18       23
21       17
20       14
22       11
23       10
25        7
28        6
34        5
30        5
27        3
29        3
26        3
24        3
31        3
35        3
33        2
48        2
32        2
43        2
50        2
52        1
44        1
46        1
41        1
40        1
36        1
45        1
68        1
56        1
120 +     1
42        1
94        1
Name: Length of Stay, dtype: int64
```



```
Emergency    4321
Urgent        315
Elective      108
Name: Type of Admission, dtype: int64
```



```
Medicare                        3618
Medicaid                         602
Private Health Insurance         206
Blue Cross/Blue Shield           185
Self-Pay                          72
Federal/State/Local/VA            22
Managed Care, Unspecified         22
Department of Corrections         12
Miscellaneous/Other                9
Name: Source of Payment 1, dtype: int64
```

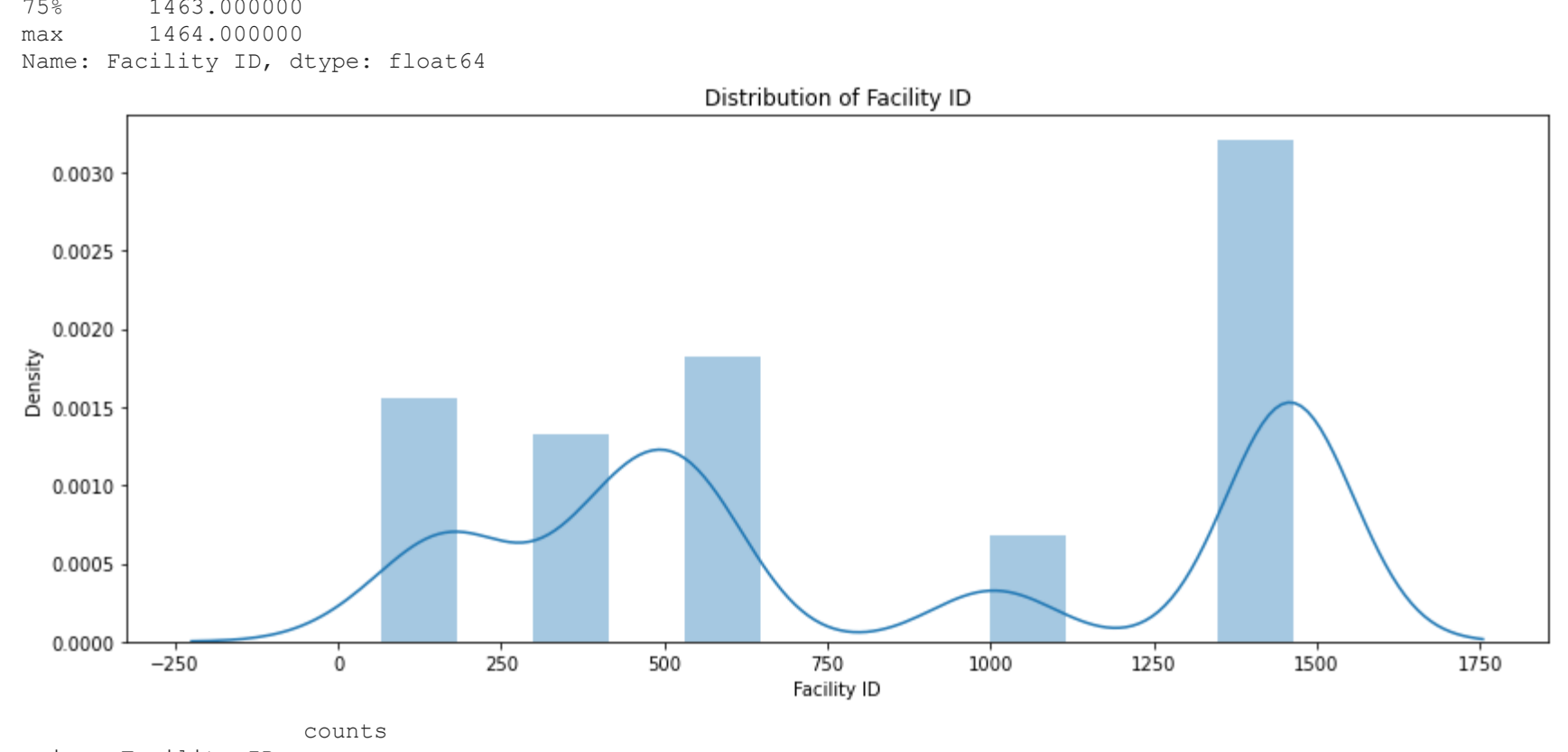


```
#### It can be seen that the most prominent age groups are "70 or older" and "50 to 69"
#### Patients are predominantly white
#### The male gender is more prominent
#### Patients typically don't stay for more than 10 days also there is inconsistency in the data entry "120+"
#### Emergency admissions are the most common
#### Most Patients pay with "Medicare" option
```

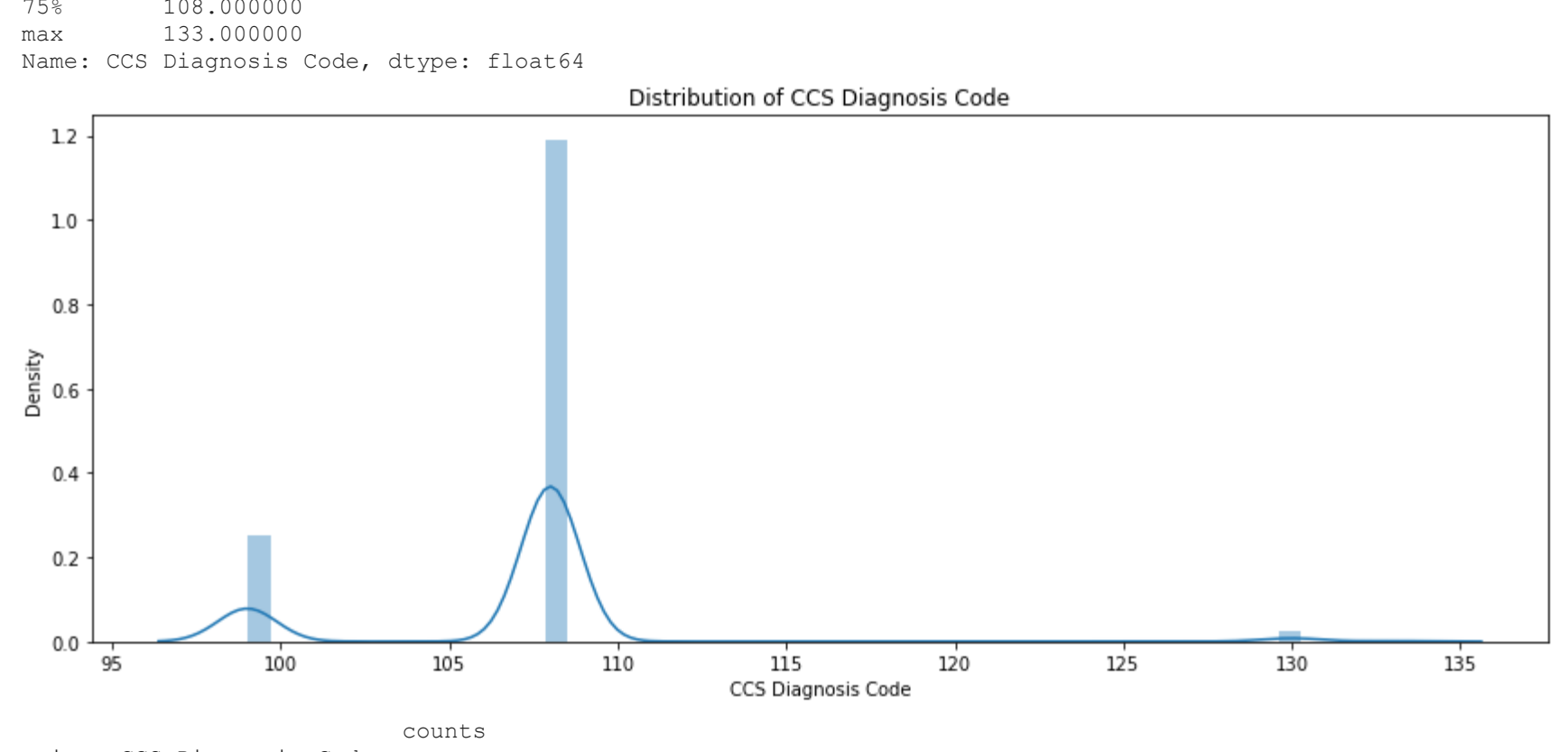
Numerical data

