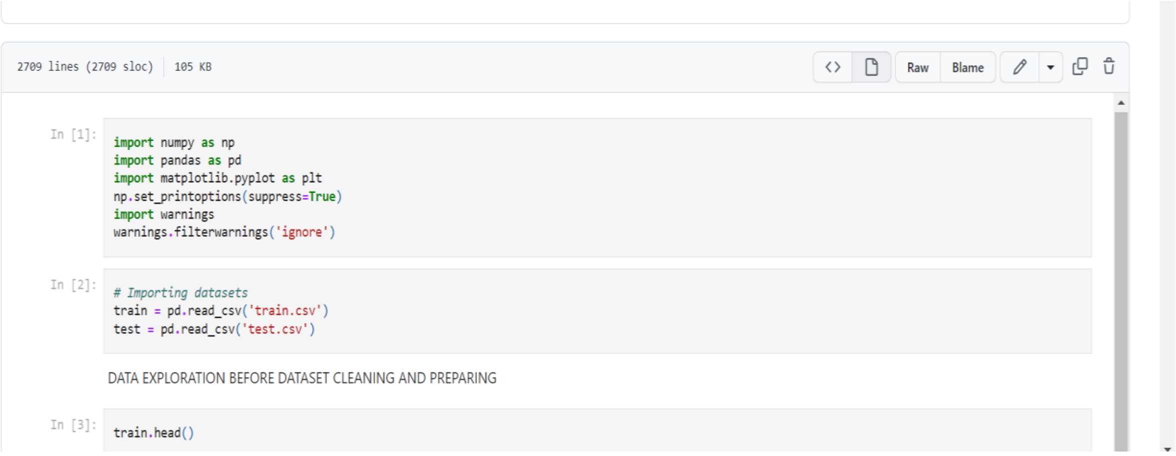
SOURCE CODE

ANALYTICS FOR HOSPITAL’S HEALTH CARE DATA

TEAM ID	**PNT2022TMID15702**
TEAM MEMBERS	**DEEPIKA S** **DIVYA A** **ABURVA SEMA I G** **FEMINA PRIYADHARSHINI X**

IMPORT THE DATASET:



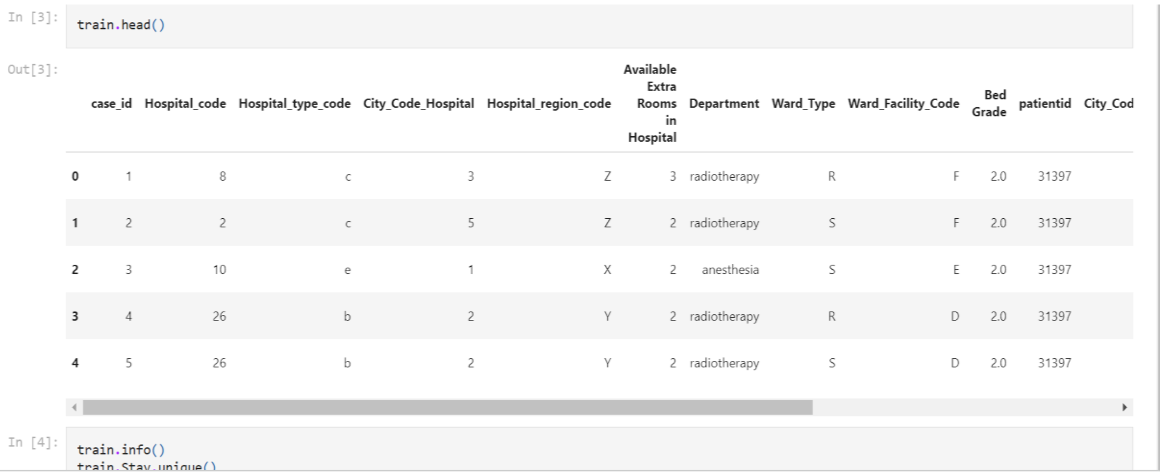
```
In [1]:
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
np.set_printoptions(suppress=True)
import warnings
warnings.filterwarnings('ignore')
```

```
In [2]:
# Importing datasets
train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')
```

DATA EXPLORATION BEFORE DATASET CLEANING AND PREPARING

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

DATA EXPLORATION BEFORE DATASET CLEANING AND PREPARING:

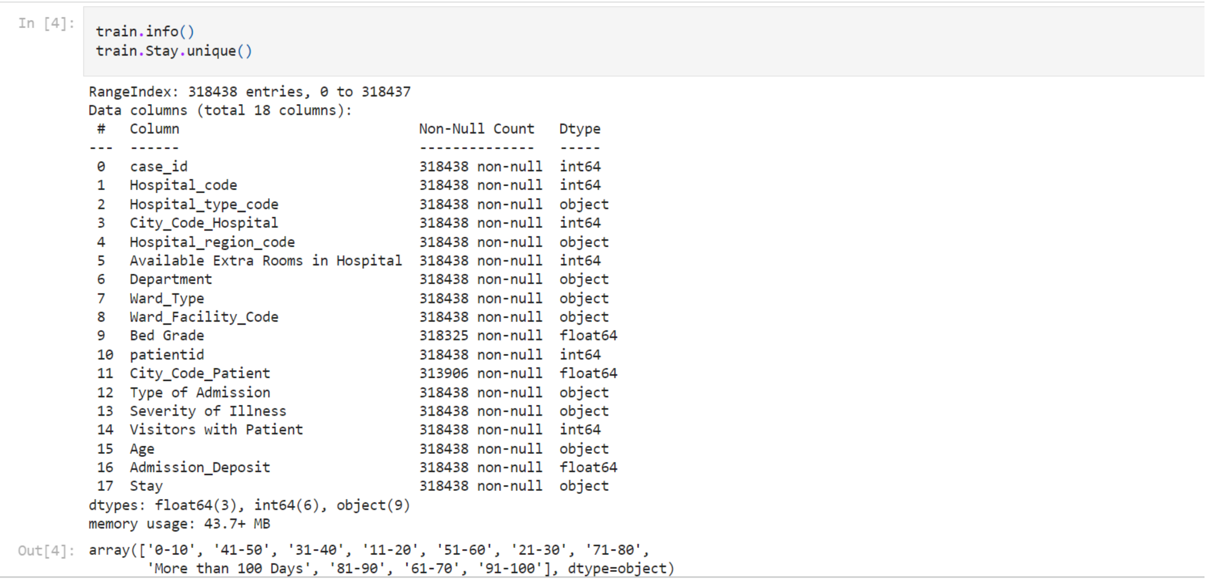


```
In [3]:
train.head()
```

Out[3]:

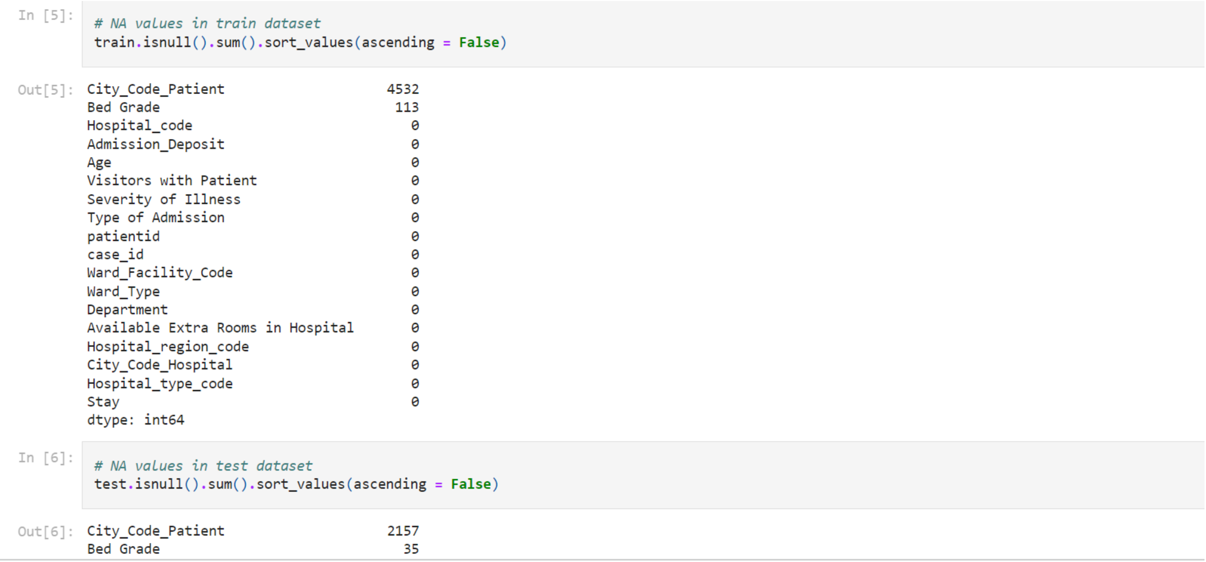
	case_id	Hospital_code	Hospital_type_code	City_Code_Hospital	Hospital_region_code	Available Extra Rooms in Hospital	Department	Ward_Type	Ward_Facility_Code	Bed Grade	patientid	City_Cod
0	1	8	c	3	Z	3	radiotherapy	R	F	2.0	31397	
1	2	2	c	5	Z	2	radiotherapy	S	F	2.0	31397	
2	3	10	e	1	X	2	anesthesia	S	E	2.0	31397	
3	4	26	b	2	Y	2	radiotherapy	R	D	2.0	31397	
4	5	26	b	2	Y	2	radiotherapy	S	D	2.0	31397	

```
In [4]:
train.info()
train.Stay.unique()
```

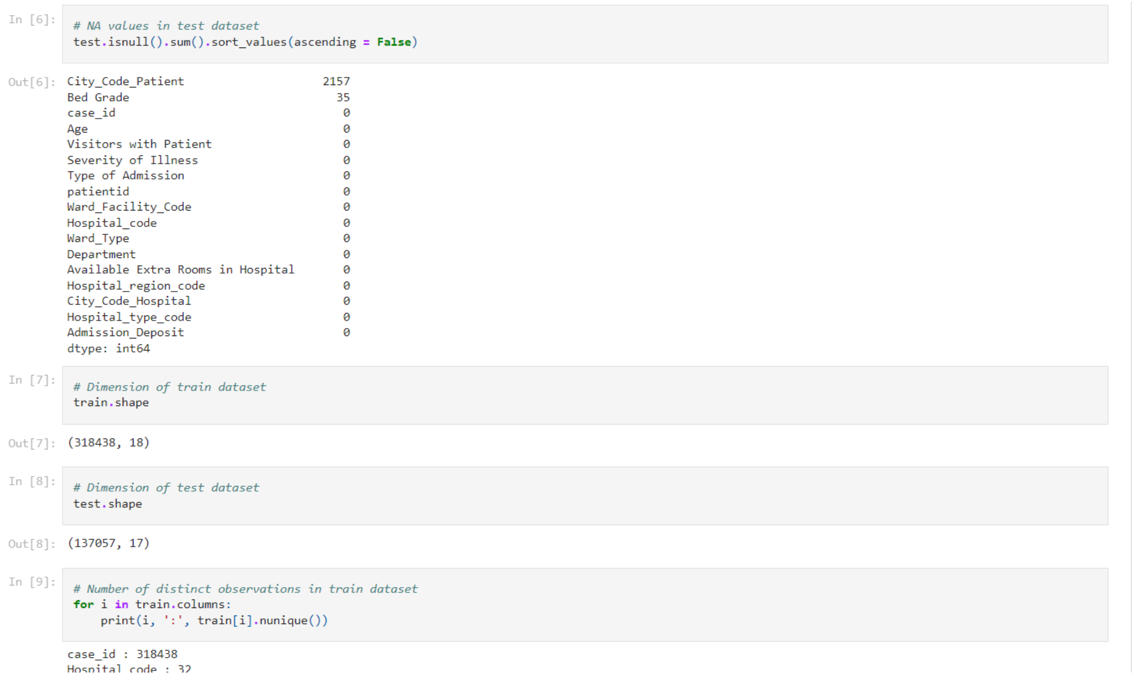


```
In [4]:  train.info()
         train.Stay.unique()

         RangeIndex: 318438 entries, 0 to 318437
         Data columns (total 18 columns):
          #   Column                             Non-Null Count   Dtype
         ---  ------                             --------------   -----
          0   case_id                            318438 non-null  int64
          1   Hospital_code                      318438 non-null  int64
          2   Hospital_type_code                 318438 non-null  object
          3   City_Code_Hospital                 318438 non-null  int64
          4   Hospital_region_code               318438 non-null  object
          5   Available Extra Rooms in Hospital  318438 non-null  int64
          6   Department                         318438 non-null  object
          7   Ward_Type                          318438 non-null  object
          8   Ward_Facility_Code                 318438 non-null  object
          9   Bed Grade                          318325 non-null  float64
          10  patientid                          318438 non-null  int64
          11  City_Code_Patient                  313906 non-null  float64
          12  Type of Admission                  318438 non-null  object
          13  Severity of Illness                318438 non-null  object
          14  Visitors with Patient              318438 non-null  int64
          15  Age                                318438 non-null  object
          16  Admission_Deposit                  318438 non-null  float64
          17  Stay                               318438 non-null  object
         dtypes: float64(3), int64(6), object(9)
         memory usage: 43.7+ MB

Out[4]:  array(['0-10', '41-50', '31-40', '11-20', '51-60', '21-30', '71-80',
                'More than 100 Days', '81-90', '61-70', '91-100'], dtype=object)
```

NULL VALUES IN TRAIN & TEST DATSET:

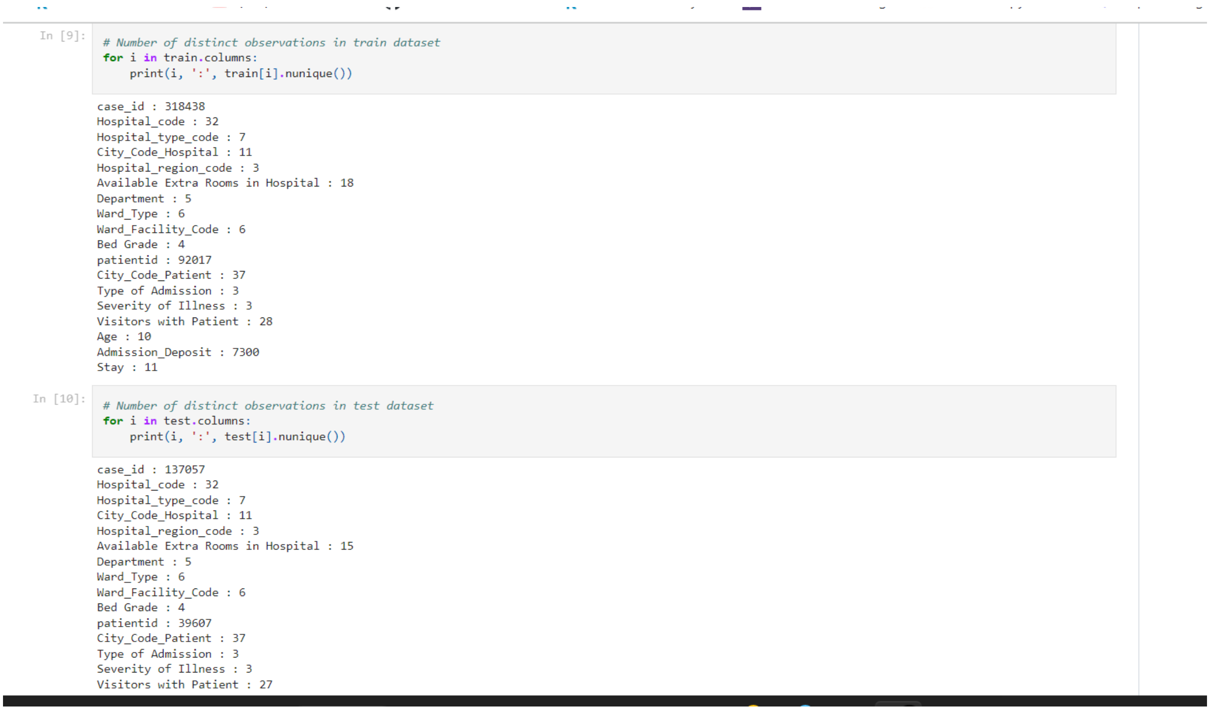


```
In [5]:  # NA values in train dataset
         train.isnull().sum().sort_values(ascending = False)

Out[5]:  City_Code_Patient                    4532
         Bed Grade                             113
         Hospital_code                           0
         Admission_Deposit                       0
         Age                                     0
         Visitors with Patient                   0
         Severity of Illness                     0
         Type of Admission                       0
         patientid                               0
         case_id                                 0
         Ward_Facility_Code                      0
         Ward_Type                               0
         Department                              0
         Available Extra Rooms in Hospital       0
         Hospital_region_code                    0
         City_Code_Hospital                      0
         Hospital_type_code                      0
         Stay                                    0
         dtype: int64

In [6]:  # NA values in test dataset
         test.isnull().sum().sort_values(ascending = False)

Out[6]:  City_Code_Patient                    2157
         Bed Grade                              35
```