```
#performing EDA for the numeric data
numeric_distribution_plot(train.drop('APR DRG Code',axis = 1))
```

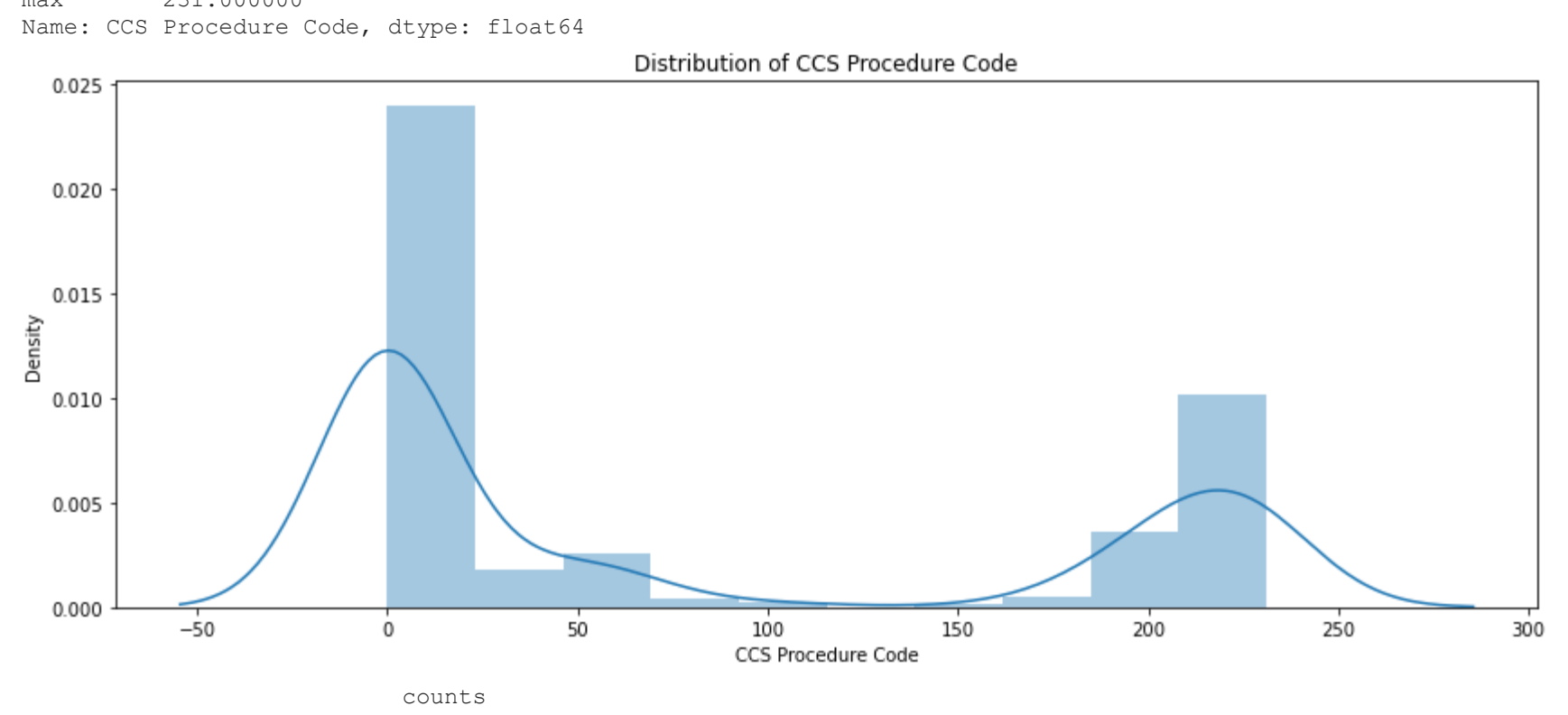
```
count    4744.000000
mean      831.144393
std       526.434521
min        66.000000
25%       413.000000
50%       541.000000
75%      1463.000000
max      1464.000000
Name: Facility ID, dtype: float64
```



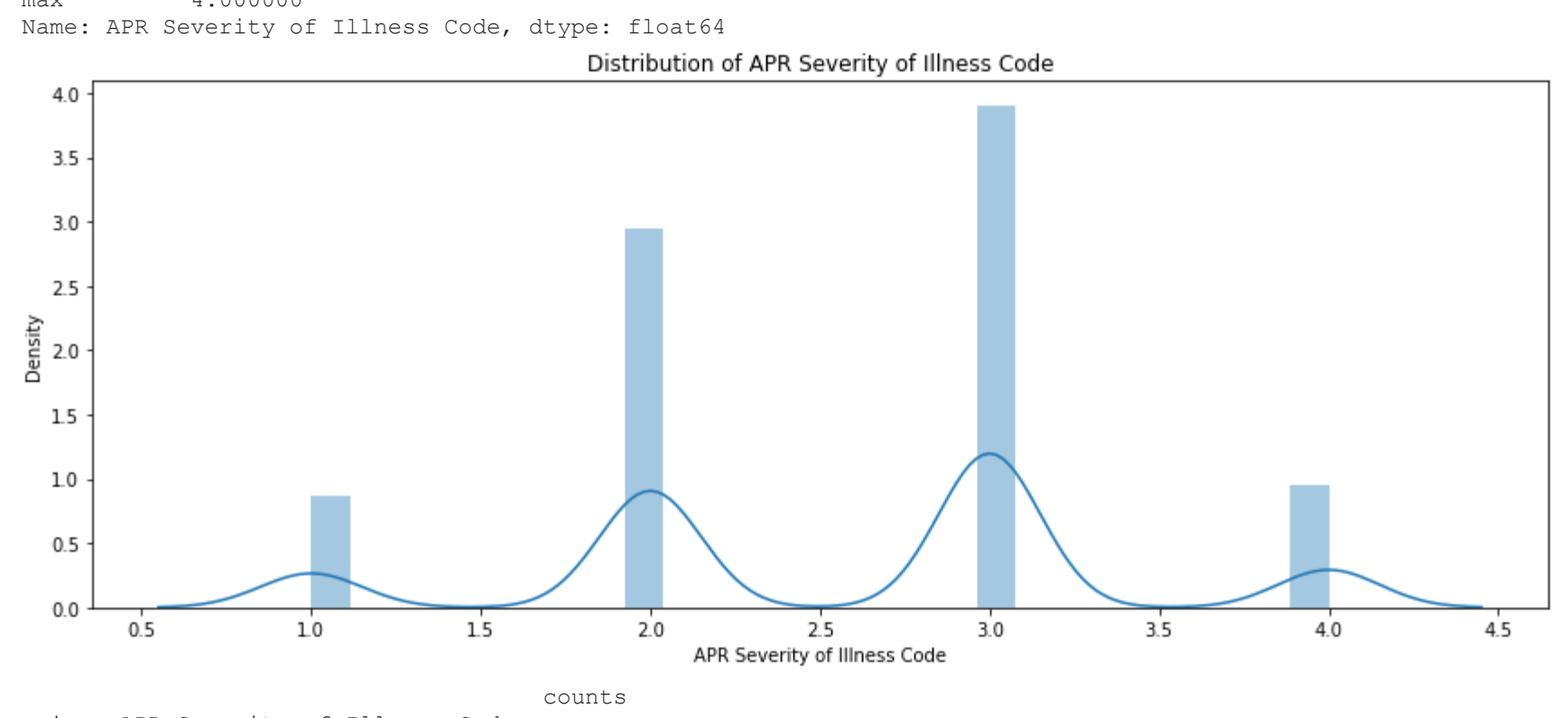
```
                    counts
unique_Facility ID
541                   1008
1464                   846
413                    730
181                    677
1463                   583
                    counts
unique_Facility ID
181                    677
1463                   583
1005                   375
1439                   343
66                     182
count    4744.000000
mean      106.897976
std         4.762604
min        99.000000
25%       108.000000
50%       108.000000
75%       108.000000
max       133.000000
Name: CCS Diagnosis Code, dtype: float64
```



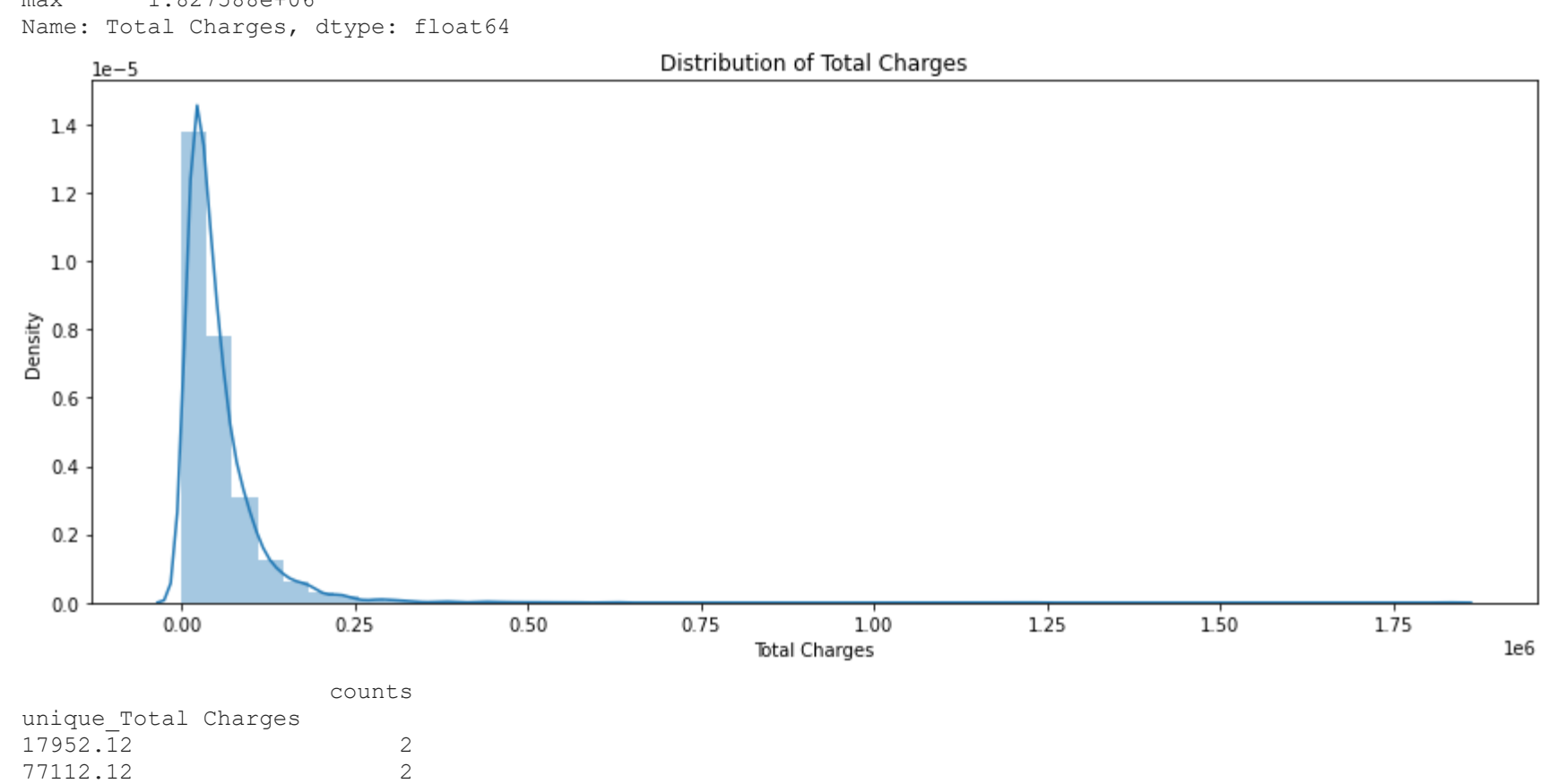
```
                          counts
unique_CCS Diagnosis Code
108                         3835
99                           815
130                           81
133                           13
                          counts
unique_CCS Diagnosis Code
108                         3835
99                           815
130                           81
133                           13
count    4744.000000
mean       77.533938
std        98.566693
min         0.000000
25%         0.000000
50%         0.000000
75%       203.000000
max       231.000000
Name: CCS Procedure Code, dtype: float64
```



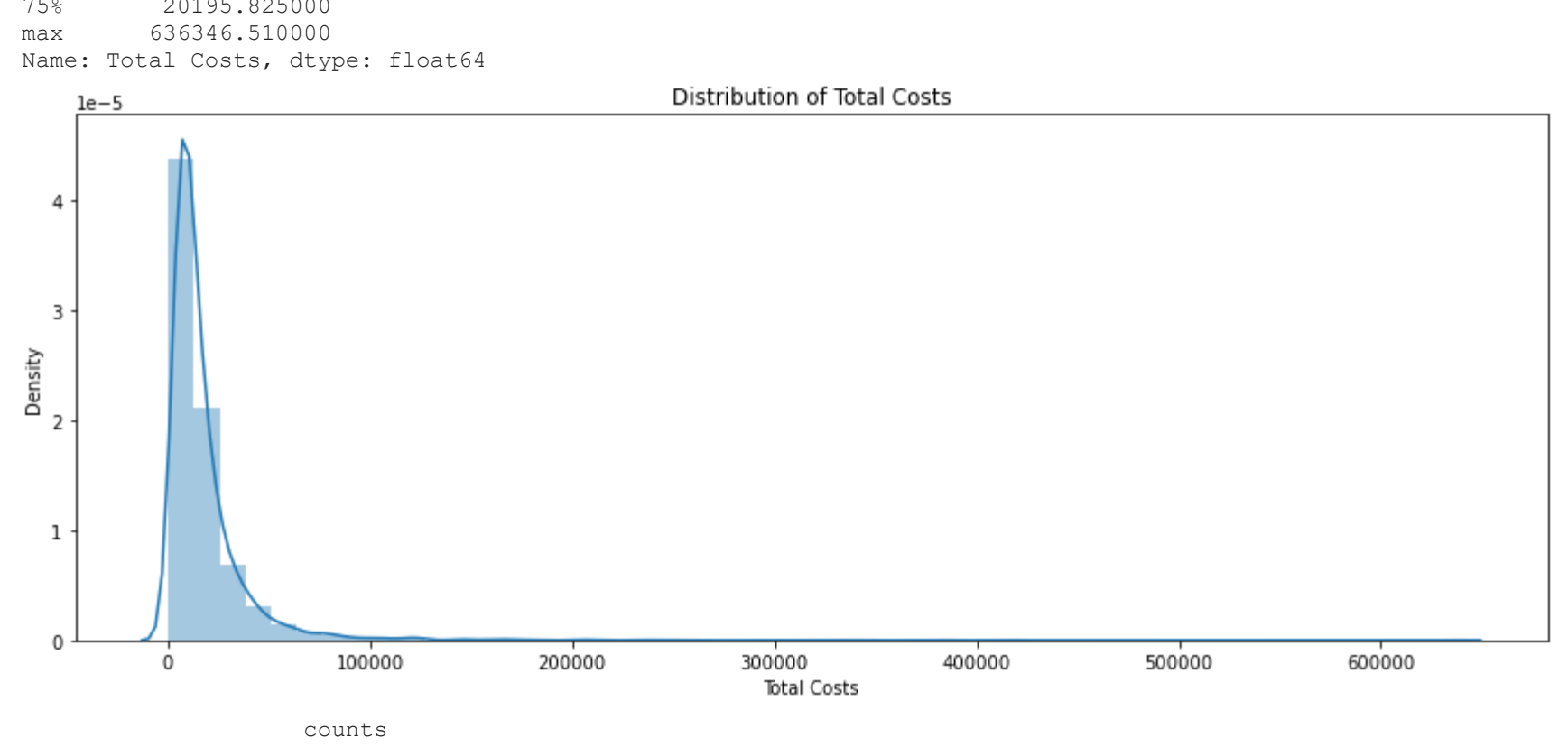
```
                          counts
unique_CCS Procedure Code
0                           2621
231                          431
216                          393
193                          210
58                           153
                          counts
unique_CCS Procedure Code
191                            1
215                            1
207                            1
211                            1
98                             1
count    4744.000000
mean        2.569983
std         0.814786
min         1.000000
25%         2.000000
50%         3.000000
75%         3.000000
max         4.000000
Name: APR Severity of Illness Code, dtype: float64
```



```
                                    counts
unique_APR Severity of Illness Code
3                                     2134
2                                     1617
4                                      521
1                                      472
                                    counts
unique_APR Severity of Illness Code
3                                     2134
2                                     1617
4                                      521
1                                      472
count    4.744000e+03
mean     5.315018e+04
std      6.307656e+04
min      1.000000e+00
25%      1.999650e+04
50%      3.638725e+04
75%      6.489832e+04
max      1.827586e+06
Name: Total Charges, dtype: float64
```



```
                     counts
unique_Total Charges
17952.12                  2
77112.12                  2
40288.46                  2
1.00                      2
12872.70                  2
                     counts
unique_Total Charges
59570.70                  1
40085.91                  1
46790.04                  1
16325.58                  1
28275.56                  1
count      4744.000000
mean      17527.486142
std       23523.993325
min           0.290000
25%        6670.457500
50%       11329.390000
75%       20195.825000
max      636346.510000
Name: Total Costs, dtype: float64
```



```
                   counts
unique_Total Costs
10293.05                2
6690.73                 2
6935.01                 2
5608.66                 2
0.29                    2
                   counts
unique_Total Costs
17414.50                1
37481.78                1
3463.59                 1
3800.74                 1
14768.31                1
```

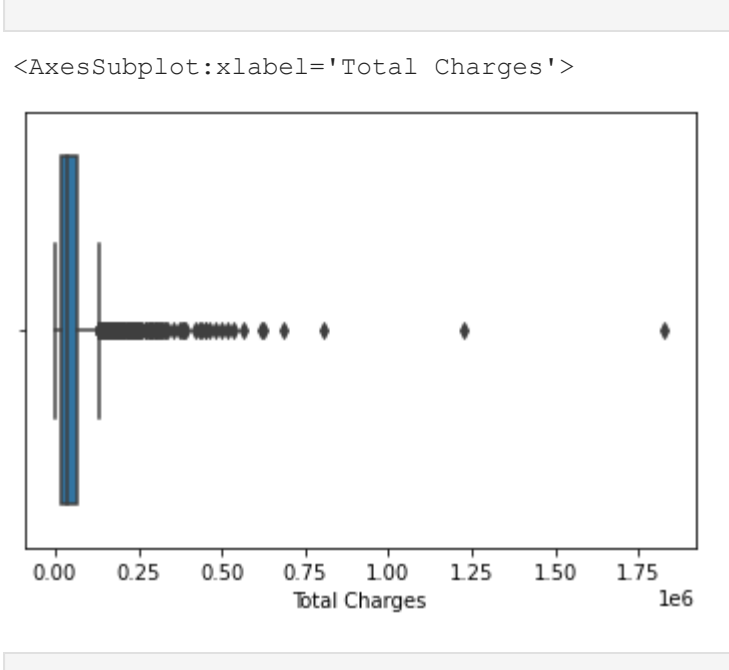
```
#the density plots and the count tables shown above.
#Facility with ID 541 i.e North Shore University Hospital has the most patients
#The most common CCS Diagnosis Code is 108 i.e Congestive heart failure; nonhypertensive
#Patients usually have APR Severity of Illness Code of 3
#Patients often have total charges of less than 250,000 and total costs of less than 100,000
```

Outlier Detection and Removal

```
#checking for outliers in total charges, total cost and length of stay using boxplots
# outliers are: train['Total Charges']>80000,train['Total Costs']>250000,train['Length of Stay']>65
```

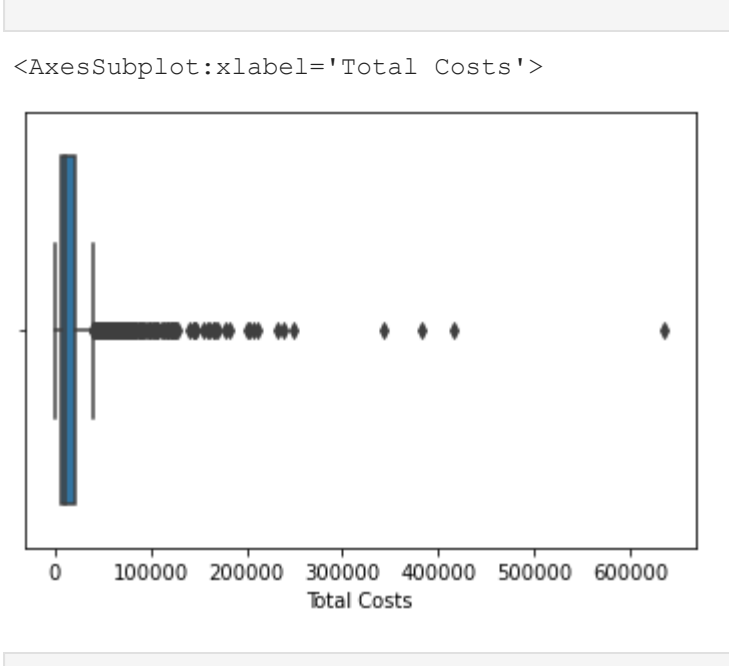
```
sns.boxplot(train['Total Charges'])
```

```
<AxesSubplot:xlabel='Total Charges'>
```



```
sns.boxplot(train['Total Costs'])
```

```
<AxesSubplot:xlabel='Total Costs'>
```



```
#to plot the box plot for length of stay
train['Length of Stay'].describe()
```