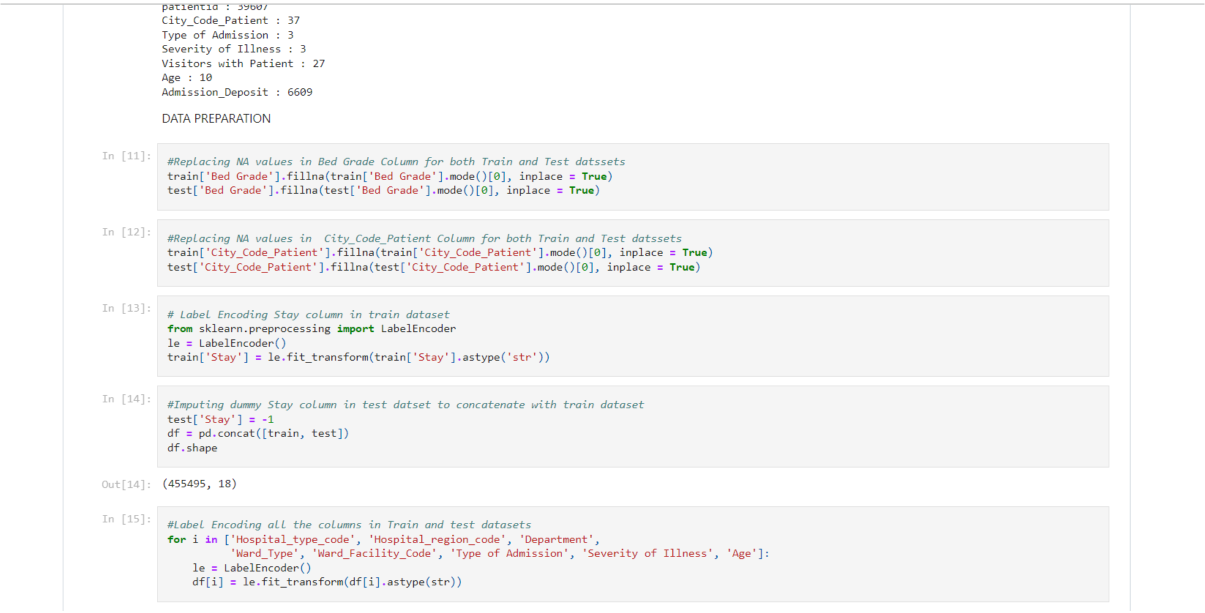
```
In [6]:  # NA values in test dataset
         test.isnull().sum().sort_values(ascending = False)

Out[6]:  City_Code_Patient                    2157
         Bed Grade                              35
         case_id                                 0
         Age                                     0
         Visitors with Patient                   0
         Severity of Illness                     0
         Type of Admission                       0
         patientid                               0
         Ward_Facility_Code                      0
         Hospital_code                           0
         Ward_Type                               0
         Department                              0
         Available Extra Rooms in Hospital       0
         Hospital_region_code                    0
         City_Code_Hospital                      0
         Hospital_type_code                      0
         Admission_Deposit                       0
         dtype: int64

In [7]:  # Dimension of train dataset
         train.shape

Out[7]:  (318438, 18)

In [8]:  # Dimension of test dataset
         test.shape

Out[8]:  (137057, 17)

In [9]:  # Number of distinct observations in train dataset
         for i in train.columns:
             print(i, ':', train[i].nunique())

         case_id : 318438
         Hospital_code : 32
```

NO OF DISTINCT OBSERVATIONS:



```
In [9]:  # Number of distinct observations in train dataset
         for i in train.columns:
             print(i, ':', train[i].nunique())

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

In [10]: # Number of distinct observations in test dataset
         for i in test.columns:
             print(i, ':', test[i].nunique())

         case_id : 137057
         Hospital_code : 32
         Hospital_type_code : 7
         City_Code_Hospital : 11
         Hospital_region_code : 3
         Available Extra Rooms in Hospital : 15
         Department : 5
         Ward_Type : 6
         Ward_Facility_Code : 6
         Bed Grade : 4
         patientid : 39607
         City_Code_Patient : 37
         Type of Admission : 3
         Severity of Illness : 3
         Visitors with Patient : 27
```

DATA PREPARATION:



```
patientid : 39607
City_Code_Patient : 37
Type of Admission : 3
Severity of Illness : 3
Visitors with Patient : 27
Age : 10
Admission_Deposit : 6609
```

DATA PREPARATION

```
In [11]:
#Replacing NA values in Bed Grade Column for both Train and Test datssets
train['Bed Grade'].fillna(train['Bed Grade'].mode()[0], inplace = True)
test['Bed Grade'].fillna(test['Bed Grade'].mode()[0], inplace = True)
```

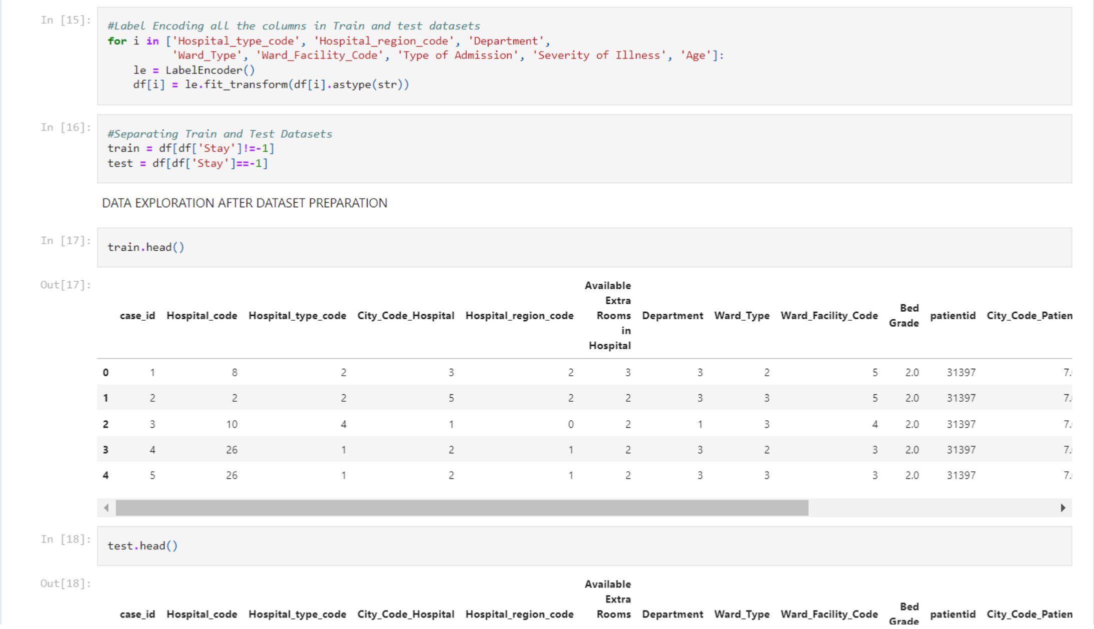
```
In [12]:
#Replacing NA values in  City_Code_Patient Column for both Train and Test datssets
train['City_Code_Patient'].fillna(train['City_Code_Patient'].mode()[0], inplace = True)
test['City_Code_Patient'].fillna(test['City_Code_Patient'].mode()[0], inplace = True)
```

```
In [13]:
# Label Encoding Stay column in train dataset
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
train['Stay'] = le.fit_transform(train['Stay'].astype('str'))
```

```
In [14]:
#Imputing dummy Stay column in test datset to concatenate with train dataset
test['Stay'] = -1
df = pd.concat([train, test])
df.shape

Out[14]: (455495, 18)
```

```
In [15]:
#Label Encoding all the columns in Train and test datasets
for i in ['Hospital_type_code', 'Hospital_region_code', 'Department',
          'Ward_Type', 'Ward_Facility_Code', 'Type of Admission', 'Severity of Illness', 'Age']:
    le = LabelEncoder()
    df[i] = le.fit_transform(df[i].astype(str))
```

DATA EXPLORATION



```
In [15]:
#Label Encoding all the columns in Train and test datasets
for i in ['Hospital_type_code', 'Hospital_region_code', 'Department',
          'Ward_Type', 'Ward_Facility_Code', 'Type of Admission', 'Severity of Illness', 'Age']:
    le = LabelEncoder()
    df[i] = le.fit_transform(df[i].astype(str))
```

```
In [16]:
#Separating Train and Test Datasets
train = df[df['Stay']!=-1]
test = df[df['Stay']==-1]
```

DATA EXPLORATION AFTER DATASET PREPARATION

```
In [17]:
train.head()
```

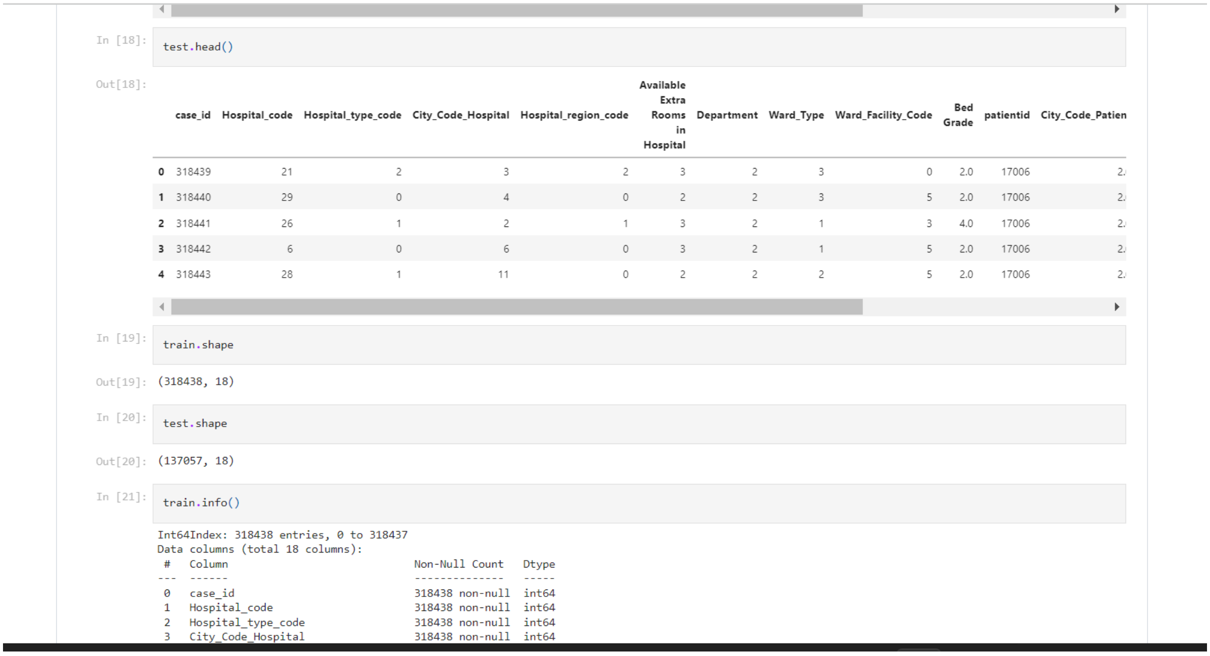
Out[17]:

	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	patientid	City_Code_Patien
0	1	8	2	3	2	3	3	2	5	2.0	31397	7.
1	2	2	2	5	2	2	3	3	5	2.0	31397	7.
2	3	10	4	1	0	2	1	3	4	2.0	31397	7.
3	4	26	1	2	1	2	3	2	3	2.0	31397	7.
4	5	26	1	2	1	2	3	3	3	2.0	31397	7.

```
In [18]:
test.head()
```

Out[18]:

	case_id	Hospital_code	Hospital_type_code	City_Code_Hospital	Hospital_region_code	Available Extra Rooms	Department	Ward_Type	Ward_Facility_Code	Bed Grade	patientid	City_Code_Patien

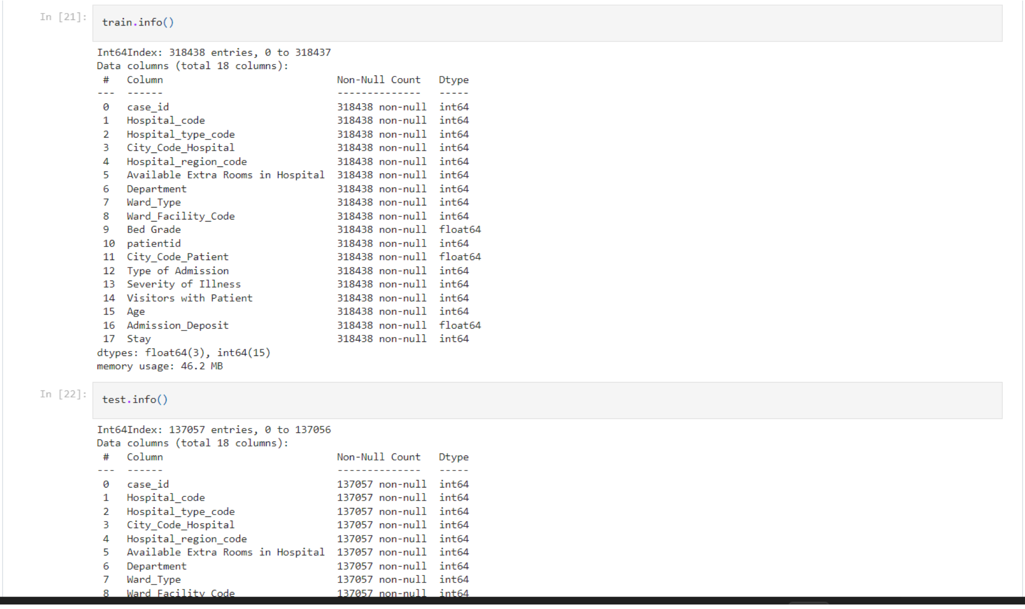


```
In [18]: test.head()
```

Out[18]:

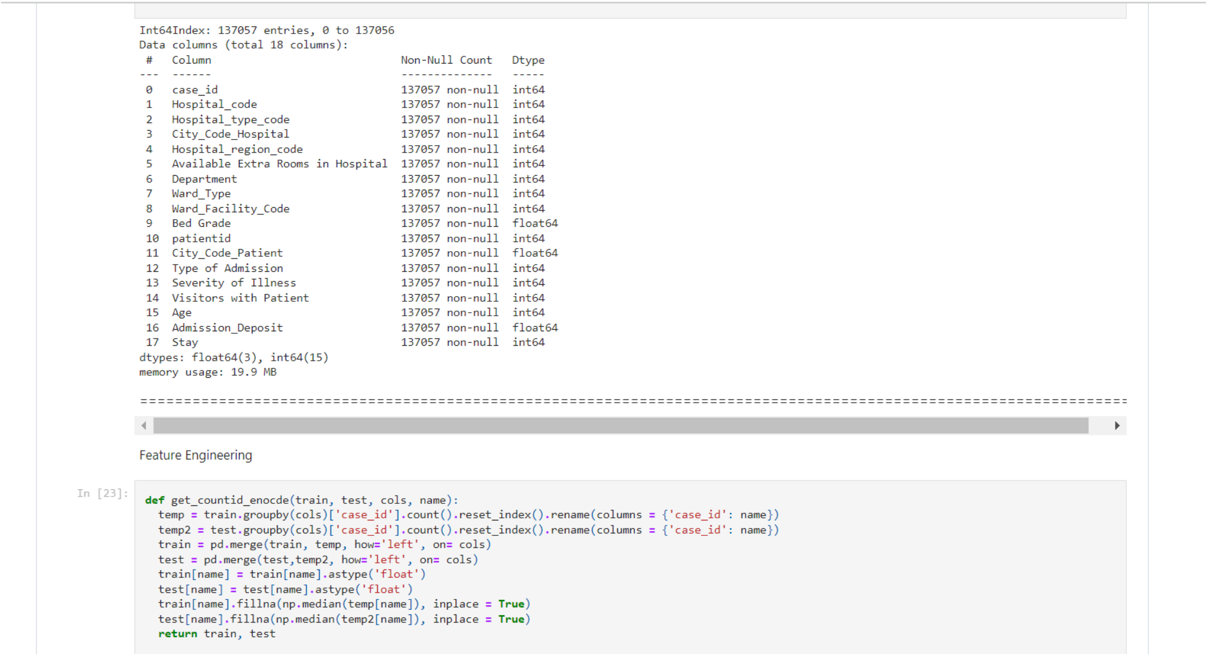
	case_id	Hospital_code	Hospital_type_code	City_Code_Hospital	Hospital_region_code	Available Extra Rooms in Hospital	Department	Ward_Type	Ward_Facility_Code	Bed Grade	patientid	City_Code_Patien
0	318439	21	2	3	2	3	2	3	0	2.0	17006	2.
1	318440	29	0	4	0	2	2	3	5	2.0	17006	2.
2	318441	26	1	2	1	3	2	1	3	4.0	17006	2.
3	318442	6	0	6	0	3	2	1	5	2.0	17006	2.
4	318443	28	1	11	0	2	2	2	5	2.0	17006	2.

```
In [19]: train.shape

Out[19]: (318438, 18)

In [20]: test.shape

Out[20]: (137057, 18)

In [21]: train.info()

Int64Index: 318438 entries, 0 to 318437
Data columns (total 18 columns):
 #   Column                             Non-Null Count   Dtype
---  ------                             --------------   -----
 0   case_id                            318438 non-null  int64
 1   Hospital_code                      318438 non-null  int64
 2   Hospital_type_code                 318438 non-null  int64
 3   City_Code_Hospital                 318438 non-null  int64
```



```
In [21]: train.info()

Int64Index: 318438 entries, 0 to 318437
Data columns (total 18 columns):
 #   Column                             Non-Null Count   Dtype
---  ------                             --------------   -----
 0   case_id                            318438 non-null  int64
 1   Hospital_code                      318438 non-null  int64
 2   Hospital_type_code                 318438 non-null  int64
 3   City_Code_Hospital                 318438 non-null  int64
 4   Hospital_region_code               318438 non-null  int64
 5   Available Extra Rooms in Hospital  318438 non-null  int64
 6   Department                         318438 non-null  int64
 7   Ward_Type                          318438 non-null  int64
 8   Ward_Facility_Code                 318438 non-null  int64
 9   Bed Grade                          318438 non-null  float64
 10  patientid                          318438 non-null  int64
 11  City_Code_Patient                  318438 non-null  float64
 12  Type of Admission                  318438 non-null  int64
 13  Severity of Illness                318438 non-null  int64
 14  Visitors with Patient              318438 non-null  int64
 15  Age                                318438 non-null  int64
 16  Admission_Deposit                  318438 non-null  float64
 17  Stay                               318438 non-null  int64
dtypes: float64(3), int64(15)
memory usage: 46.2 MB

In [22]: test.info()

Int64Index: 137057 entries, 0 to 137056
Data columns (total 18 columns):
 #   Column                             Non-Null Count   Dtype
---  ------                             --------------   -----
 0   case_id                            137057 non-null  int64
 1   Hospital_code                      137057 non-null  int64
 2   Hospital_type_code                 137057 non-null  int64
 3   City_Code_Hospital                 137057 non-null  int64
 4   Hospital_region_code               137057 non-null  int64
 5   Available Extra Rooms in Hospital  137057 non-null  int64
 6   Department                         137057 non-null  int64
 7   Ward_Type                          137057 non-null  int64
 8   Ward_Facility_Code                 137057 non-null  int64
```