```
count     4744
unique      50
top          2
freq       655
Name: Length of Stay, dtype: object
```

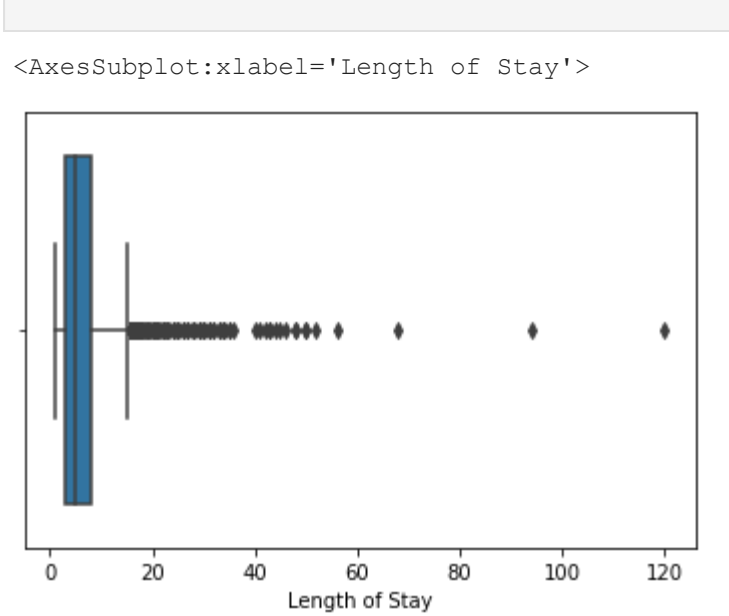
```
#found an unconventional input in the index
train[train['Length of Stay']=='120 +'].index.values
```

```
array([3262], dtype=int64)
```

```
#capping this observation at index 3262 to 120
train['Length of Stay'][3262] = 120
train['Length of Stay'] = train['Length of Stay'].apply(lambda x: float(x))
```

```
sns.boxplot(train['Length of Stay'])
```

```
<AxesSubplot:xlabel='Length of Stay'>
```



```
train.shape
```

```
(4744, 22)
```

```
train = train.drop(train.loc[train['Total Charges']>1000000].index)
train.shape
```

```
(4742, 22)
```

```
train = train.drop(train.loc[train['Total Costs']>275000].index)
train.shape
```

```
(4740, 22)
```

```
train = train.drop(train.loc[train['Length of Stay']>65].index)
train.shape
```

```
(4738, 22)
```

```
train.describe()
```

	Facility ID	Length of Stay	CCS Diagnosis Code	CCS Procedure Code	APR DRG Code	APR Severity of Illness Code	Total Charges	Total Costs
count	4738.000000	4738.000000	4738.000000	4738.000000	4738.0	4738.000000	4738.000000	4738.000000
mean	830.354158	6.138244	106.896581	77.426973	194.0	2.568594	52025.613527	17086.367617
std	526.298669	5.137141	4.765458	98.546669	0.0	0.814193	52123.538960	19062.998834
min	66.000000	1.000000	99.000000	0.000000	194.0	1.000000	1.000000	0.290000
25%	413.000000	3.000000	108.000000	0.000000	194.0	2.000000	19987.777500	6667.802500
50%	541.000000	5.000000	108.000000	0.000000	194.0	3.000000	36311.345000	11325.590000
75%	1463.000000	8.000000	108.000000	203.000000	194.0	3.000000	64795.425000	20129.780000
max	1464.000000	56.000000	133.000000	231.000000	194.0	4.000000	623272.480000	237953.710000

4. Data Preprocessing

```
train.columns
```

```
Index(['Facility ID', 'Age Group', 'Zip Code - 3 digits', 'Gender', 'Race',
       'Ethnicity', 'Length of Stay', 'Type of Admission',
       'Patient Disposition', 'CCS Diagnosis Code',
       'CCS Diagnosis Description', 'CCS Procedure Code',
       'CCS Procedure Description', 'APR DRG Code', 'APR MDC Description',
       'APR Severity of Illness Code', 'APR Severity of Illness Description',
       'APR Risk of Mortality', 'Source of Payment 1',
       'Emergency Department Indicator', 'Total Charges', 'Total Costs'],
      dtype='object')
```

```
quality_report(train)
```

	Total NaN	Percent of NaN	Nunique	Dtype
Facility ID	0	0.0	8	int64
Age Group	0	0.0	5	object
Total Charges	0	0.0	4733	float64
Emergency Department Indicator	0	0.0	2	object
Source of Payment 1	0	0.0	9	object
APR Risk of Mortality	0	0.0	4	object
APR Severity of Illness Description	0	0.0	4	object
APR Severity of Illness Code	0	0.0	4	int64
APR MDC Description	0	0.0	1	object
APR DRG Code	0	0.0	1	int64
CCS Procedure Description	0	0.0	75	object
CCS Procedure Code	0	0.0	75	int64
CCS Diagnosis Description	0	0.0	4	object
CCS Diagnosis Code	0	0.0	4	int64
Patient Disposition	0	0.0	17	object
Type of Admission	0	0.0	3	object
Length of Stay	0	0.0	47	float64
Ethnicity	0	0.0	4	object
Race	0	0.0	4	object
Gender	0	0.0	2	object
Zip Code - 3 digits	0	0.0	47	object
Total Costs	0	0.0	4733	float64

```
#selecting all categorical columns
fields = []
for col in train.columns:
    if train[col].dtype == 'object':
        fields.append(col)
print(fields)
```

```
['Age Group', 'Zip Code - 3 digits', 'Gender', 'Race', 'Ethnicity', 'Type of Admission', 'Patient Disposition',
'CCS Diagnosis Description', 'CCS Procedure Description', 'APR MDC Description', 'APR Severity of Illness Descr
iption', 'APR Risk of Mortality', 'Source of Payment 1', 'Emergency Department Indicator']
```

Label Encoding

```
train['Length_of_Stay'] = train['Length of Stay']
train.drop('Length of Stay',axis = 1,inplace = True)
train['Total_Costs'] = train['Total Costs']
train.drop('Total Costs',axis = 1,inplace = True)
```

```
train['Total_Charges'] = train['Total Charges']
train.drop('Total Charges',axis = 1,inplace = True)
```

```
train['Total_Costs'][17]
```

```
7337.06
```

```
#transforming the 'timeline' columns into gaussian distibution - NOT NEEDED FOR TREE BASED MODELS
#train['timeline'] = np.log(train['timeline']).values
#test['timeline'] = np.log(test['timeline']).values
#label encoding the target variables: 'Total Cost'

Length_of_Stay_map =['>12','10-12','6-10','<6']
Total_Cost_map=['>19000','17,000-19,000','11,000-17,000','<11,000']
#manual mapping
#fields = ['Length_of_Stay', 'Total_Costs']
for index in list(train.index):
    if (train['Total_Costs'][index] <= 11000):
        train['Total_Costs'][index] = 1
    elif (train['Total_Costs'][index] >= 11000)&(train['Total_Costs'][index] <= 17000):
        train['Total_Costs'][index] = 2
    elif (train['Total_Costs'][index] >= 17000)&(train['Total_Costs'][index] <= 19000):
        train['Total_Costs'][index] = 3
    elif (train['Total_Costs'][index] >= 19000):
        train['Total_Costs'][index] = 4
    else:
        index+1
```

```
#label encoding the target variables: 'Length_of_Stay'
#Length_of_Stay_map =['>12','10-12','6-10','<6']
#Total_Cost_map=['>34000','17,000-19,000','11,000-17,000','<11,000']
#manual mapping

for index in list(train.index):
    if (train['Length_of_Stay'][index] <= 6):
        train['Length_of_Stay'][index] = 1
    elif (train['Length_of_Stay'][index] >= 6)&(train['Length_of_Stay'][index] <= 10):
        train['Length_of_Stay'][index] = 2
    elif (train['Length_of_Stay'][index] > 10)&(train['Length_of_Stay'][index] <= 12):
        train['Length_of_Stay'][index] = 3
    elif (train['Length_of_Stay'][index] > 12):
        train['Length_of_Stay'][index] = 4
    else:
        index+1
```

```
#representation of the four categories of Total Costs
train['Total_Costs'].value_counts()
```

```
1.0    2283
4.0    1281
2.0     947
3.0     227
Name: Total_Costs, dtype: int64
```

```
#representation of the four categories of Total Costs
train['Length_of_Stay'].value_counts()
```

```
1.0    3153
2.0     945
4.0     426
3.0     214
Name: Length_of_Stay, dtype: int64
```

```
quality_report(train)
```

	Total NaN	Percent of NaN	Nunique	Dtype
Facility ID	0	0.0	8	int64
Age Group	0	0.0	5	object
Total_Costs	0	0.0	4	float64
Length_of_Stay	0	0.0	4	float64
Emergency Department Indicator	0	0.0	2	object
Source of Payment 1	0	0.0	9	object
APR Risk of Mortality	0	0.0	4	object
APR Severity of Illness Description	0	0.0	4	object
APR Severity of Illness Code	0	0.0	4	int64
APR MDC Description	0	0.0	1	object
APR DRG Code	0	0.0	1	int64
CCS Procedure Description	0	0.0	75	object
CCS Procedure Code	0	0.0	75	int64
CCS Diagnosis Description	0	0.0	4	object
CCS Diagnosis Code	0	0.0	4	int64
Patient Disposition	0	0.0	17	object
Type of Admission	0	0.0	3	object
Ethnicity	0	0.0	4	object
Race	0	0.0	4	object
Gender	0	0.0	2	object
Zip Code - 3 digits	0	0.0	47	object
Total_Charges	0	0.0	4733	float64

```
#for object classes
le = LabelEncoder()
for col in train.columns:
    if train[col].dtype == 'object':
        train[col]=le.fit_transform(train[col]).astype(float)
```

```
#reviewing the transformations
quality_report(train)
```

	Total NaN	Percent of NaN	Nunique	Dtype
Facility ID	0	0.0	8	int64
Age Group	0	0.0	5	float64
Total_Costs	0	0.0	4	float64
Length_of_Stay	0	0.0	4	float64
Emergency Department Indicator	0	0.0	2	float64
Source of Payment 1	0	0.0	9	float64
APR Risk of Mortality	0	0.0	4	float64
APR Severity of Illness Description	0	0.0	4	float64
APR Severity of Illness Code	0	0.0	4	int64
APR MDC Description	0	0.0	1	float64
APR DRG Code	0	0.0	1	int64
CCS Procedure Description	0	0.0	75	float64
CCS Procedure Code	0	0.0	75	int64
CCS Diagnosis Description	0	0.0	4	float64
CCS Diagnosis Code	0	0.0	4	int64
Patient Disposition	0	0.0	17	float64
Type of Admission	0	0.0	3	float64
Ethnicity	0	0.0	4	float64
Race	0	0.0	4	float64
Gender	0	0.0	2	float64
Zip Code - 3 digits	0	0.0	47	float64
Total_Charges	0	0.0	4733	float64

```
#renaming some columns
#train['Total_Costs']=train['Total Costs']
#train.drop('Total Costs',axis =1,inplace = True)

#train['Total_Charges']=train['Total Charges']
#train.drop('Total_Charges',axis =1,inplace = True)
```

```
train.head(6)
```

	Facility ID	Age Group	Zip Code - 3 digits	Gender	Race	Ethnicity	Type of Admission	Patient Disposition	CCS Diagnosis Code	CCS Diagnosis Description	CCS Procedure Code	CCS Procedure Description	APR DRG Code	APR MDC Description
0	66	4.0	43.0	1.0	3.0	1.0	1.0	6.0	108	0.0	58	28.0	194	0.0
1	66	3.0	43.0	0.0	3.0	1.0	1.0	11.0	108	0.0	0	36.0	194	0.0
2	66	4.0	43.0	1.0	3.0	1.0	1.0	16.0	108	0.0	0	36.0	194	0.0
3	66	3.0	43.0	1.0	3.0	1.0	1.0	6.0	108	0.0	201	7.0	194	0.0
4	66	4.0	43.0	1.0	3.0	1.0	1.0	11.0	108	0.0	0	36.0	194	0.0
5	66	4.0	43.0	0.0	3.0	1.0	1.0	16.0	108	0.0	0	36.0	194	0.0

6.Feature Engineering

Polynomial features

This generates extra features based on percieved interactions between attributes

```
poly_feature_1 = ['Gender','Race','Ethnicity']
poly_feature_2 = ['CCS Diagnosis Code','CCS Diagnosis Description', 'CCS Procedure Code','CCS Procedure Descrip
poly_feature_3 = ['APR DRG Code', 'APR MDC Description','APR Severity of Illness Description','APR Severity of Illnes
```

```
from sklearn.preprocessing import PolynomialFeatures
```

```
poly = PolynomialFeatures(degree=2, interaction_only=True, include_bias=False)

poly1 = poly.fit_transform(train[poly_feature_1])
poly2 = poly.fit_transform(train[poly_feature_2])
poly3 = poly.fit_transform(train[poly_feature_3])
```

```
df_poly1 = pd.DataFrame(poly1, columns=[f"poly1_{i}" for i in range(poly1.shape[1])])
df_poly2 = pd.DataFrame(poly2, columns=[f"poly2_{i}" for i in range(poly2.shape[1])])
df_poly3 = pd.DataFrame(poly3, columns=[f"poly3_{i}" for i in range(poly3.shape[1])])
```

```
train.columns
```

```
Index(['Facility ID', 'Age Group', 'Zip Code - 3 digits', 'Gender', 'Race',
       'Ethnicity', 'Type of Admission', 'Patient Disposition',
       'CCS Diagnosis Code', 'CCS Diagnosis Description', 'CCS Procedure Code',
       'CCS Procedure Description', 'APR DRG Code', 'APR MDC Description',
       'APR Severity of Illness Code', 'APR Severity of Illness Description',
       'APR Risk of Mortality', 'Source of Payment 1',
       'Emergency Department Indicator', 'Length_of_Stay', 'Total_Costs',
       'Total_Charges'],
      dtype='object')
```

```
new_data = pd.concat([train, df_poly1], axis = 1)
new_data = pd.concat([new_data, df_poly2],axis = 1)
new_data = pd.concat([new_data, df_poly3],axis = 1)
```

```
new_data.columns
```

```
Index(['Facility ID', 'Age Group', 'Zip Code - 3 digits', 'Gender', 'Race',
       'Ethnicity', 'Type of Admission', 'Patient Disposition',
       'CCS Diagnosis Code', 'CCS Diagnosis Description', 'CCS Procedure Code',
       'CCS Procedure Description', 'APR DRG Code', 'APR MDC Description',
       'APR Severity of Illness Code', 'APR Severity of Illness Description',
       'APR Risk of Mortality', 'Source of Payment 1',
       'Emergency Department Indicator', 'Length_of_Stay', 'Total_Costs',
       'Total_Charges', 'poly1_0', 'poly1_1', 'poly1_2', 'poly1_3', 'poly1_4',
       'poly1_5', 'poly2_0', 'poly2_1', 'poly2_2', 'poly2_3', 'poly2_4',
       'poly2_5', 'poly2_6', 'poly2_7', 'poly2_8', 'poly2_9', 'poly3_0',
       'poly3_1', 'poly3_2', 'poly3_3', 'poly3_4', 'poly3_5', 'poly3_6',
       'poly3_7', 'poly3_8', 'poly3_9', 'poly3_10', 'poly3_11', 'poly3_12',
       'poly3_13', 'poly3_14'],
      dtype='object')
```

```
new_data['geography'] = new_data[poly_feature_1].sum(axis = 1)
new_data['CCS Profile'] = new_data[poly_feature_2].sum(axis = 1)
new_data['APR Profile'] = new_data[poly_feature_3].sum(axis = 1)
```

```
# bin_label = [1,2,3,4,5]
new_data['Total_Charges_binned'] = pd.qcut(new_data.Total_Charges, q = [0, .2, .4, .6, .8, 1],duplicates='drop')
#new_data.drop('Total_Charges',axis =1,inplace = True)
```