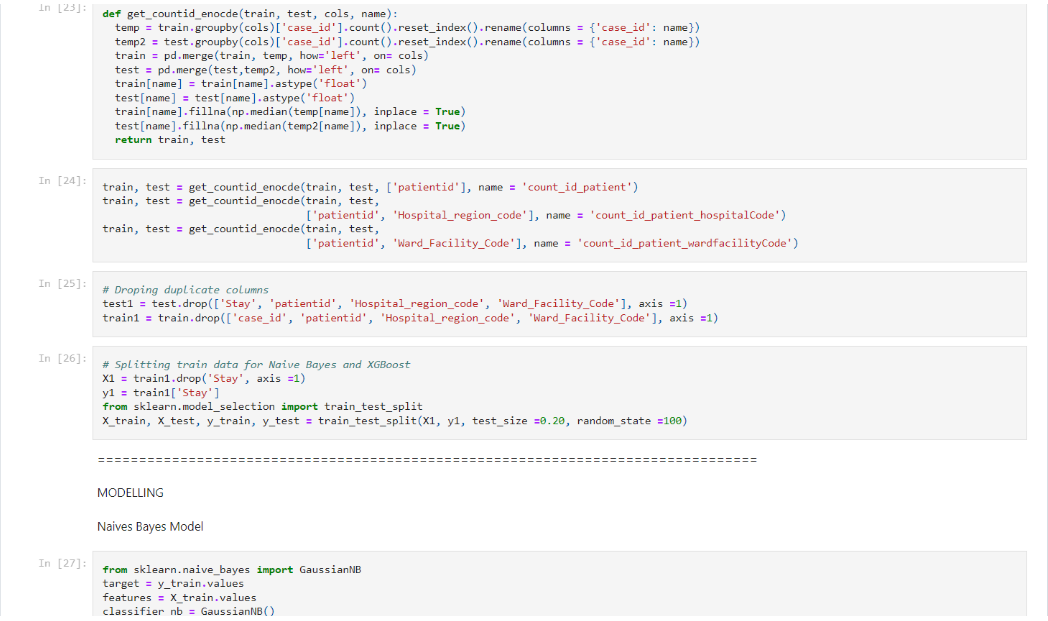
FEATURE ENGINEERING:



```
Int64Index: 137057 entries, 0 to 137056
Data columns (total 18 columns):
 #   Column                             Non-Null Count   Dtype  
---  ------                             --------------   -----  
 0   case_id                            137057 non-null  int64  
 1   Hospital_code                      137057 non-null  int64  
 2   Hospital_type_code                 137057 non-null  int64  
 3   City_Code_Hospital                 137057 non-null  int64  
 4   Hospital_region_code               137057 non-null  int64  
 5   Available Extra Rooms in Hospital  137057 non-null  int64  
 6   Department                         137057 non-null  int64  
 7   Ward_Type                          137057 non-null  int64  
 8   Ward_Facility_Code                 137057 non-null  int64  
 9   Bed Grade                          137057 non-null  float64
 10  patientid                          137057 non-null  int64  
 11  City_Code_Patient                  137057 non-null  float64
 12  Type of Admission                  137057 non-null  int64  
 13  Severity of Illness                137057 non-null  int64  
 14  Visitors with Patient              137057 non-null  int64  
 15  Age                                137057 non-null  int64  
 16  Admission_Deposit                  137057 non-null  float64
 17  Stay                               137057 non-null  int64  
dtypes: float64(3), int64(15)
memory usage: 19.9 MB

==========================================================================================================================================================
```

Feature Engineering

```
In [23]:
def get_countid_enocde(train, test, cols, name):
    temp = train.groupby(cols)['case_id'].count().reset_index().rename(columns = {'case_id': name})
    temp2 = test.groupby(cols)['case_id'].count().reset_index().rename(columns = {'case_id': name})
    train = pd.merge(train, temp, how='left', on= cols)
    test = pd.merge(test,temp2, how='left', on= cols)
    train[name] = train[name].astype('float')
    test[name] = test[name].astype('float')
    train[name].fillna(np.median(temp[name]), inplace = True)
    test[name].fillna(np.median(temp2[name]), inplace = True)
    return train, test
```



```
In [23]:
def get_countid_enocde(train, test, cols, name):
    temp = train.groupby(cols)['case_id'].count().reset_index().rename(columns = {'case_id': name})
    temp2 = test.groupby(cols)['case_id'].count().reset_index().rename(columns = {'case_id': name})
    train = pd.merge(train, temp, how='left', on= cols)
    test = pd.merge(test,temp2, how='left', on= cols)
    train[name] = train[name].astype('float')
    test[name] = test[name].astype('float')
    train[name].fillna(np.median(temp[name]), inplace = True)
    test[name].fillna(np.median(temp2[name]), inplace = True)
    return train, test
```

```
In [24]:
train, test = get_countid_enocde(train, test, ['patientid'], name = 'count_id_patient')
train, test = get_countid_enocde(train, test,
                                 ['patientid', 'Hospital_region_code'], name = 'count_id_patient_hospitalCode')
train, test = get_countid_enocde(train, test,
                                 ['patientid', 'Ward_Facility_Code'], name = 'count_id_patient_wardfacilityCode')
```

```
In [25]:
# Droping duplicate columns
test1 = test.drop(['Stay', 'patientid', 'Hospital_region_code', 'Ward_Facility_Code'], axis =1)
train1 = train.drop(['case_id', 'patientid', 'Hospital_region_code', 'Ward_Facility_Code'], axis =1)
```

```
In [26]:
# Splitting train data for Naive Bayes and XGBoost
X1 = train1.drop('Stay', axis =1)
y1 = train1['Stay']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X1, y1, test_size =0.20, random_state =100)
```

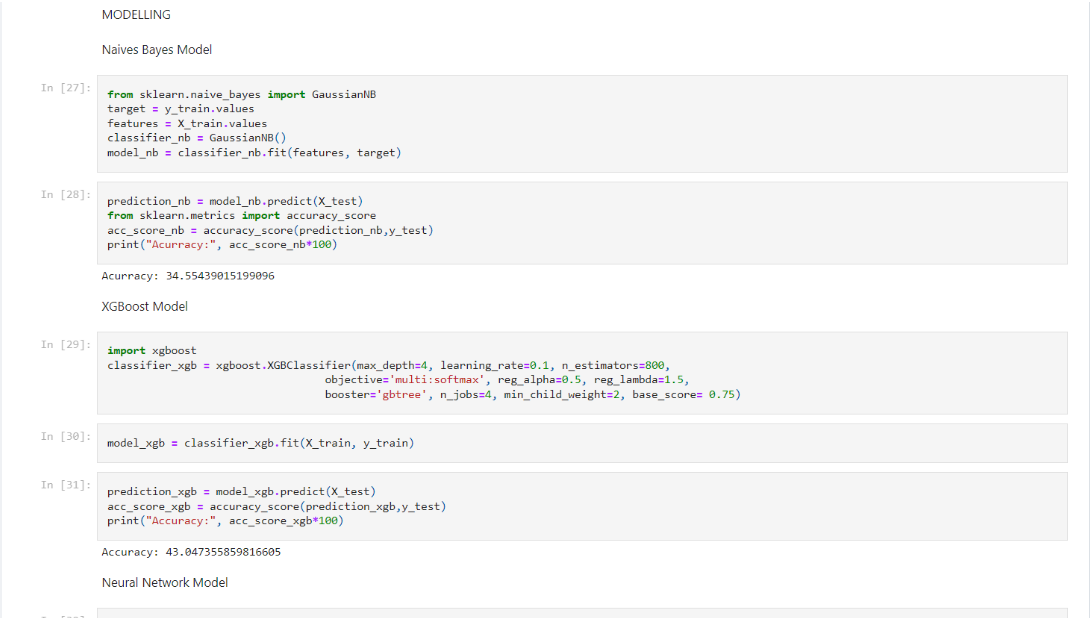
```
===============================================================================
```

MODELLING

Naives Bayes Model

```
In [27]:
from sklearn.naive_bayes import GaussianNB
target = y_train.values
features = X_train.values
classifier_nb = GaussianNB()
```

MODELLING:



MODELLING

Naives Bayes Model

```
In [27]:
from sklearn.naive_bayes import GaussianNB
target = y_train.values
features = X_train.values
classifier_nb = GaussianNB()
model_nb = classifier_nb.fit(features, target)
```

```
In [28]:
prediction_nb = model_nb.predict(X_test)
from sklearn.metrics import accuracy_score
acc_score_nb = accuracy_score(prediction_nb,y_test)
print("Acurracy:", acc_score_nb*100)
```

```
Acurracy: 34.55439015199096
```

XGBoost Model

```
In [29]:
import xgboost
classifier_xgb = xgboost.XGBClassifier(max_depth=4, learning_rate=0.1, n_estimators=800,
                                  objective='multi:softmax', reg_alpha=0.5, reg_lambda=1.5,
                                  booster='gbtree', n_jobs=4, min_child_weight=2, base_score= 0.75)
```