```
new_data.columns
```

```
Index(['Facility ID', 'Age Group', 'Zip Code - 3 digits', 'Gender', 'Race',
       'Ethnicity', 'Type of Admission', 'Patient Disposition',
       'CCS Diagnosis Code', 'CCS Diagnosis Description', 'CCS Procedure Code',
       'CCS Procedure Description', 'APR DRG Code', 'APR MDC Description',
       'APR Severity of Illness Code', 'APR Severity of Illness Description',
       'APR Risk of Mortality', 'Source of Payment 1',
       'Emergency Department Indicator', 'Length_of_Stay', 'Total_Costs',
       'Total_Charges', 'poly1_0', 'poly1_1', 'poly1_2', 'poly1_3', 'poly1_4',
       'poly1_5', 'poly2_0', 'poly2_1', 'poly2_2', 'poly2_3', 'poly2_4',
       'poly2_5', 'poly2_6', 'poly2_7', 'poly2_8', 'poly2_9', 'poly3_0',
       'poly3_1', 'poly3_2', 'poly3_3', 'poly3_4', 'poly3_5', 'poly3_6',
       'poly3_7', 'poly3_8', 'poly3_9', 'poly3_10', 'poly3_11', 'poly3_12',
       'poly3_13', 'poly3_14', 'geography', 'CCS Profile', 'APR Profile',
       'Total_Charges_binned'],
      dtype='object')
```

```
#splitting the data into train and test features
# randomly predicting on 10% of the data set

test = new_data.sample(frac = 0.10)

#dropping the prediction set from the data to ensure unique training data
train = new_data.drop(list(test.index),axis = 0)
```

```
new_data.shape
```

```
(4749, 57)
```

```
test.shape
```

```
(475, 57)
```

```
train.shape
```

```
(4274, 57)
```

```
features = train.select_dtypes(include = 'number').columns
features
```

```
Index(['Facility ID', 'Age Group', 'Zip Code - 3 digits', 'Gender', 'Race',
       'Ethnicity', 'Type of Admission', 'Patient Disposition',
       'CCS Diagnosis Code', 'CCS Diagnosis Description', 'CCS Procedure Code',
       'CCS Procedure Description', 'APR DRG Code', 'APR MDC Description',
       'APR Severity of Illness Code', 'APR Severity of Illness Description',
       'APR Risk of Mortality', 'Source of Payment 1',
       'Emergency Department Indicator', 'Length_of_Stay', 'Total_Costs',
       'Total_Charges', 'poly1_0', 'poly1_1', 'poly1_2', 'poly1_3', 'poly1_4',
       'poly1_5', 'poly2_0', 'poly2_1', 'poly2_2', 'poly2_3', 'poly2_4',
       'poly2_5', 'poly2_6', 'poly2_7', 'poly2_8', 'poly2_9', 'poly3_0',
       'poly3_1', 'poly3_2', 'poly3_3', 'poly3_4', 'poly3_5', 'poly3_6',
       'poly3_7', 'poly3_8', 'poly3_9', 'poly3_10', 'poly3_11', 'poly3_12',
       'poly3_13', 'poly3_14', 'geography', 'CCS Profile', 'APR Profile',
       'Total_Charges_binned'],
      dtype='object')
```

```
train.head(7)
```

	Facility ID	Age Group	Zip Code - 3 digits	Gender	Race	Ethnicity	Type of Admission	Patient Disposition	CCS Diagnosis Code	CCS Diagnosis Description	CCS Procedure Code	CCS Procedure Description	APR DRG Code	APR MDC Description
0	66.0	4.0	43.0	1.0	3.0	1.0	1.0	6.0	108.0	0.0	58.0	28.0	194.0	0.0
1	66.0	3.0	43.0	0.0	3.0	1.0	1.0	11.0	108.0	0.0	0.0	36.0	194.0	0.0
2	66.0	4.0	43.0	1.0	3.0	1.0	1.0	16.0	108.0	0.0	0.0	36.0	194.0	0.0
3	66.0	3.0	43.0	1.0	3.0	1.0	1.0	6.0	108.0	0.0	201.0	7.0	194.0	0.0
4	66.0	4.0	43.0	1.0	3.0	1.0	1.0	11.0	108.0	0.0	0.0	36.0	194.0	0.0
5	66.0	4.0	43.0	0.0	3.0	1.0	1.0	16.0	108.0	0.0	0.0	36.0	194.0	0.0
6	66.0	3.0	43.0	1.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0	0.0

```
#dropping missing values
train.dropna(axis = 0,inplace= True)
train.shape
```

```
(4255, 57)
```

6. Modelling

```
from catboost import CatBoostClassifier
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import log_loss, f1_score
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
from sklearn.model_selection import train_test_split, cross_val_score
```

```
#defining both length of stay and total costs
X = train[features]
X.drop('Length_of_Stay',axis =1,inplace=True)
#X.drop('Total_Costs',axis =1,inplace=True)
target = 'Length_of_Stay'
#target = 'Total_Costs'
y = train[target]
```

```
X.columns
```

```
Index(['Facility ID', 'Age Group', 'Zip Code - 3 digits', 'Gender', 'Race',
       'Ethnicity', 'Type of Admission', 'Patient Disposition',
       'CCS Diagnosis Code', 'CCS Diagnosis Description', 'CCS Procedure Code',
       'CCS Procedure Description', 'APR DRG Code', 'APR MDC Description',
       'APR Severity of Illness Code', 'APR Severity of Illness Description',
       'APR Risk of Mortality', 'Source of Payment 1',
       'Emergency Department Indicator', 'Total_Costs', 'Total_Charges',
       'poly1_0', 'poly1_1', 'poly1_2', 'poly1_3', 'poly1_4', 'poly1_5',
       'poly2_0', 'poly2_1', 'poly2_2', 'poly2_3', 'poly2_4', 'poly2_5',
       'poly2_6', 'poly2_7', 'poly2_8', 'poly2_9', 'poly3_0', 'poly3_1',
       'poly3_2', 'poly3_3', 'poly3_4', 'poly3_5', 'poly3_6', 'poly3_7',
       'poly3_8', 'poly3_9', 'poly3_10', 'poly3_11', 'poly3_12', 'poly3_13',
       'poly3_14', 'geography', 'CCS Profile', 'APR Profile',
       'Total_Charges_binned'],
      dtype='object')
```

```
#splitting with train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
X_train.columns
```

```
Index(['Facility ID', 'Age Group', 'Zip Code - 3 digits', 'Gender', 'Race',
       'Ethnicity', 'Type of Admission', 'Patient Disposition',
       'CCS Diagnosis Code', 'CCS Diagnosis Description', 'CCS Procedure Code',
       'CCS Procedure Description', 'APR DRG Code', 'APR MDC Description',
       'APR Severity of Illness Code', 'APR Severity of Illness Description',
       'APR Risk of Mortality', 'Source of Payment 1',
       'Emergency Department Indicator', 'Total_Costs', 'Total_Charges',
       'poly1_0', 'poly1_1', 'poly1_2', 'poly1_3', 'poly1_4', 'poly1_5',
       'poly2_0', 'poly2_1', 'poly2_2', 'poly2_3', 'poly2_4', 'poly2_5',
       'poly2_6', 'poly2_7', 'poly2_8', 'poly2_9', 'poly3_0', 'poly3_1',
       'poly3_2', 'poly3_3', 'poly3_4', 'poly3_5', 'poly3_6', 'poly3_7',
       'poly3_8', 'poly3_9', 'poly3_10', 'poly3_11', 'poly3_12', 'poly3_13',
       'poly3_14', 'geography', 'CCS Profile', 'APR Profile',
       'Total_Charges_binned'],
      dtype='object')
```

Baseline Models: for length of stay prediction

MODEL 1 XGBClassifier #Benchmark Accuracy: 84.86%

```
from xgboost.sklearn import XGBClassifier
#initial model
xgb1 = XGBClassifier(learning_rate=0.01,
                    n_estimators=1000,
                    max_depth=5,
                    min_child_weight=1,
                    gamma=0,
                     average = 'micro',
                    subsample=0.8,
                    colsample_bytree=0.8,
                    objective='multi:softmax',
                    nthread=4,
                    num_class=4,
                    seed=27)

xgb1.fit(X_train, y_train)
prediction = xgb1.predict(X_test)
```

```
[12:41:53] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/learner.cc:573:
Parameters: { "average" } might not be used.

  This may not be accurate due to some parameters are only used in language bindings but
  passed down to XGBoost core.  Or some parameters are not used but slip through this
  verification. Please open an issue if you find above cases.


[12:41:53] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/learner.cc:1095: Starting
in XGBoost 1.3.0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merr
or' to 'mlogloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
```

```
#uncomment to see the real value predictions
#print(prediction)
accuracy = accuracy_score(y_test, prediction)
#f1_score = f1_score(prediction, y_test,average='micro')
print("Accuracy: %.2f%%" % (accuracy * 100.0))
#print("f1_score: %.2f%%" % (f1_score * 100.0))
```

```
Accuracy: 85.33%
```

```
from sklearn.metrics import accuracy_score

#MODEL 1 XGBClassifier #Benchmark f1 score 0.450828729281768

model = XGBClassifier(max_depth = 8, n_estimators = 500,num_class=4,objective='multi:softmax',learning_rate=0.0
model.fit(X_train, y_train)
prediction = model.predict(X_test)

#print(f1_score(prediction, y_test))

# evaluate predictions
accuracy = accuracy_score(y_test, prediction)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
[12:42:47] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/learner.cc:1095: Starting
in XGBoost 1.3.0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merr
or' to 'mlogloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
Accuracy: 84.86%
```

```
# evaluate predictions
#print(y_test)
```

```
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
Accuracy: 84.86%
```

MODEL 2 LGBM Classifier with oversampled inputs #Benchmark Accuracy: 95.77%

```
#defining both length of stay and total costs
X = train[features]
X.drop('Length_of_Stay',axis =1,inplace=True)
#X.drop('Total_Costs',axis =1,inplace=True)
target = 'Length_of_Stay'
#target = 'Total_Costs'
y = train[target]
```

```
#undersampling
#from imblearn.under_sampling import RandomUnderSampler
#rus = RandomUnderSampler(random_state = 0)

#X_train_res, y_train_res = rus.fit_resample(X, y)
```

```
X.columns
```

```
Index(['Facility ID', 'Age Group', 'Zip Code - 3 digits', 'Gender', 'Race',
       'Ethnicity', 'Type of Admission', 'Patient Disposition',
       'CCS Diagnosis Code', 'CCS Diagnosis Description', 'CCS Procedure Code',
       'CCS Procedure Description', 'APR DRG Code', 'APR MDC Description',
       'APR Severity of Illness Code', 'APR Severity of Illness Description',
       'APR Risk of Mortality', 'Source of Payment 1',
       'Emergency Department Indicator', 'Total_Costs', 'Total_Charges',
       'poly1_0', 'poly1_1', 'poly1_2', 'poly1_3', 'poly1_4', 'poly1_5',
       'poly2_0', 'poly2_1', 'poly2_2', 'poly2_3', 'poly2_4', 'poly2_5',
       'poly2_6', 'poly2_7', 'poly2_8', 'poly2_9', 'poly3_0', 'poly3_1',
       'poly3_2', 'poly3_3', 'poly3_4', 'poly3_5', 'poly3_6', 'poly3_7',
       'poly3_8', 'poly3_9', 'poly3_10', 'poly3_11', 'poly3_12', 'poly3_13',
       'poly3_14', 'geography', 'CCS Profile', 'APR Profile',
       'Total_Charges_binned'],
      dtype='object')
```

```
#oversampling
from imblearn.over_sampling import SMOTE

sm = SMOTE(random_state=42)
X_train_res, y_train_res = sm.fit_resample(X, y)
```

```
#MODEL 2 LGBM Classifier with oversampled inputs #Benchmark f1 score 0.4517184942716858
#X_train_res.drop('Length_of_Stay',axis = 1, inplace = True)
model = LGBMClassifier(max_depth=6, n_estimators = 1000, learning_rate=0.01,num_class=4,objective='multi:softm
model.fit(X_train_res, y_train_res)

#X_test.drop('Length_of_Stay',axis = 1, inplace = True)
prediction = model.predict(X_test)

print("Accuracy: %.2f%%" %  (accuracy_score(prediction, y_test)*100.0))

#X_train_res.drop('Length_of_Stay',axis = 1, inplace = 1)
```

```
Accuracy: 95.77%
```

MODEL 3 CatBoostClassifier no sampling #Benchmark Accuracy: 83.57%

```
#MODEL 3 CatBoostClassifier no sampling #Benchmark f1 score 0.3827314238273142
model = CatBoostClassifier(loss_function='MultiClass', class_weights= [.15, .35, .35,.15], iterations=1000, dep
model.fit(X_train, y_train)
prediction = model.predict(X_test)

print("Accuracy: %.2f%%" %  (accuracy_score(prediction, y_test)*100.0))
```

```
957:	learn: 0.0291888	total: 1m 56s	remaining: 5.11s
958:	learn: 0.0291517	total: 1m 56s	remaining: 4.99s
959:	learn: 0.0291007	total: 1m 56s	remaining: 4.87s
960:	learn: 0.0290947	total: 1m 56s	remaining: 4.75s
961:	learn: 0.0290000	total: 1m 57s	remaining: 4.62s
962:	learn: 0.0289902	total: 1m 57s	remaining: 4.5s
963:	learn: 0.0289131	total: 1m 57s	remaining: 4.38s
964:	learn: 0.0288877	total: 1m 57s	remaining: 4.26s
965:	learn: 0.0288243	total: 1m 57s	remaining: 4.13s
966:	learn: 0.0287817	total: 1m 57s	remaining: 4.01s
967:	learn: 0.0287384	total: 1m 57s	remaining: 3.89s
968:	learn: 0.0287067	total: 1m 57s	remaining: 3.77s
969:	learn: 0.0286808	total: 1m 57s	remaining: 3.64s
970:	learn: 0.0286661	total: 1m 57s	remaining: 3.52s
971:	learn: 0.0286398	total: 1m 58s	remaining: 3.4s
972:	learn: 0.0285939	total: 1m 58s	remaining: 3.28s
973:	learn: 0.0285715	total: 1m 58s	remaining: 3.16s
974:	learn: 0.0285295	total: 1m 58s	remaining: 3.03s
975:	learn: 0.0284884	total: 1m 58s	remaining: 2.91s
976:	learn: 0.0284538	total: 1m 58s	remaining: 2.79s
977:	learn: 0.0284126	total: 1m 58s	remaining: 2.67s
978:	learn: 0.0283778	total: 1m 58s	remaining: 2.55s
979:	learn: 0.0283558	total: 1m 58s	remaining: 2.42s
980:	learn: 0.0282940	total: 1m 58s	remaining: 2.3s
981:	learn: 0.0282880	total: 1m 59s	remaining: 2.18s
982:	learn: 0.0282853	total: 1m 59s	remaining: 2.06s
983:	learn: 0.0282467	total: 1m 59s	remaining: 1.94s
984:	learn: 0.0282149	total: 1m 59s	remaining: 1.82s
985:	learn: 0.0281867	total: 1m 59s	remaining: 1.7s
986:	learn: 0.0281705	total: 1m 59s	remaining: 1.57s
987:	learn: 0.0281372	total: 1m 59s	remaining: 1.45s
988:	learn: 0.0281039	total: 1m 59s	remaining: 1.33s
989:	learn: 0.0280683	total: 1m 59s	remaining: 1.21s
990:	learn: 0.0280371	total: 1m 59s	remaining: 1.09s
991:	learn: 0.0280088	total: 2m	remaining: 968ms
992:	learn: 0.0279724	total: 2m	remaining: 847ms
993:	learn: 0.0279493	total: 2m	remaining: 726ms
994:	learn: 0.0279113	total: 2m	remaining: 605ms
995:	learn: 0.0278917	total: 2m	remaining: 484ms
996:	learn: 0.0278605	total: 2m	remaining: 363ms
997:	learn: 0.0278228	total: 2m	remaining: 242ms
998:	learn: 0.0277942	total: 2m	remaining: 121ms
999:	learn: 0.0277657	total: 2m	remaining: 0us
Accuracy: 83.5%
```

Hyperparameter Tuning with optuna

In [110]:

```python
import optuna
```

In [110]:

```python
#defining both length of stay and total costs
X = train[features]
X.drop('Length_of_Stay',axis =1,inplace=True)
#X.drop('Total_Costs',axis =1,inplace=True)
target = 'Length_of_Stay'
#target = 'Total_Costs'
y = train[target]
```

In [110]:

```python
#X = train_df[new_feat]
```

Model 4 XGboost tuned, f1_score 0.4749944377084578 with 5 fold Stratified Cross Validation Accuracy 78.5%

In [110]:

```python
def objective(trial):
    accuracy_score=[]
    accuracy_predictions=[]

    param = {
#       param["objective"] = "Logloss"
    param['learning_rate'] = trial.suggest_discrete_uniform("learning_rate", 0.001, 0.1, 0.001)
    param['depth'] = trial.suggest_int('depth', 3, 15)
    param['l2_leaf_reg'] = trial.suggest_discrete_uniform('l2_leaf_reg', 1.0, 5.5, 0.5)
    param['min_child_samples'] = trial.suggest_categorical('min_child_samples', [1, 4, 8, 16, 32])
    param['iterations'] = trial.suggest_discrete_uniform("iterations", 200, 2000, 50)
    param['use_best_model'] = True
    param['colsample_bylevel'] = trial.suggest_float("colsample_bylevel", 0.01, 0.1)
    param['random_state'] = 0
#   param['scale_pos_weight'] = trial.suggest_int('scale_pos_weight', 4, 30)
    param['bootstrap_type'] = trial.suggest_categorical("bootstrap_type", ["Bayesian", "Bernoulli", "MVS"])
    param['logging_level'] = 'Silent'
    param['verbosity'] = 0



    fold=StratifiedKFold(n_splits=5)#15#5#10
    i=1
    for train_index, test_index in fold.split(X,y):

        X_train, X_test = X.iloc[train_index], X.iloc[test_index]
        y_train, y_test = y.iloc[train_index], y.iloc[test_index]
#       model = CatBoostClassifier(**param)
        model = XGBClassifier(**param)


        model.fit(X_train,y_train,eval_set=[(X_train,y_train),(X_test, y_test)], early_stopping_rounds=200, ver
        preds=model.predict(X_test)

        accuracy_score.append(accuracy_score(y_test,preds))

    return np.mean(accuracy_score)
```

In [110]:

```python
study = optuna.create_study(direction="maximize")
study.optimize(objective, n_trials=50)
```

```
[I 2021-11-28 13:46:56,603] A new study created in memory with name: no-name-a494571d-28a6-45e8-a3a6-52f12bcb8d
[I 2021-11-28 13:47:01,891] Trial 0 finished with value: 0.7728013990720666 and parameters: {'learning_rate':
0.094, 'depth': 12, 'l2_leaf_reg': 1.0, 'min_child_samples': 16, 'iterations': 900.0, 'colsample_bylevel': 0.0
...
[I 2021-11-28 13:50:59,111] Trial 49 finished with value: 0.7791857523463799 and parameters: {'learning_rate':
0.089, 'depth': 10, 'l2_leaf_reg': 1.5, 'min_child_samples': 32, 'iterations': 400.0, 'colsample_bylevel': 0.0
871685167948077, 'bootstrap_type': 'MVS'}. Best is trial 37 with value: 0.7854915233665999.
```

In [111]:

```python
#obtaining and using the results
print("Number of finished trials: {}".format(len(study.trials)))

print("Best trial:")
trial = study.best_trial

print("  Accuracy: {}".format(trial.value))

print("  Params: ")
for key, value in trial.params.items():
    print("    {}: {}".format(key, value))
```

```
Number of finished trials: 50
Best trial:
  Accuracy: 0.7854915233665999
  Params:
    learning_rate: 0.093
    depth: 7
    l2_leaf_reg: 3.5
    min_child_samples: 4
    iterations: 300.0
    colsample_bylevel: 0.03289855124398806
    bootstrap_type: MVS
```

7. Prediction using Model 1

In [112]:

```python
test
```

Out[112…

	Facility ID	Age Group	Zip Code - 3 digits	Gender	Race	Ethnicity	Type of Admission	Patient Disposition	CCS Diagnosis Code	CCS Diagnosis Description	CCS Procedure Code	CCS Procedure Description	APR DRG Code
4544	1464.0	4.0	3.0	1.0	3.0	2.0	1.0	7.0	108.0	0.0	193.0	20.0	194.0
3463	1463.0	3.0	12.0	1.0	2.0	1.0	0.0	15.0	108.0	0.0	213.0	64.0	194.0
2270	541.0	3.0	16.0	1.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
999	413.0	3.0	41.0	1.0	0.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
1609	541.0	3.0	16.0	1.0	3.0	2.0	1.0	7.0	99.0	1.0	222.0	4.0	194.0
3113	1438.0	4.0	11.0	1.0	2.0	2.0	1.0	7.0	108.0	0.0	179.0	13.0	194.0
1631	541.0	4.0	14.0	0.0	3.0	1.0	1.0	7.0	108.0	0.0	216.0	67.0	194.0
818	181.0	3.0	23.0	1.0	3.0	1.0	1.0	6.0	108.0	0.0	228.0	65.0	194.0
3173	1438.0	3.0	11.0	1.0	2.0	2.0	1.0	6.0	108.0	0.0	231.0	62.0	194.0
2369	541.0	4.0	9.0	0.0	3.0	1.0	1.0	7.0	108.0	0.0	217.0	61.0	194.0
717	181.0	3.0	23.0	0.0	3.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
2972	1005.0	4.0	26.0	1.0	3.0	1.0	1.0	6.0	99.0	1.0	0.0	36.0	194.0
798	181.0	2.0	24.0	1.0	2.0	3.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
4198	1464.0	4.0	3.0	0.0	2.0	2.0	1.0	7.0	99.0	1.0	0.0	36.0	194.0
3076	1438.0	3.0	11.0	1.0	0.0	1.0	1.0	12.0	108.0	0.0	231.0	62.0	194.0
2049	541.0	4.0	11.0	1.0	3.0	1.0	1.0	6.0	108.0	0.0	217.0	61.0	194.0
1440	413.0	4.0	44.0	1.0	3.0	1.0	2.0	16.0	108.0	0.0	0.0	36.0	194.0
1728	541.0	3.0	14.0	0.0	2.0	2.0	1.0	6.0	108.0	0.0	58.0	28.0	194.0
2828	1005.0	4.0	26.0	1.0	2.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
2920	1005.0	4.0	26.0	0.0	3.0	1.0	1.0	16.0	108.0	0.0	0.0	36.0	194.0
4628	1464.0	4.0	0.0	0.0	3.0	1.0	1.0	16.0	108.0	0.0	0.0	36.0	194.0
2346	541.0	4.0	16.0	0.0	2.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
4589	1464.0	4.0	0.0	0.0	2.0	1.0	1.0	3.0	99.0	1.0	216.0	67.0	194.0
1540	413.0	3.0	44.0	1.0	3.0	1.0	1.0	9.0	130.0	3.0	41.0	47.0	194.0
4737	1464.0	4.0	0.0	1.0	2.0	2.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
1552	413.0	4.0	42.0	1.0	3.0	1.0	1.0	10.0	108.0	0.0	0.0	36.0	194.0
228	181.0	4.0	24.0	0.0	3.0	1.0	1.0	16.0	99.0	1.0	0.0	36.0	194.0
1066	413.0	4.0	41.0	0.0	3.0	1.0	1.0	9.0	108.0	0.0	0.0	36.0	194.0
4131	1464.0	3.0	2.0	1.0	3.0	1.0	1.0	16.0	108.0	0.0	26.0	54.0	194.0
3090	1438.0	3.0	0.0	0.0	2.0	1.0	1.0	12.0	108.0	0.0	193.0	20.0	194.0
4317	1464.0	2.0	3.0	1.0	2.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
2904	1005.0	4.0	26.0	1.0	3.0	1.0	1.0	7.0	99.0	1.0	0.0	36.0	194.0
1084	413.0	4.0	40.0	1.0	3.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
3219	1438.0	2.0	0.0	1.0	0.0	1.0	1.0	6.0	108.0	0.0	231.0	62.0	194.0
4475	1464.0	3.0	0.0	1.0	1.0	0.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
2462	541.0	4.0	14.0	1.0	0.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
1895	541.0	4.0	12.0	1.0	3.0	1.0	1.0	16.0	108.0	0.0	0.0	36.0	194.0
4090	1464.0	4.0	3.0	0.0	2.0	1.0	1.0	7.0	108.0	0.0	193.0	20.0	194.0
3059	1438.0	2.0	3.0	1.0	0.0	1.0	1.0	6.0	108.0	0.0	231.0	62.0	194.0
302	181.0	3.0	23.0	0.0	3.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
2443	541.0	4.0	16.0	1.0	2.0	1.0	1.0	7.0	108.0	0.0	41.0	47.0	194.0
155	66.0	4.0	37.0	0.0	3.0	1.0	1.0	16.0	108.0	0.0	0.0	36.0	194.0
2283	541.0	4.0	13.0	0.0	3.0	2.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
3089	1438.0	4.0	13.0	0.0	2.0	1.0	1.0	9.0	108.0	0.0	231.0	62.0	194.0
1100	413.0	3.0	38.0	1.0	3.0	1.0	2.0	3.0	108.0	0.0	62.0	37.0	194.0
460	181.0	4.0	23.0	0.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
1302	413.0	3.0	42.0	1.0	0.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
886	413.0	4.0	42.0	0.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
3371	1463.0	4.0	0.0	1.0	3.0	1.0	1.0	16.0	108.0	0.0	0.0	36.0	194.0
1040	413.0	4.0	42.0	0.0	3.0	1.0	2.0	6.0	108.0	0.0	0.0	36.0	194.0
858	181.0	4.0	24.0	0.0	2.0	3.0	1.0	6.0	99.0	1.0	0.0	36.0	194.0
3401	1463.0	4.0	11.0	1.0	3.0	1.0	0.0	7.0	108.0	0.0	213.0	64.0	194.0
2366	541.0	4.0	12.0	0.0	3.0	1.0	1.0	6.0	108.0	0.0	193.0	20.0	194.0
519	181.0	3.0	24.0	1.0	0.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
1282	413.0	4.0	42.0	0.0	3.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
4688	1464.0	4.0	0.0	0.0	2.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
2930	1005.0	4.0	26.0	0.0	3.0	1.0	1.0	7.0	99.0	1.0	0.0	36.0	194.0
1037	413.0	4.0	36.0	1.0	3.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
4446	1464.0	3.0	3.0	1.0	0.0	1.0	1.0	7.0	108.0	0.0	193.0	20.0	194.0
425	181.0	4.0	24.0	0.0	3.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
3016	1438.0	3.0	12.0	1.0	3.0	1.0	1.0	6.0	108.0	0.0	198.0	33.0	194.0
1494	413.0	4.0	41.0	1.0	3.0	1.0	1.0	7.0	108.0	0.0	216.0	67.0	194.0
340	181.0	4.0	24.0	1.0	3.0	1.0	1.0	11.0	108.0	0.0	0.0	36.0	194.0
929	413.0	3.0	29.0	1.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
4667	1464.0	2.0	3.0	0.0	1.0	1.0	1.0	6.0	133.0	2.0	216.0	67.0	194.0
2696	1005.0	3.0	26.0	1.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
4668	1464.0	3.0	0.0	0.0	0.0	2.0	1.0	7.0	108.0	0.0	231.0	62.0	194.0
4700	1464.0	4.0	0.0	0.0	0.0	1.0	1.0	6.0	99.0	1.0	193.0	20.0	194.0
102	66.0	4.0	43.0	0.0	3.0	1.0	1.0	16.0	108.0	0.0	0.0	36.0	194.0
202	181.0	3.0	23.0	0.0	3.0	1.0	1.0	7.0	108.0	0.0	193.0	20.0	194.0
1550	413.0	3.0	31.0	0.0	3.0	1.0	2.0	6.0	108.0	0.0	0.0	36.0	194.0
2715	1005.0	3.0	26.0	0.0	3.0	1.0	1.0	7.0	99.0	1.0	91.0	63.0	194.0
3896	1463.0	3.0	0.0	0.0	2.0	1.0	1.0	7.0	108.0	0.0	193.0	20.0	194.0
2441	541.0	4.0	14.0	0.0	3.0	1.0	1.0	16.0	108.0	0.0	216.0	67.0	194.0
1574	413.0	3.0	42.0	1.0	0.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
4329	1464.0	3.0	3.0	0.0	2.0	1.0	1.0	7.0	99.0	1.0	231.0	62.0	194.0
1926	541.0	4.0	14.0	0.0	3.0	1.0	1.0	6.0	108.0	0.0	40.0	40.0	194.0
2673	1005.0	4.0	26.0	1.0	3.0	1.0	2.0	3.0	108.0	0.0	0.0	36.0	194.0
4710	1464.0	3.0	46.0	1.0	2.0	1.0	2.0	7.0	108.0	0.0	65.0	5.0	194.0
4673	1464.0	3.0	46.0	1.0	2.0	1.0	1.0	6.0	130.0	3.0	41.0	47.0	194.0
4255	1464.0	2.0	46.0	0.0	3.0	1.0	0.0	6.0	108.0	0.0	0.0	36.0	194.0
3235	1438.0	3.0	0.0	0.0	0.0	1.0	1.0	6.0	108.0	0.0	231.0	62.0	194.0
3225	1438.0	3.0	11.0	0.0	0.0	2.0	1.0	6.0	108.0	0.0	231.0	62.0	194.0
2589	541.0	4.0	14.0	0.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
4670	1464.0	4.0	0.0	0.0	2.0	1.0	1.0	6.0	99.0	1.0	231.0	62.0	194.0
343	181.0	3.0	24.0	1.0	3.0	1.0	1.0	6.0	99.0	1.0	0.0	36.0	194.0
402	181.0	3.0	23.0	1.0	3.0	1.0	1.0	6.0	130.0	3.0	41.0	47.0	194.0
2234	541.0	3.0	13.0	1.0	3.0	1.0	1.0	16.0	108.0	0.0	0.0	36.0	194.0
1828	541.0	4.0	13.0	0.0	0.0	1.0	1.0	11.0	108.0	0.0	0.0	36.0	194.0
2821	1005.0	3.0	26.0	0.0	3.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
3699	1463.0	2.0	0.0	0.0	2.0	2.0	1.0	6.0	99.0	1.0	0.0	36.0	194.0
646	181.0	4.0	23.0	1.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
2088	541.0	3.0	14.0	1.0	0.0	1.0	1.0	6.0	108.0	0.0	70.0	73.0	194.0
2319	541.0	4.0	14.0	0.0	3.0	1.0	1.0	6.0	108.0	0.0	217.0	61.0	194.0
439	181.0	3.0	24.0	0.0	3.0	1.0	1.0	16.0	108.0	0.0	0.0	36.0	194.0
816	181.0	4.0	24.0	1.0	0.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
1659	541.0	3.0	13.0	1.0	0.0	2.0	1.0	6.0	108.0	0.0	216.0	67.0	194.0
4242	1464.0	3.0	0.0	1.0	2.0	2.0	1.0	6.0	108.0	0.0	92.0	56.0	194.0
1680	541.0	3.0	13.0	1.0	0.0	2.0	1.0	7.0	108.0	0.0	216.0	67.0	194.0
304	181.0	4.0	23.0	0.0	2.0	3.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
4629	1464.0	4.0	4.0	1.0	0.0	1.0	0.0	3.0	108.0	0.0	216.0	67.0	194.0
3361	1463.0	3.0	46.0	1.0	3.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
4128	1464.0	3.0	46.0	0.0	3.0	1.0	1.0	6.0	108.0	0.0	231.0	62.0	194.0
2258	541.0	4.0	12.0	0.0	3.0	1.0	1.0	6.0	108.0	0.0	222.0	4.0	194.0
3428	1463.0	3.0	46.0	0.0	3.0	1.0	2.0	3.0	108.0	0.0	54.0	55.0	194.0
3894	1463.0	3.0	46.0	1.0	3.0	1.0	2.0	6.0	108.0	0.0	0.0	36.0	194.0
4353	1464.0	4.0	0.0	0.0	2.0	1.0	1.0	16.0	108.0	0.0	0.0	36.0	194.0
3670	1463.0	4.0	3.0	0.0	3.0	1.0	0.0	7.0	108.0	0.0	213.0	64.0	194.0
3141	1438.0	4.0	0.0	0.0	2.0	2.0	1.0	7.0	108.0	0.0	193.0	20.0	194.0
2231	541.0	4.0	12.0	0.0	2.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
1714	541.0	2.0	16.0	1.0	0.0	1.0	1.0	6.0	133.0	2.0	0.0	36.0	194.0
724	181.0	3.0	22.0	1.0	3.0	1.0	2.0	6.0	99.0	1.0	0.0	36.0	194.0
4624	1464.0	4.0	0.0	0.0	2.0	0.0	1.0	7.0	108.0	0.0	231.0	62.0	194.0
2147	541.0	4.0	9.0	0.0	3.0	1.0	1.0	6.0	108.0	0.0	216.0	67.0	194.0
1622	541.0	3.0	14.0	1.0	3.0	1.0	1.0	6.0	108.0	0.0	58.0	28.0	194.0
2997	1438.0	4.0	11.0	0.0	0.0	1.0	1.0	6.0	99.0	1.0	193.0	20.0	194.0
245	181.0	4.0	24.0	0.0	3.0	1.0	1.0	7.0	99.0	1.0	0.0	36.0	194.0
2756	1005.0	4.0	26.0	0.0	3.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
2625	1005.0	4.0	26.0	0.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
2389	541.0	4.0	12.0	0.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
3003	1438.0	4.0	0.0	1.0	0.0	1.0	1.0	7.0	108.0	0.0	216.0	67.0	194.0
4364	1464.0	4.0	0.0	1.0	2.0	2.0	1.0	16.0	108.0	0.0	155.0	3.0	194.0
1511	413.0	3.0	41.0	1.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
3925	1464.0	4.0	3.0	0.0	2.0	1.0	1.0	6.0	108.0	0.0	231.0	62.0	194.0
3964	1464.0	3.0	0.0	1.0	0.0	1.0	1.0	7.0	99.0	1.0	0.0	36.0	194.0
1265	413.0	3.0	42.0	1.0	3.0	0.0	1.0	7.0	99.0	1.0	0.0	36.0	194.0
3590	1463.0	4.0	13.0	0.0	3.0	1.0	1.0	9.0	108.0	0.0	108.0	31.0	194.0
1122	413.0	4.0	41.0	1.0	3.0	1.0	1.0	7.0	108.0	0.0	201.0	7.0	194.0
2254	541.0	4.0	13.0	1.0	0.0	1.0	2.0	7.0	99.0	1.0	0.0	36.0	194.0
561	181.0	4.0	23.0	0.0	3.0	1.0	1.0	7.0	108.0	0.0	203.0	25.0	194.0
2417	541.0	4.0	16.0	0.0	3.0	1.0	1.0	7.0	108.0	0.0	216.0	67.0	194.0
353	181.0	4.0	24.0	1.0	3.0	1.0	1.0	16.0	108.0	0.0	0.0	36.0	194.0
3456	1463.0	3.0	0.0	0.0	3.0	1.0	1.0	7.0	108.0	0.0	225.0	12.0	194.0
2947	1005.0	4.0	26.0	0.0	3.0	1.0	1.0	3.0	108.0	0.0	216.0	67.0	194.0
15	66.0	4.0	46.0	1.0	0.0	1.0	1.0	16.0	108.0	0.0	0.0	36.0	194.0
1986	541.0	3.0	13.0	1.0	0.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
655	181.0	4.0	23.0	1.0	0.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
1731	541.0	4.0	14.0	1.0	0.0	1.0	1.0	16.0	99.0	1.0	0.0	36.0	194.0
1939	541.0	4.0	9.0	1.0	3.0	2.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
2826	1005.0	3.0	26.0	1.0	3.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
3932	1464.0	4.0	0.0	0.0	2.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
499	181.0	4.0	24.0	1.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
3277	1438.0	3.0	13.0	1.0	2.0	1.0	1.0	6.0	108.0	0.0	180.0	57.0	194.0
1905	541.0	4.0	11.0	1.0	2.0	2.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
1480	413.0	4.0	41.0	0.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
2265	541.0	4.0	9.0	0.0	0.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
1181	413.0	4.0	42.0	1.0	3.0	1.0	1.0	7.0	108.0	0.0	225.0	12.0	194.0
3315	1438.0	4.0	11.0	1.0	2.0	2.0	1.0	6.0	108.0	0.0	183.0	68.0	194.0
3478	1463.0	4.0	11.0	1.0	3.0	1.0	1.0	11.0	108.0	0.0	222.0	4.0	194.0
2060	541.0	4.0	15.0	0.0	0.0	2.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
2362	541.0	4.0	9.0	0.0	3.0	1.0	1.0	3.0	99.0	1.0	216.0	67.0	194.0
1747	541.0	4.0	12.0	0.0	3.0	1.0	1.0	16.0	108.0	0.0	216.0	67.0	194.0
4181	1464.0	4.0	3.0	0.0	2.0	1.0	2.0	6.0	108.0	0.0	0.0	36.0	194.0
2249	541.0	4.0	9.0	0.0	2.0	1.0	1.0	16.0	108.0	0.0	171.0	69.0	194.0
4349	1464.0	4.0	0.0	0.0	1.0	2.0	1.0	9.0	108.0	0.0	54.0	55.0	194.0
827	181.0	3.0	23.0	1.0	0.0	1.0	1.0	16.0	108.0	0.0	0.0	36.0	194.0
1349	413.0	3.0	40.0	1.0	3.0	1.0	1.0	7.0	99.0	1.0	216.0	67.0	194.0
577	181.0	3.0	24.0	1.0	0.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
1630	541.0	4.0	9.0	0.0	3.0	2.0	1.0	7.0	99.0	1.0	0.0	36.0	194.0
2410	541.0	3.0	13.0	0.0	3.0	1.0	1.0	7.0	108.0	0.0	191.0	2.0	194.0
3218	1438.0	3.0	0.0	1.0	0.0	1.0	1.0	6.0	108.0	0.0	231.0	62.0	194.0
4481	1464.0	4.0	25.0	1.0	3.0	1.0	1.0	3.0	108.0	0.0	54.0	55.0	194.0
1683	541.0	3.0	9.0	0.0	2.0	1.0	1.0	6.0	108.0	0.0	216.0	67.0	194.0
787	181.0	4.0	24.0	0.0	3.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
3715	1463.0	3.0	11.0	1.0	0.0	1.0	1.0	6.0	99.0	1.0	0.0	36.0	194.0
2555	541.0	4.0	14.0	1.0	3.0	1.0	2.0	7.0	99.0	1.0	216.0	67.0	194.0
684	181.0	3.0	23.0	1.0	3.0	1.0	2.0	6.0	108.0	0.0	0.0	36.0	194.0
1816	541.0	3.0	10.0	0.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
1627	541.0	4.0	12.0	1.0	2.0	1.0	1.0	7.0	99.0	1.0	39.0	30.0	194.0
4236	1464.0	4.0	3.0	0.0	0.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
4656	1464.0	4.0	0.0	0.0	2.0	0.0	1.0	7.0	99.0	1.0	0.0	36.0	194.0
4371	1464.0	4.0	0.0	1.0	2.0	2.0	1.0	7.0	108.0	0.0	227.0	38.0	194.0
2890	1005.0	3.0	26.0	1.0	3.0	1.0	1.0	7.0	99.0	1.0	0.0	36.0	194.0
869	413.0	4.0	46.0	1.0	3.0	1.0	1.0	16.0	108.0	0.0	0.0	36.0	194.0
3891	1463.0	3.0	0.0	0.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
4397	1464.0	4.0	0.0	0.0	2.0	1.0	1.0	12.0	99.0	1.0	231.0	62.0	194.0
3260	1438.0	2.0	11.0	0.0	2.0	1.0	1.0	16.0	108.0	0.0	0.0	36.0	194.0
991	413.0	4.0	42.0	1.0	3.0	1.0	1.0	6.0	108.0	0.0	178.0	14.0	194.0
679	181.0	3.0	24.0	1.0	3.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
803	181.0	4.0	23.0	0.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
4674	1464.0	4.0	3.0	0.0	2.0	1.0	1.0	6.0	99.0	1.0	231.0	62.0	194.0
935	413.0	4.0	29.0	1.0	3.0	1.0	1.0	6.0	108.0	0.0	231.0	62.0	194.0
272	181.0	4.0	22.0	0.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
2395	541.0	3.0	18.0	1.0	3.0	1.0	1.0	6.0	130.0	3.0	39.0	30.0	194.0
2204	541.0	4.0	12.0	1.0	3.0	2.0	1.0	6.0	108.0	0.0	216.0	67.0	194.0
4109	1464.0	3.0	0.0	0.0	1.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
1602	541.0	3.0	12.0	0.0	3.0	1.0	1.0	16.0	108.0	0.0	231.0	62.0	194.0
1033	413.0	3.0	40.0	1.0	3.0	1.0	1.0	6.0	130.0	3.0	39.0	30.0	194.0
3883	1463.0	4.0	0.0	1.0	0.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
3694	1463.0	4.0	11.0	0.0	3.0	1.0	1.0	9.0	108.0	0.0	0.0	36.0	194.0
525	181.0	3.0	4.0	0.0	3.0	1.0	1.0	7.0	99.0	1.0	54.0	55.0	194.0
938	413.0	4.0	41.0	1.0	0.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
1431	413.0	3.0	42.0	1.0	0.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
2355	541.0	3.0	9.0	1.0	0.0	2.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
4626	1464.0	3.0	3.0	1.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
1042	541.0	4.0	12.0	0.0	3.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
949	413.0	4.0	40.0	1.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
1352	413.0	4.0	36.0	1.0	3.0	1.0	1.0	16.0	99.0	1.0	216.0	67.0	194.0
4222	1464.0	4.0	0.0	1.0	2.0	1.0	1.0	6.0	99.0	1.0	0.0	36.0	194.0
386	181.0	3.0	24.0	0.0	3.0	1.0	1.0	16.0	108.0	0.0	0.0	36.0	194.0
4356	1464.0	4.0	0.0	1.0	0.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
2663	1005.0	4.0	26.0	1.0	3.0	1.0	1.0	6.0	108.0	0.0	222.0	4.0	194.0
4264	1464.0	4.0	0.0	0.0	3.0	1.0	2.0	16.0	108.0	0.0	0.0	36.0	194.0
1912	541.0	4.0	13.0	0.0	3.0	1.0	1.0	16.0	99.0	1.0	0.0	36.0	194.0
222	181.0	2.0	23.0	1.0	3.0	1.0	1.0	16.0	108.0	0.0	0.0	36.0	194.0
2829	1005.0	4.0	26.0	0.0	3.0	1.0	1.0	16.0	108.0	0.0	39.0	30.0	194.0
3806	1463.0	3.0	12.0	1.0	3.0	1.0	1.0	7.0	99.0	1.0	216.0	67.0	194.0
3674	1463.0	4.0	11.0	1.0	0.0	2.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
1985	1464.0	4.0	0.0	0.0	2.0	1.0	2.0	16.0	108.0	0.0	222.0	4.0	194.0
1857	541.0	4.0	16.0	1.0	3.0	2.0	1.0	7.0	108.0	0.0	193.0	20.0	194.0
1653	541.0	4.0	12.0	0.0	3.0	1.0	1.0	7.0	108.0	0.0	231.0	62.0	194.0
1941	541.0	4.0	12.0	1.0	3.0	1.0	1.0	6.0	130.0	3.0	41.0	47.0	194.0
1937	541.0	4.0	13.0	0.0	3.0	1.0	1.0	16.0	108.0	0.0	0.0	36.0	194.0
505	181.0	4.0	24.0	0.0	3.0	1.0	1.0	7.0	130.0	3.0	39.0	30.0	194.0
4028	1464.0	3.0	0.0	1.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
2264	541.0	4.0	14.0	1.0	3.0	1.0	1.0	16.0	108.0	0.0	156.0	32.0	194.0
139	66.0	4.0	43.0	0.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
3329	1463.0	2.0	0.0	1.0	0.0	1.0	1.0	7.0	99.0	1.0	222.0	4.0	194.0
1547	541.0	4.0	13.0	1.0	2.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
2899	1005.0	4.0	26.0	0.0	3.0	1.0	1.0	12.0	108.0	0.0	58.0	28.0	194.0
4608	1464.0	3.0	0.0	1.0	2.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
2268	541.0	2.0	14.0	1.0	3.0	1.0	1.0	16.0	108.0	0.0	0.0	36.0	194.0
1335	413.0	4.0	40.0	1.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
3352	1463.0	4.0	0.0	1.0	3.0	1.0	2.0	6.0	108.0	0.0	225.0	12.0	194.0
3196	1438.0	2.0	0.0	1.0	3.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
1581	413.0	4.0	42.0	0.0	3.0	1.0	1.0	11.0	108.0	0.0	0.0	36.0	194.0
2990	1438.0	3.0	11.0	1.0	3.0	1.0	1.0	6.0	108.0	0.0	183.0	68.0	194.0
534	181.0	4.0	24.0	0.0	3.0	1.0	1.0	6.0	130.0	3.0	41.0	47.0	194.0
1028	413.0	3.0	42.0	1.0	3.0	0.0	1.0	16.0	108.0	0.0	231.0	62.0	194.0
1025	413.0	3.0	42.0	0.0	0.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
4087	1464.0	4.0	0.0	1.0	2.0	0.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
4405	1464.0	4.0	0.0	0.0	2.0	0.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
3705	1463.0	4.0	0.0	1.0	0.0	1.0	1.0	6.0	108.0	0.0	231.0	62.0	194.0
4396	1464.0	4.0	0.0	0.0	2.0	1.0	1.0	16.0	108.0	0.0	216.0	67.0	194.0
933	413.0	3.0	40.0	0.0	3.0	1.0	1.0	16.0	108.0	0.0	216.0	67.0	194.0
876	413.0	4.0	41.0	1.0	3.0	1.0	1.0	16.0	108.0	0.0	216.0	67.0	194.0
2550	541.0	4.0	14.0	1.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