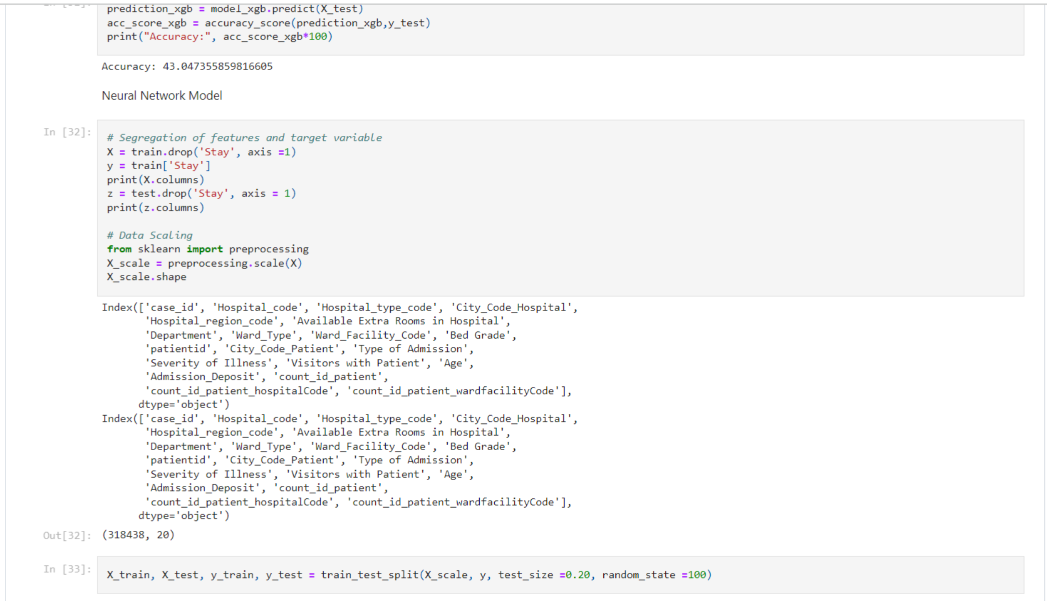
```
In [30]:
model_xgb = classifier_xgb.fit(X_train, y_train)
```

```
In [31]:
prediction_xgb = model_xgb.predict(X_test)
acc_score_xgb = accuracy_score(prediction_xgb,y_test)
print("Accuracy:", acc_score_xgb*100)
```

```
Accuracy: 43.047355859816605
```

Neural Network Model

NEURAL NETWORK MODEL



```
prediction_xgb = model_xgb.predict(X_test)
acc_score_xgb = accuracy_score(prediction_xgb,y_test)
print("Accuracy:", acc_score_xgb*100)
```

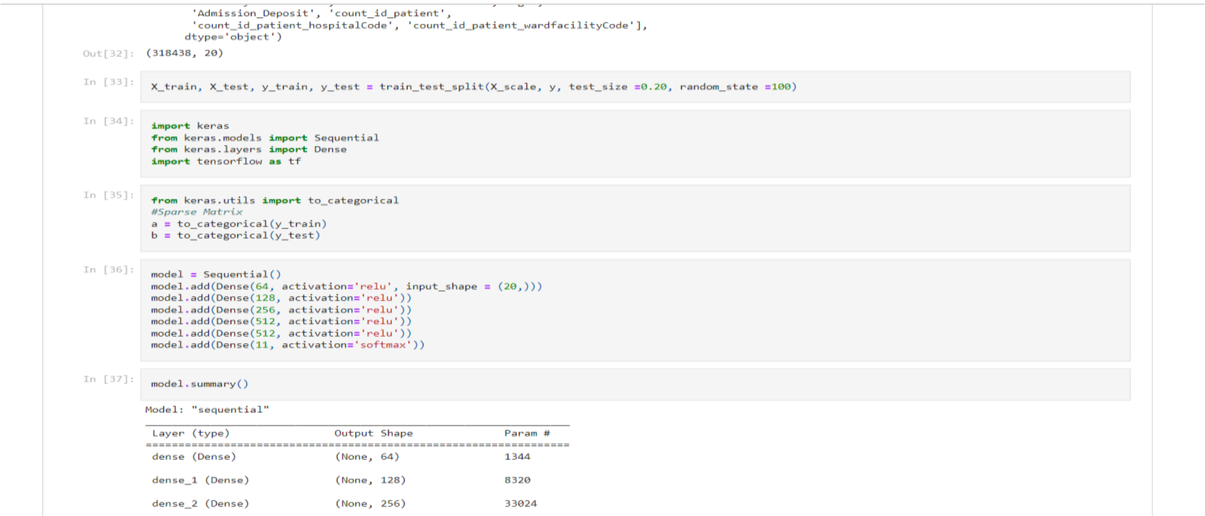
```
Accuracy: 43.047355859816605
```

Neural Network Model

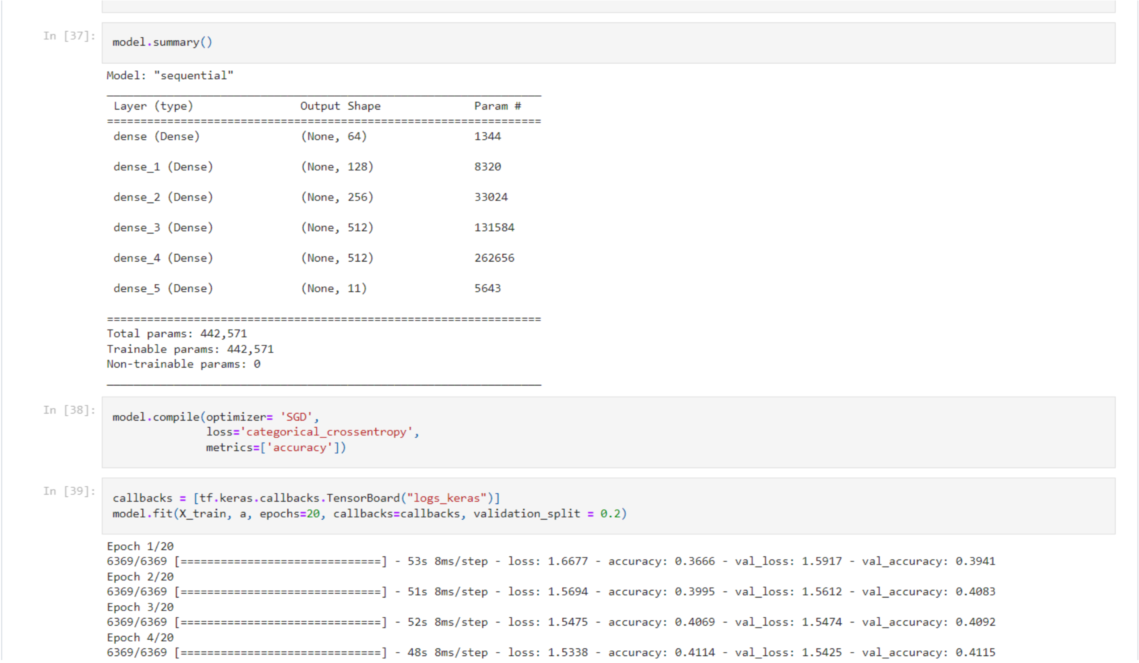
```
In [32]:
# Segregation of features and target variable
X = train.drop('Stay', axis =1)
y = train['Stay']
print(X.columns)
z = test.drop('Stay', axis = 1)
print(z.columns)

# Data Scaling
from sklearn import preprocessing
X_scale = preprocessing.scale(X)
X_scale.shape
```

```
Index(['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',
       'patientid', 'City_Code_Patient', 'Type of Admission',
       'Severity of Illness', 'Visitors with Patient', 'Age',
       'Admission_Deposit', 'count_id_patient',
       'count_id_patient_hospitalCode', 'count_id_patient_wardfacilityCode'],
      dtype='object')
Index(['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',
       'patientid', 'City_Code_Patient', 'Type of Admission',
       'Severity of Illness', 'Visitors with Patient', 'Age',
       'Admission_Deposit', 'count_id_patient',
       'count_id_patient_hospitalCode', 'count_id_patient_wardfacilityCode'],
      dtype='object')
Out[32]: (318438, 20)
```

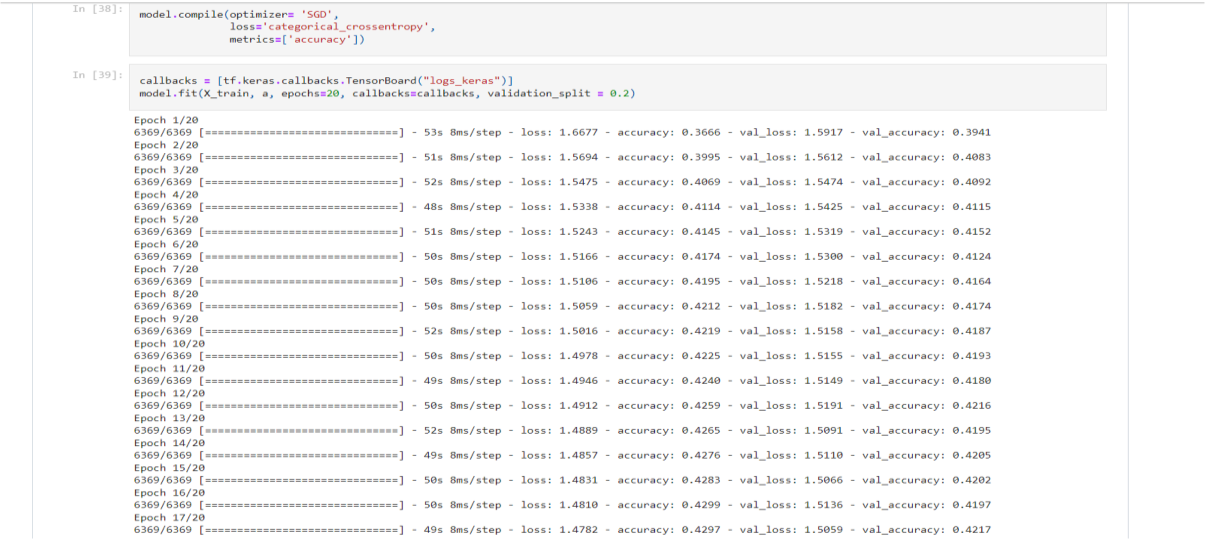
```
In [33]:
X_train, X_test, y_train, y_test = train_test_split(X_scale, y, test_size =0.20, random_state =100)
```



```
            'Admission_Deposit', 'count_id_patient',
            'count_id_patient_hospitalCode', 'count_id_patient_wardfacilityCode'],
           dtype='object')
Out[32]: (318438, 20)

In [33]: X_train, X_test, y_train, y_test = train_test_split(X_scale, y, test_size =0.20, random_state =100)

In [34]: import keras
         from keras.models import Sequential
         from keras.layers import Dense
         import tensorflow as tf

In [35]: from keras.utils import to_categorical
         #Sparse Matrix
         a = to_categorical(y_train)
         b = to_categorical(y_test)

In [36]: model = Sequential()
         model.add(Dense(64, activation='relu', input_shape = (20,)))
         model.add(Dense(128, activation='relu'))
         model.add(Dense(256, activation='relu'))
         model.add(Dense(512, activation='relu'))
         model.add(Dense(512, activation='relu'))
         model.add(Dense(11, activation='softmax'))

In [37]: model.summary()

         Model: "sequential"
         _________________________________________________________________
          Layer (type)                Output Shape              Param #
         =================================================================
          dense (Dense)               (None, 64)                1344

          dense_1 (Dense)             (None, 128)               8320

          dense_2 (Dense)             (None, 256)               33024
```



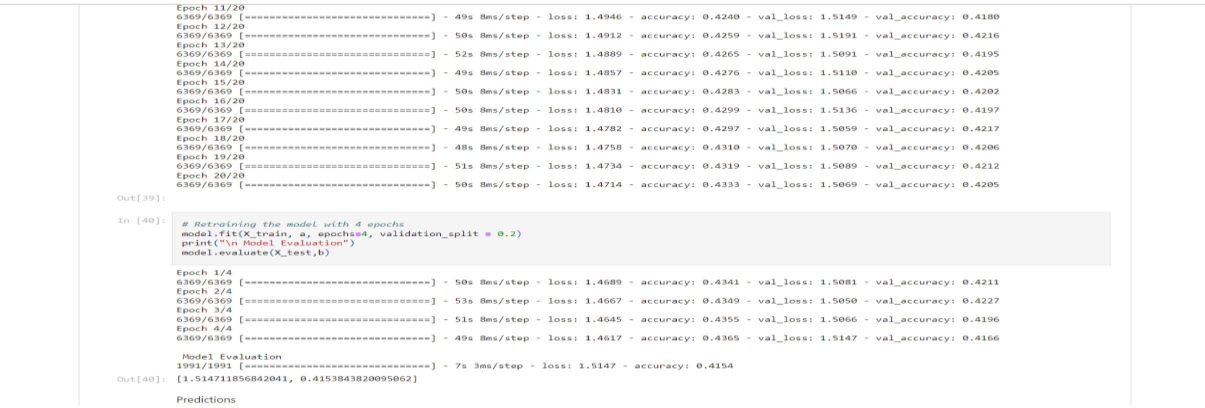
```
In [37]: model.summary()

         Model: "sequential"
         _________________________________________________________________
          Layer (type)                Output Shape              Param #
         =================================================================
          dense (Dense)               (None, 64)                1344

          dense_1 (Dense)             (None, 128)               8320

          dense_2 (Dense)             (None, 256)               33024

          dense_3 (Dense)             (None, 512)               131584

          dense_4 (Dense)             (None, 512)               262656

          dense_5 (Dense)             (None, 11)                5643

         =================================================================
         Total params: 442,571
         Trainable params: 442,571
         Non-trainable params: 0
         _________________________________________________________________

In [38]: model.compile(optimizer= 'SGD',
                       loss='categorical_crossentropy',
                       metrics=['accuracy'])

In [39]: callbacks = [tf.keras.callbacks.TensorBoard("logs_keras")]
         model.fit(X_train, a, epochs=20, callbacks=callbacks, validation_split = 0.2)

         Epoch 1/20
         6369/6369 [==============================] - 53s 8ms/step - loss: 1.6677 - accuracy: 0.3666 - val_loss: 1.5917 - val_accuracy: 0.3941
         Epoch 2/20
         6369/6369 [==============================] - 51s 8ms/step - loss: 1.5694 - accuracy: 0.3995 - val_loss: 1.5612 - val_accuracy: 0.4083
         Epoch 3/20
         6369/6369 [==============================] - 52s 8ms/step - loss: 1.5475 - accuracy: 0.4069 - val_loss: 1.5474 - val_accuracy: 0.4092
         Epoch 4/20
         6369/6369 [==============================] - 48s 8ms/step - loss: 1.5338 - accuracy: 0.4114 - val_loss: 1.5425 - val_accuracy: 0.4115
```



In [38]:
```
model.compile(optimizer= 'SGD',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

In [39]:
```
callbacks = [tf.keras.callbacks.TensorBoard("logs_keras")]
model.fit(X_train, a, epochs=20, callbacks=callbacks, validation_split = 0.2)
```

```
Epoch 1/20
6369/6369 [==============================] - 53s 8ms/step - loss: 1.6677 - accuracy: 0.3666 - val_loss: 1.5917 - val_accuracy: 0.3941
Epoch 2/20
6369/6369 [==============================] - 51s 8ms/step - loss: 1.5694 - accuracy: 0.3995 - val_loss: 1.5612 - val_accuracy: 0.4083
Epoch 3/20
6369/6369 [==============================] - 52s 8ms/step - loss: 1.5475 - accuracy: 0.4069 - val_loss: 1.5474 - val_accuracy: 0.4092
Epoch 4/20
6369/6369 [==============================] - 48s 8ms/step - loss: 1.5338 - accuracy: 0.4114 - val_loss: 1.5425 - val_accuracy: 0.4115
Epoch 5/20
6369/6369 [==============================] - 51s 8ms/step - loss: 1.5243 - accuracy: 0.4145 - val_loss: 1.5319 - val_accuracy: 0.4152
Epoch 6/20
6369/6369 [==============================] - 50s 8ms/step - loss: 1.5166 - accuracy: 0.4174 - val_loss: 1.5300 - val_accuracy: 0.4124
Epoch 7/20
6369/6369 [==============================] - 50s 8ms/step - loss: 1.5106 - accuracy: 0.4195 - val_loss: 1.5218 - val_accuracy: 0.4164
Epoch 8/20
6369/6369 [==============================] - 50s 8ms/step - loss: 1.5059 - accuracy: 0.4212 - val_loss: 1.5182 - val_accuracy: 0.4174
Epoch 9/20
6369/6369 [==============================] - 52s 8ms/step - loss: 1.5016 - accuracy: 0.4219 - val_loss: 1.5158 - val_accuracy: 0.4187
Epoch 10/20
6369/6369 [==============================] - 50s 8ms/step - loss: 1.4978 - accuracy: 0.4225 - val_loss: 1.5155 - val_accuracy: 0.4193
Epoch 11/20
6369/6369 [==============================] - 49s 8ms/step - loss: 1.4946 - accuracy: 0.4240 - val_loss: 1.5149 - val_accuracy: 0.4180
Epoch 12/20
6369/6369 [==============================] - 50s 8ms/step - loss: 1.4912 - accuracy: 0.4259 - val_loss: 1.5191 - val_accuracy: 0.4216
Epoch 13/20
6369/6369 [==============================] - 52s 8ms/step - loss: 1.4889 - accuracy: 0.4265 - val_loss: 1.5091 - val_accuracy: 0.4195
Epoch 14/20
6369/6369 [==============================] - 49s 8ms/step - loss: 1.4857 - accuracy: 0.4276 - val_loss: 1.5110 - val_accuracy: 0.4205
Epoch 15/20
6369/6369 [==============================] - 50s 8ms/step - loss: 1.4831 - accuracy: 0.4283 - val_loss: 1.5066 - val_accuracy: 0.4202
Epoch 16/20
6369/6369 [==============================] - 50s 8ms/step - loss: 1.4810 - accuracy: 0.4299 - val_loss: 1.5136 - val_accuracy: 0.4197
Epoch 17/20
6369/6369 [==============================] - 49s 8ms/step - loss: 1.4782 - accuracy: 0.4297 - val_loss: 1.5059 - val_accuracy: 0.4217
```



```
Epoch 11/20
6369/6369 [==============================] - 49s 8ms/step - loss: 1.4946 - accuracy: 0.4240 - val_loss: 1.5149 - val_accuracy: 0.4180
Epoch 12/20
6369/6369 [==============================] - 50s 8ms/step - loss: 1.4912 - accuracy: 0.4259 - val_loss: 1.5191 - val_accuracy: 0.4216
Epoch 13/20
6369/6369 [==============================] - 52s 8ms/step - loss: 1.4889 - accuracy: 0.4265 - val_loss: 1.5091 - val_accuracy: 0.4195
Epoch 14/20
6369/6369 [==============================] - 49s 8ms/step - loss: 1.4857 - accuracy: 0.4276 - val_loss: 1.5110 - val_accuracy: 0.4205
Epoch 15/20
6369/6369 [==============================] - 50s 8ms/step - loss: 1.4831 - accuracy: 0.4283 - val_loss: 1.5066 - val_accuracy: 0.4202
Epoch 16/20
6369/6369 [==============================] - 50s 8ms/step - loss: 1.4810 - accuracy: 0.4299 - val_loss: 1.5136 - val_accuracy: 0.4197
Epoch 17/20
6369/6369 [==============================] - 49s 8ms/step - loss: 1.4782 - accuracy: 0.4297 - val_loss: 1.5059 - val_accuracy: 0.4217
Epoch 18/20
6369/6369 [==============================] - 48s 8ms/step - loss: 1.4758 - accuracy: 0.4310 - val_loss: 1.5070 - val_accuracy: 0.4206
Epoch 19/20
6369/6369 [==============================] - 51s 8ms/step - loss: 1.4734 - accuracy: 0.4319 - val_loss: 1.5089 - val_accuracy: 0.4212
Epoch 20/20
6369/6369 [==============================] - 50s 8ms/step - loss: 1.4714 - accuracy: 0.4333 - val_loss: 1.5069 - val_accuracy: 0.4205
```

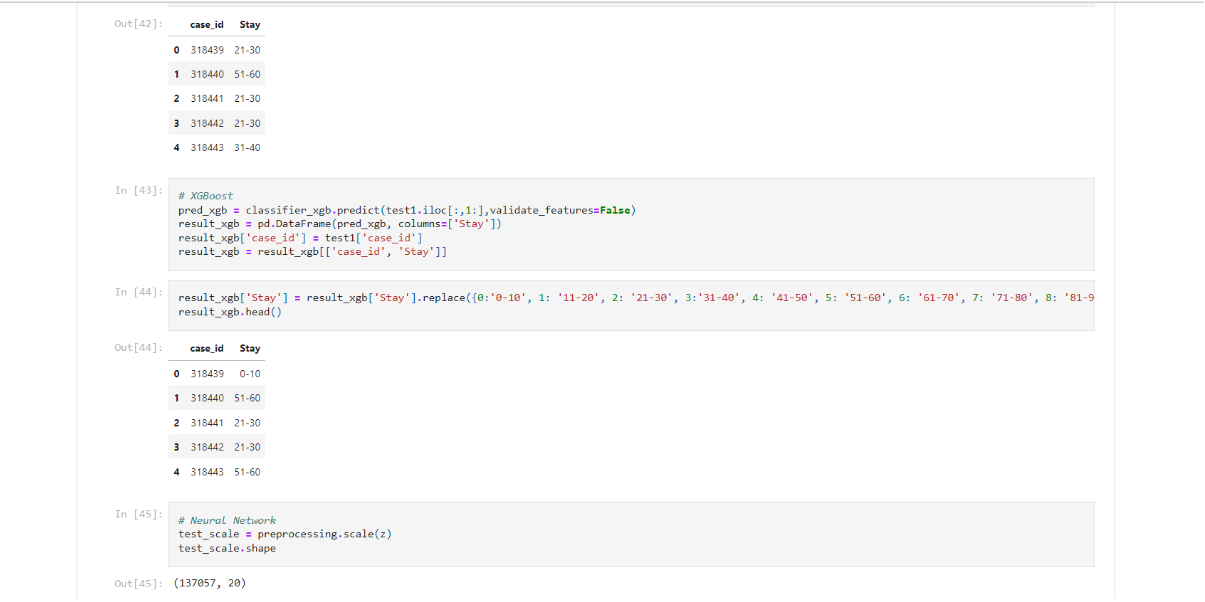
Out[39]:

In [40]:
```
# Retraining the model with 4 epochs
model.fit(X_train, a, epochs=4, validation_split = 0.2)
print("\n Model Evaluation")
model.evaluate(X_test,b)
```

```
Epoch 1/4
6369/6369 [==============================] - 50s 8ms/step - loss: 1.4689 - accuracy: 0.4341 - val_loss: 1.5081 - val_accuracy: 0.4211
Epoch 2/4
6369/6369 [==============================] - 53s 8ms/step - loss: 1.4667 - accuracy: 0.4349 - val_loss: 1.5050 - val_accuracy: 0.4227
Epoch 3/4
6369/6369 [==============================] - 51s 8ms/step - loss: 1.4645 - accuracy: 0.4355 - val_loss: 1.5066 - val_accuracy: 0.4196
Epoch 4/4
6369/6369 [==============================] - 49s 8ms/step - loss: 1.4617 - accuracy: 0.4365 - val_loss: 1.5147 - val_accuracy: 0.4166

 Model Evaluation
1991/1991 [==============================] - 7s 3ms/step - loss: 1.5147 - accuracy: 0.4154
```

Out[40]:
```
[1.514711856842041, 0.4153843820095062]
```

Predictions



Out[42]:

	case_id	Stay
0	318439	21-30
1	318440	51-60
2	318441	21-30
3	318442	21-30
4	318443	31-40

In [43]:
```
# XGBoost
pred_xgb = classifier_xgb.predict(test1.iloc[:,1:],validate_features=False)
result_xgb = pd.DataFrame(pred_xgb, columns=['Stay'])
result_xgb['case_id'] = test1['case_id']
result_xgb = result_xgb[['case_id', 'Stay']]
```

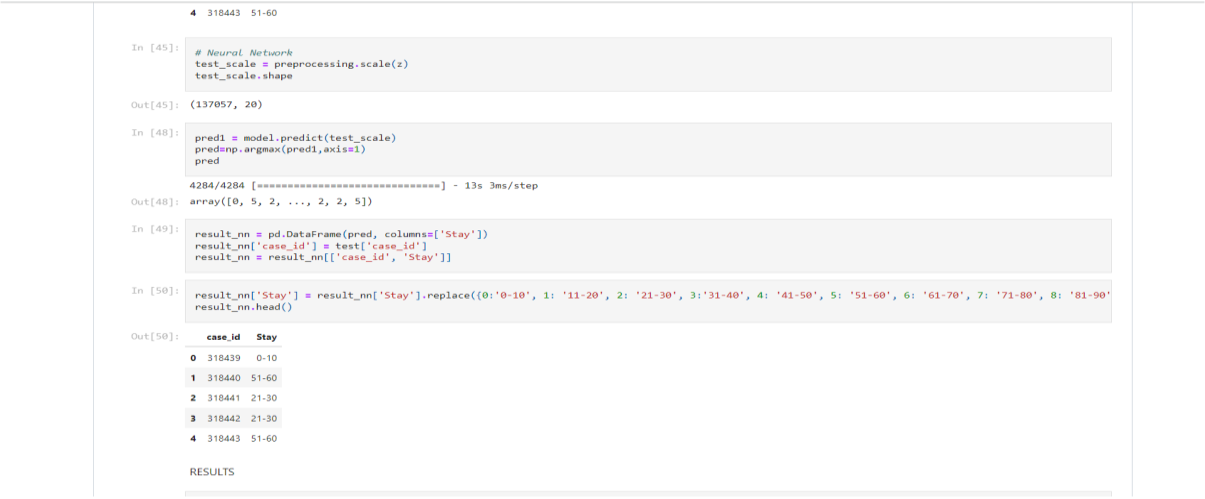
In [44]:
```
result_xgb['Stay'] = result_xgb['Stay'].replace({0:'0-10', 1: '11-20', 2: '21-30', 3:'31-40', 4: '41-50', 5: '51-60', 6: '61-70', 7: '71-80', 8: '81-9
result_xgb.head()
```

Out[44]:

	case_id	Stay
0	318439	0-10
1	318440	51-60
2	318441	21-30
3	318442	21-30
4	318443	51-60

In [45]:
```
# Neural Network
test_scale = preprocessing.scale(z)
test_scale.shape
```

Out[45]:
```
(137057, 20)
```



4	318443	51-60

In [45]:
```
# Neural Network
test_scale = preprocessing.scale(z)
test_scale.shape
```

Out[45]:
```
(137057, 20)
```

In [48]:
```
pred1 = model.predict(test_scale)
pred=np.argmax(pred1,axis=1)
pred
```

```
4284/4284 [==============================] - 13s 3ms/step
```

Out[48]:
```
array([0, 5, 2, ..., 2, 2, 5])
```