3026	1438.0	2.0	3.0	1.0	0.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
4304	1464.0	4.0	3.0	1.0	2.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
506	181.0	4.0	24.0	1.0	3.0	1.0	1.0	6.0	108.0	0.0	231.0	62.0	194.0
1183	413.0	4.0	42.0	0.0	0.0	2.0	1.0	3.0	108.0	0.0	216.0	67.0	194.0
317	181.0	4.0	23.0	1.0	3.0	1.0	1.0	7.0	108.0	0.0	11.0	16.0	194.0
1531	413.0	4.0	42.0	0.0	3.0	1.0	1.0	12.0	99.0	1.0	0.0	36.0	194.0
2183	541.0	4.0	9.0	1.0	0.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
3400	1463.0	4.0	0.0	0.0	0.0	2.0	1.0	16.0	108.0	0.0	216.0	67.0	194.0
731	181.0	2.0	24.0	1.0	2.0	3.0	1.0	7.0	108.0	0.0	231.0	62.0	194.0
3425	1463.0	4.0	13.0	1.0	2.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
2403	541.0	4.0	13.0	1.0	2.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
4483	1464.0	4.0	12.0	0.0	2.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
2659	1005.0	4.0	26.0	0.0	3.0	1.0	1.0	12.0	99.0	1.0	0.0	36.0	194.0
3559	1463.0	4.0	11.0	0.0	3.0	1.0	1.0	11.0	108.0	0.0	193.0	20.0	194.0
2468	541.0	4.0	13.0	1.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
221	181.0	4.0	24.0	1.0	3.0	1.0	0.0	7.0	108.0	0.0	213.0	64.0	194.0
3228	1438.0	4.0	0.0	0.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
4280	1464.0	1.0	0.0	1.0	3.0	1.0	1.0	16.0	130.0	3.0	0.0	36.0	194.0
2549	541.0	4.0	13.0	1.0	3.0	1.0	1.0	16.0	108.0	0.0	231.0	62.0	194.0
1583	413.0	3.0	42.0	1.0	3.0	1.0	1.0	7.0	99.0	1.0	58.0	28.0	194.0
970	413.0	3.0	42.0	0.0	3.0	1.0	1.0	6.0	99.0	1.0	216.0	67.0	194.0
2692	1005.0	4.0	26.0	0.0	3.0	1.0	1.0	16.0	108.0	0.0	0.0	36.0	194.0
3773	1463.0	4.0	0.0	1.0	3.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
4426	1464.0	3.0	0.0	1.0	2.0	1.0	1.0	7.0	99.0	1.0	0.0	36.0	194.0
2126	541.0	4.0	11.0	1.0	3.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
4503	1464.0	4.0	0.0	1.0	2.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
4510	1464.0	4.0	0.0	0.0	3.0	1.0	1.0	12.0	99.0	1.0	216.0	67.0	194.0
3507	1463.0	2.0	0.0	0.0	3.0	1.0	1.0	7.0	108.0	0.0	193.0	20.0	194.0
4119	1464.0	2.0	0.0	1.0	2.0	1.0	1.0	7.0	108.0	0.0	58.0	28.0	194.0
100	66.0	4.0	36.0	0.0	3.0	1.0	1.0	7.0	108.0	0.0	213.0	64.0	194.0
1788	541.0	4.0	16.0	0.0	2.0	2.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
1312	413.0	4.0	41.0	0.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
1623	541.0	3.0	13.0	1.0	3.0	1.0	1.0	6.0	108.0	0.0	231.0	62.0	194.0
3581	1463.0	4.0	46.0	0.0	3.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
2765	1005.0	4.0	26.0	0.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
1329	413.0	3.0	42.0	1.0	3.0	1.0	1.0	6.0	108.0	0.0	216.0	67.0	194.0
2363	541.0	3.0	17.0	1.0	3.0	1.0	1.0	16.0	108.0	0.0	0.0	36.0	194.0
4200	1464.0	4.0	0.0	1.0	0.0	1.0	1.0	7.0	108.0	0.0	216.0	67.0	194.0
2840	1005.0	3.0	46.0	1.0	3.0	1.0	1.0	6.0	108.0	0.0	231.0	62.0	194.0
3704	1463.0	3.0	46.0	0.0	2.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
920	413.0	4.0	41.0	0.0	3.0	1.0	1.0	6.0	99.0	1.0	0.0	36.0	194.0
575	181.0	3.0	23.0	1.0	3.0	1.0	1.0	7.0	99.0	1.0	225.0	12.0	194.0
2858	1005.0	4.0	26.0	1.0	3.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
2350	541.0	4.0	12.0	0.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
2042	541.0	4.0	46.0	1.0	3.0	1.0	1.0	16.0	108.0	0.0	216.0	67.0	194.0
2402	541.0	4.0	12.0	0.0	3.0	1.0	1.0	3.0	108.0	0.0	41.0	47.0	194.0
3183	1438.0	4.0	0.0	1.0	2.0	1.0	1.0	7.0	99.0	1.0	0.0	36.0	194.0
3804	1463.0	2.0	11.0	0.0	2.0	1.0	1.0	6.0	108.0	0.0	193.0	20.0	194.0
3226	1438.0	4.0	11.0	0.0	3.0	1.0	1.0	6.0	99.0	1.0	0.0	36.0	194.0
3954	1464.0	3.0	3.0	0.0	2.0	1.0	1.0	6.0	108.0	0.0	216.0	67.0	194.0
4166	1464.0	4.0	0.0	0.0	2.0	1.0	1.0	7.0	108.0	0.0	217.0	61.0	194.0
1208	413.0	4.0	42.0	0.0	3.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
2420	541.0	4.0	12.0	1.0	3.0	1.0	1.0	7.0	99.0	1.0	216.0	67.0	194.0
1196	413.0	4.0	36.0	1.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
3521	1463.0	4.0	0.0	1.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
3840	1463.0	4.0	12.0	0.0	3.0	1.0	1.0	16.0	99.0	1.0	216.0	67.0	194.0
1464	413.0	4.0	40.0	1.0	3.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
2253	541.0	4.0	14.0	1.0	3.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
2176	541.0	3.0	13.0	1.0	0.0	1.0	2.0	6.0	108.0	0.0	0.0	36.0	194.0
1317	413.0	4.0	41.0	0.0	3.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
1339	413.0	3.0	42.0	0.0	3.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
3845	1463.0	4.0	0.0	0.0	3.0	1.0	0.0	7.0	108.0	0.0	222.0	4.0	194.0
2888	1005.0	4.0	46.0	1.0	3.0	1.0	1.0	15.0	108.0	0.0	0.0	36.0	194.0
1147	413.0	4.0	40.0	0.0	3.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
1935	541.0	2.0	13.0	1.0	0.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
1660	541.0	4.0	9.0	1.0	0.0	2.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
368	181.0	4.0	24.0	0.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
3675	1463.0	4.0	11.0	0.0	3.0	1.0	1.0	7.0	99.0	1.0	0.0	36.0	194.0
2076	541.0	3.0	13.0	0.0	3.0	1.0	1.0	16.0	99.0	1.0	217.0	61.0	194.0
4530	1464.0	4.0	0.0	0.0	0.0	1.0	1.0	7.0	108.0	0.0	216.0	67.0	194.0
3057	1438.0	3.0	11.0	0.0	0.0	1.0	1.0	9.0	99.0	1.0	0.0	36.0	194.0
3981	1464.0	2.0	0.0	1.0	0.0	1.0	1.0	6.0	108.0	0.0	193.0	20.0	194.0
2123	541.0	3.0	13.0	1.0	3.0	1.0	1.0	7.0	108.0	0.0	173.0	39.0	194.0
4474	1464.0	2.0	0.0	0.0	2.0	1.0	1.0	6.0	99.0	1.0	0.0	36.0	194.0
2132	541.0	4.0	16.0	0.0	3.0	1.0	1.0	7.0	108.0	0.0	231.0	62.0	194.0
2546	541.0	4.0	12.0	0.0	3.0	1.0	1.0	9.0	99.0	1.0	222.0	4.0	194.0
727	181.0	4.0	23.0	0.0	3.0	1.0	1.0	6.0	108.0	0.0	231.0	62.0	194.0
4693	1464.0	3.0	0.0	0.0	2.0	0.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
1449	413.0	3.0	42.0	0.0	0.0	1.0	1.0	7.0	108.0	0.0	58.0	28.0	194.0
2206	541.0	3.0	14.0	0.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
3234	1438.0	4.0	0.0	1.0	2.0	1.0	1.0	7.0	99.0	1.0	0.0	36.0	194.0
380	181.0	4.0	23.0	0.0	3.0	1.0	1.0	6.0	108.0	0.0	231.0	62.0	194.0
754	181.0	3.0	23.0	1.0	3.0	1.0	1.0	6.0	99.0	1.0	0.0	36.0	194.0
4293	1464.0	3.0	46.0	0.0	3.0	1.0	1.0	6.0	108.0	0.0	231.0	62.0	194.0
4274	1464.0	4.0	18.0	0.0	1.0	0.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
3912	1464.0	4.0	4.0	0.0	0.0	2.0	1.0	7.0	108.0	0.0	216.0	67.0	194.0
1337	413.0	4.0	40.0	0.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
4604	1464.0	4.0	11.0	0.0	2.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
4100	1464.0	3.0	3.0	0.0	0.0	1.0	1.0	7.0	108.0	0.0	231.0	62.0	194.0
1642	541.0	4.0	9.0	1.0	3.0	1.0	1.0	7.0	108.0	0.0	227.0	38.0	194.0
3462	1463.0	3.0	46.0	1.0	2.0	1.0	1.0	6.0	108.0	0.0	193.0	20.0	194.0
4743	1464.0	4.0	0.0	0.0	0.0	1.0	1.0	7.0	99.0	1.0	231.0	62.0	194.0
2729	1005.0	4.0	26.0	1.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
2058	541.0	4.0	9.0	0.0	0.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
1730	541.0	3.0	14.0	1.0	3.0	1.0	1.0	6.0	99.0	1.0	0.0	36.0	194.0
3452	1463.0	4.0	0.0	0.0	3.0	1.0	1.0	7.0	108.0	0.0	222.0	4.0	194.0
3191	1438.0	3.0	3.0	1.0	0.0	1.0	1.0	6.0	108.0	0.0	231.0	62.0	194.0
3312	1438.0	3.0	12.0	1.0	3.0	1.0	1.0	7.0	108.0	0.0	193.0	20.0	194.0
4547	1464.0	3.0	0.0	0.0	2.0	1.0	1.0	16.0	99.0	1.0	0.0	36.0	194.0
739	181.0	4.0	24.0	0.0	3.0	1.0	1.0	16.0	99.0	1.0	0.0	36.0	194.0
4509	1464.0	4.0	0.0	0.0	2.0	2.0	1.0	6.0	99.0	1.0	231.0	62.0	194.0
3873	1463.0	4.0	0.0	1.0	3.0	1.0	0.0	7.0	108.0	0.0	213.0	64.0	194.0
1856	541.0	4.0	13.0	1.0	2.0	1.0	1.0	6.0	99.0	1.0	216.0	67.0	194.0
4209	1464.0	2.0	0.0	0.0	2.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
2574	541.0	4.0	17.0	0.0	3.0	1.0	1.0	7.0	108.0	0.0	225.0	12.0	194.0
4553	1464.0	4.0	3.0	1.0	2.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
1577	413.0	4.0	40.0	1.0	3.0	1.0	1.0	16.0	108.0	0.0	0.0	36.0	194.0
3442	1463.0	2.0	46.0	1.0	0.0	1.0	1.0	6.0	99.0	1.0	54.0	55.0	194.0
1426	413.0	3.0	42.0	0.0	3.0	1.0	1.0	15.0	108.0	0.0	58.0	28.0	194.0
3846	1463.0	3.0	11.0	1.0	2.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
3882	1463.0	3.0	0.0	1.0	3.0	1.0	1.0	7.0	108.0	0.0	193.0	20.0	194.0
734	181.0	3.0	23.0	1.0	2.0	3.0	1.0	11.0	99.0	1.0	204.0	70.0	194.0
1331	413.0	4.0	41.0	0.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
3618	1463.0	3.0	11.0	0.0	0.0	1.0	0.0	7.0	108.0	0.0	213.0	64.0	194.0
840	181.0	3.0	23.0	1.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
145	66.0	4.0	43.0	0.0	3.0	1.0	0.0	9.0	108.0	0.0	0.0	36.0	194.0
2556	541.0	4.0	13.0	1.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
725	181.0	4.0	24.0	0.0	3.0	1.0	1.0	4.0	99.0	1.0	0.0	36.0	194.0
3636	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3367	1463.0	3.0	12.0	1.0	3.0	1.0	2.0	11.0	108.0	0.0	0.0	36.0	194.0
4039	1464.0	3.0	3.0	0.0	0.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
3876	1463.0	3.0	11.0	1.0	3.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
814	181.0	4.0	24.0	0.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
4441	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4115	1464.0	4.0	6.0	0.0	0.0	2.0	1.0	7.0	108.0	0.0	231.0	62.0	194.0
2430	541.0	3.0	13.0	0.0	3.0	1.0	1.0	7.0	108.0	0.0	54.0	55.0	194.0
1760	541.0	4.0	9.0	1.0	3.0	1.0	1.0	6.0	99.0	1.0	0.0	36.0	194.0
1889	541.0	4.0	16.0	0.0	3.0	1.0	1.0	7.0	108.0	0.0	216.0	67.0	194.0
4476	1464.0	3.0	0.0	1.0	2.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
1469	413.0	4.0	40.0	1.0	3.0	1.0	1.0	6.0	108.0	0.0	58.0	28.0	194.0
283	181.0	3.0	23.0	0.0	3.0	1.0	1.0	6.0	108.0	0.0	222.0	4.0	194.0
4292	1464.0	3.0	0.0	1.0	2.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
3775	1463.0	4.0	0.0	1.0	3.0	1.0	1.0	9.0	108.0	0.0	0.0	36.0	194.0
658	181.0	4.0	23.0	1.0	3.0	1.0	1.0	16.0	108.0	0.0	0.0	36.0	194.0
299	181.0	4.0	22.0	0.0	3.0	1.0	1.0	6.0	99.0	1.0	70.0	73.0	194.0
790	181.0	4.0	23.0	0.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
3047	1438.0	3.0	0.0	1.0	3.0	1.0	1.0	6.0	108.0	0.0	231.0	62.0	194.0
3468	1463.0	3.0	11.0	1.0	3.0	1.0	2.0	6.0	108.0	0.0	225.0	12.0	194.0
1942	541.0	4.0	14.0	1.0	3.0	1.0	1.0	16.0	108.0	0.0	216.0	67.0	194.0
2458	541.0	4.0	12.0	1.0	3.0	1.0	1.0	3.0	108.0	0.0	0.0	36.0	194.0
1920	541.0	4.0	17.0	0.0	3.0	2.0	1.0	6.0	130.0	3.0	41.0	47.0	194.0
940	413.0	3.0	41.0	1.0	3.0	1.0	1.0	6.0	130.0	3.0	39.0	30.0	194.0
2426	541.0	4.0	12.0	0.0	3.0	1.0	1.0	3.0	108.0	0.0	0.0	36.0	194.0
2784	1005.0	4.0	26.0	1.0	3.0	1.0	1.0	16.0	108.0	0.0	0.0	36.0	194.0
4036	1464.0	4.0	0.0	1.0	3.0	2.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
1636	541.0	4.0	12.0	1.0	3.0	1.0	1.0	7.0	108.0	0.0	222.0	4.0	194.0
2695	1005.0	4.0	26.0	0.0	3.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
1471	413.0	4.0	42.0	0.0	3.0	1.0	1.0	16.0	108.0	0.0	0.0	36.0	194.0
3655	1463.0	3.0	11.0	1.0	3.0	1.0	1.0	7.0	99.0	1.0	216.0	67.0	194.0
3908	1464.0	3.0	3.0	1.0	1.0	0.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
670	181.0	4.0	23.0	1.0	3.0	1.0	1.0	6.0	99.0	1.0	0.0	36.0	194.0
1568	413.0	4.0	40.0	1.0	3.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
3743	1463.0	4.0	0.0	1.0	3.0	1.0	1.0	16.0	108.0	0.0	0.0	36.0	194.0
1666	541.0	3.0	9.0	1.0	3.0	1.0	1.0	6.0	108.0	0.0	193.0	20.0	194.0
1741	541.0	4.0	9.0	0.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
1109	413.0	4.0	42.0	0.0	3.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
2307	541.0	4.0	13.0	1.0	0.0	1.0	1.0	3.0	108.0	0.0	224.0	6.0	194.0
4298	1464.0	3.0	0.0	1.0	2.0	1.0	1.0	7.0	108.0	0.0	216.0	67.0	194.0
3038	1438.0	3.0	3.0	1.0	0.0	1.0	1.0	7.0	108.0	0.0	231.0	62.0	194.0
2112	541.0	4.0	13.0	0.0	0.0	1.0	1.0	6.0	108.0	0.0	39.0	30.0	194.0
3276	1438.0	4.0	11.0	1.0	3.0	1.0	1.0	16.0	108.0	0.0	231.0	62.0	194.0
226	181.0	4.0	23.0	0.0	0.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
687	181.0	4.0	23.0	1.0	3.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
1089	413.0	3.0	40.0	1.0	0.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
1703	541.0	4.0	16.0	1.0	3.0	1.0	1.0	7.0	108.0	0.0	216.0	67.0	194.0
1982	541.0	4.0	13.0	1.0	2.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
4454	1464.0	4.0	11.0	1.0	3.0	1.0	2.0	7.0	108.0	0.0	0.0	36.0	194.0
186	181.0	4.0	23.0	0.0	3.0	1.0	1.0	16.0	130.0	3.0	41.0	47.0	194.0
879	413.0	4.0	41.0	0.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
8	66.0	4.0	37.0	0.0	3.0	1.0	1.0	11.0	108.0	0.0	0.0	36.0	194.0
3423	1463.0	4.0	11.0	0.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
2306	541.0	3.0	10.0	1.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
2758	1005.0	4.0	26.0	1.0	3.0	1.0	1.0	7.0	99.0	1.0	0.0	36.0	194.0
2559	541.0	4.0	9.0	0.0	3.0	2.0	1.0	6.0	108.0	0.0	216.0	67.0	194.0
3188	1438.0	4.0	0.0	0.0	0.0	1.0	1.0	6.0	108.0	0.0	193.0	20.0	194.0
3795	1463.0	4.0	11.0	0.0	3.0	1.0	1.0	7.0	108.0	0.0	216.0	67.0	194.0
2181	541.0	4.0	16.0	1.0	3.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
942	413.0	4.0	42.0	0.0	0.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
2413	541.0	4.0	9.0	0.0	3.0	1.0	1.0	16.0	108.0	0.0	231.0	62.0	194.0
2009	541.0	4.0	15.0	0.0	3.0	2.0	1.0	6.0	108.0	0.0	193.0	20.0	194.0
2442	541.0	4.0	9.0	1.0	3.0	1.0	1.0	7.0	99.0	1.0	0.0	36.0	194.0
1336	413.0	3.0	40.0	1.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
2677	1005.0	4.0	26.0	1.0	3.0	1.0	1.0	16.0	108.0	0.0	0.0	36.0	194.0
769	181.0	4.0	23.0	1.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
4144	1464.0	4.0	5.0	1.0	0.0	1.0	2.0	16.0	108.0	0.0	41.0	47.0	194.0
3798	1463.0	4.0	0.0	0.0	2.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
1236	413.0	4.0	42.0	0.0	3.0	1.0	1.0	9.0	108.0	0.0	0.0	36.0	194.0
1448	413.0	3.0	42.0	0.0	3.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
1527	413.0	4.0	42.0	1.0	0.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
136	66.0	3.0	36.0	0.0	3.0	1.0	1.0	7.0	108.0	0.0	231.0	62.0	194.0
440	181.0	4.0	23.0	0.0	3.0	1.0	1.0	9.0	99.0	1.0	222.0	4.0	194.0
3028	1438.0	4.0	0.0	0.0	2.0	2.0	1.0	16.0	108.0	0.0	41.0	47.0	194.0
4722	1464.0	4.0	0.0	0.0	2.0	0.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
760	181.0	4.0	24.0	0.0	2.0	3.0	1.0	16.0	108.0	0.0	0.0	36.0	194.0
3941	1464.0	0.0	3.0	1.0	0.0	0.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
4359	1464.0	4.0	0.0	1.0	2.0	2.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
802	181.0	4.0	24.0	1.0	3.0	1.0	1.0	16.0	99.0	1.0	0.0	36.0	194.0
3949	1464.0	4.0	0.0	1.0	3.0	2.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
713	181.0	4.0	23.0	0.0	3.0	1.0	1.0	7.0	108.0	0.0	228.0	65.0	194.0
4708	1464.0	4.0	8.0	0.0	3.0	1.0	1.0	3.0	108.0	0.0	54.0	55.0	194.0
993	413.0	3.0	42.0	1.0	3.0	1.0	2.0	6.0	108.0	0.0	0.0	36.0	194.0
4429	1464.0	3.0	16.0	0.0	3.0	1.0	1.0	6.0	108.0	0.0	231.0	62.0	194.0
3536	1463.0	4.0	0.0	0.0	3.0	1.0	1.0	7.0	99.0	1.0	227.0	38.0	194.0
3118	1438.0	2.0	3.0	0.0	0.0	1.0	1.0	6.0	99.0	1.0	58.0	28.0	194.0
4477	1464.0	4.0	11.0	1.0	2.0	1.0	2.0	7.0	108.0	0.0	71.0	27.0	194.0
653	181.0	4.0	24.0	0.0	0.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
3865	1463.0	4.0	12.0	1.0	2.0	1.0	1.0	6.0	108.0	0.0	108.0	31.0	194.0
1386	413.0	4.0	44.0	0.0	3.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
772	181.0	3.0	24.0	0.0	2.0	3.0	2.0	6.0	108.0	0.0	54.0	55.0	194.0
4697	1464.0	4.0	0.0	0.0	2.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
117	66.0	4.0	43.0	0.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
2110	541.0	3.0	12.0	1.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
211	181.0	4.0	23.0	0.0	3.0	1.0	1.0	7.0	99.0	1.0	0.0	36.0	194.0
544	181.0	4.0	23.0	1.0	3.0	1.0	2.0	3.0	108.0	0.0	0.0	36.0	194.0
1300	413.0	3.0	42.0	0.0	0.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
2536	541.0	3.0	10.0	1.0	3.0	2.0	1.0	6.0	108.0	0.0	216.0	67.0	194.0
3500	1463.0	4.0	11.0	0.0	3.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
3094	1438.0	3.0	11.0	0.0	0.0	1.0	1.0	7.0	108.0	0.0	231.0	62.0	194.0
1128	413.0	4.0	40.0	0.0	3.0	1.0	2.0	7.0	99.0	1.0	0.0	36.0	194.0
2504	541.0	4.0	12.0	1.0	3.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
294	181.0	3.0	24.0	0.0	0.0	1.0	1.0	6.0	99.0	1.0	0.0	36.0	194.0
1150	413.0	4.0	36.0	0.0	3.0	1.0	1.0	16.0	108.0	0.0	0.0	36.0	194.0
770	181.0	4.0	23.0	0.0	3.0	1.0	2.0	7.0	108.0	0.0	0.0	36.0	194.0
34	66.0	4.0	43.0	0.0	3.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
227	181.0	4.0	24.0	1.0	3.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
2170	541.0	4.0	12.0	1.0	3.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
806	181.0	3.0	23.0	1.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
2649	1005.0	4.0	26.0	1.0	3.0	1.0	0.0	6.0	99.0	1.0	0.0	36.0	194.0
860	413.0	3.0	46.0	1.0	3.0	1.0	2.0	6.0	108.0	0.0	0.0	36.0	194.0
1476	413.0	4.0	42.0	0.0	3.0	1.0	1.0	7.0	108.0	0.0	0.0	36.0	194.0
4720	1464.0	4.0	12.0	1.0	3.0	1.0	2.0	7.0	108.0	0.0	231.0	62.0	194.0
4188	1464.0	3.0	3.0	1.0	3.0	1.0	2.0	6.0	108.0	0.0	231.0	62.0	194.0
1903	541.0	4.0	12.0	1.0	3.0	2.0	1.0	6.0	108.0	0.0	193.0	20.0	194.0
2284	541.0	3.0	14.0	1.0	0.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
2813	1005.0	4.0	26.0	1.0	3.0	1.0	1.0	7.0	130.0	3.0	41.0	47.0	194.0
500	181.0	4.0	23.0	1.0	3.0	1.0	1.0	6.0	108.0	0.0	0.0	36.0	194.0
4550	1464.0	4.0	0.0	1.0	2.0	2.0	1.0	6.0	99.0	1.0	108.0	31.0	194.0
4030	1464.0	4.0	0.0	0.0	2.0	2.0	1.0	16.0	108.0	0.0	0.0	36.0	194.0

In [ ]:

```
xgb1.predict(X_test)
```

In [112…

```
#need to drop the corresponding target in the testdataset here
test.drop(['Length_of_Stay'],axis=1,inplace=True)
preds=xgb1.predict(test)
```

In [ ]:

```
for index in list(preds.index):
    if (train['Length_of_Stay'][index] <= 6):
        train['Length_of_Stay'][index] = 1
    elif (train['Length_of_Stay'][index] >= 6)&(train['Length_of_Stay'][index] <= 10):
        train['Length_of_Stay'][index] = 2
    elif (train['Length_of_Stay'][index] > 10)&(train['Length_of_Stay'][index] <= 12):
        train['Length_of_Stay'][index] = 3
    elif (train['Length_of_Stay'][index] > 12):
        train['Length_of_Stay'][index] = 4
    else:
        index+1
```

In [ ]:

In [112…

```
print(preds)
```

```
[1. 1. 2. 1. 4. 1. 1. 1. 1. 2. 1. 1. 1. 1. 1. 1. 1. 1. 1. 2. 1. 1. 4. 1.
 1. 1. 2. 1. 4. 2. 4. 4. 2. 1. 1. 1. 4. 1. 1. 1. 1. 1. 1. 1. 3. 4. 1. 1. 1.
 2. 1. 1. 4. 1. 4. 2. 1. 1. 1. 1. 1. 2. 4. 1. 1. 1. 1. 1. 1. 1. 1. 4. 1.
 2. 4. 2. 4. 1. 1. 2. 2. 1. 1. 1. 1. 1. 1. 1. 1. 4. 1. 1. 1. 1. 1. 2. 2. 1.
 4. 2. 2. 1. 1. 1. 1. 4. 4. 1. 1. 4. 1. 1. 1. 1. 2. 1. 2. 4. 1. 1. 1. 1.
 3. 1. 2. 1. 3. 4. 1. 1. 1. 1. 2. 1. 1. 1. 1. 1. 1. 2. 1. 1. 2. 1. 1. 2.
 1. 1. 2. 1. 1. 1. 1. 4. 1. 2. 1. 1. 3. 1. 3. 2. 1. 3. 1. 1. 2. 1. 1. 2.
 2. 1. 1. 1. 2. 1. 4. 2. 1. 4. 1. 1. 1. 3. 1. 1. 4. 1. 1. 2. 1. 1. 2. 1. 1.
 1. 1. 1. 2. 1. 1. 1. 1. 2. 1. 1. 4. 1. 2. 2. 2. 4. 3. 1. 1. 2. 1. 2.
 1. 1. 2. 4. 1. 2. 1. 1. 1. 1. 4. 1. 3. 1. 1. 2. 2. 1. 1. 1. 1. 1. 4. 1. 3.
 1. 1. 2. 1. 1. 4. 1. 2. 2. 1. 4. 1. 2. 4. 1. 4. 1. 1. 1. 1. 1. 3. 1.
 4. 1. 1. 1. 1. 2. 1. 1. 1. 1. 1. 1. 1. 3. 1. 1. 4. 4. 2. 1. 1. 3. 4. 1.
 1. 2. 1. 1. 1. 1. 1. 4. 2. 1. 4. 1. 2. 2. 1. 1. 2. 4. 1. 1. 3. 1. 4. 1.
 4. 1. 1. 1. 1. 2. 1. 1. 1. 1. 2. 1. 1. 2. 2. 1. 1. 1. 1. 1. 2. 2. 1. 4.
 2. 1. 4. 2. 1. 1. 2. 1. 3. 1. 1. 2. 1. 1. 4. 1. 1. 1. 1. 1. 1. 1. 2. 1.
 1. 1. 4. 1. 2. 1. 1. 1. 1. 1. 2. 1. 1. 2. 1. 2. 1. 4. 1. 1. 1. 2. 1. 1.
 4. 1. 2. 1. 1. 2. 1. 4. 2. 4. 2. 1. 2. 4. 2. 1. 1. 1. 2. 1. 1. 1. 1. 1.
 3. 1. 1. 1. 1. 2. 1. 2. 1. 1. 1. 2. 1. 1. 4. 1. 2. 1. 1. 1. 2. 1. 1. 2.
 1. 1. 1. 1. 1. 4. 2. 1. 1. 2. 1. 1. 2. 1. 1. 1. 1. 1. 1. 1. 3. 1. 4. 2.
 1. 1. 1. 1. 1. 1. 2. 1. 1. 1. 4. 2. 1. 1. 1. 2. 1. 1. 1. 1.]
```

In [114…

```
#key
#1 - '<6' days
#2- '6 to 10' days
#3- '10 to 12'days
#4- '>12'days
```

In [ ]: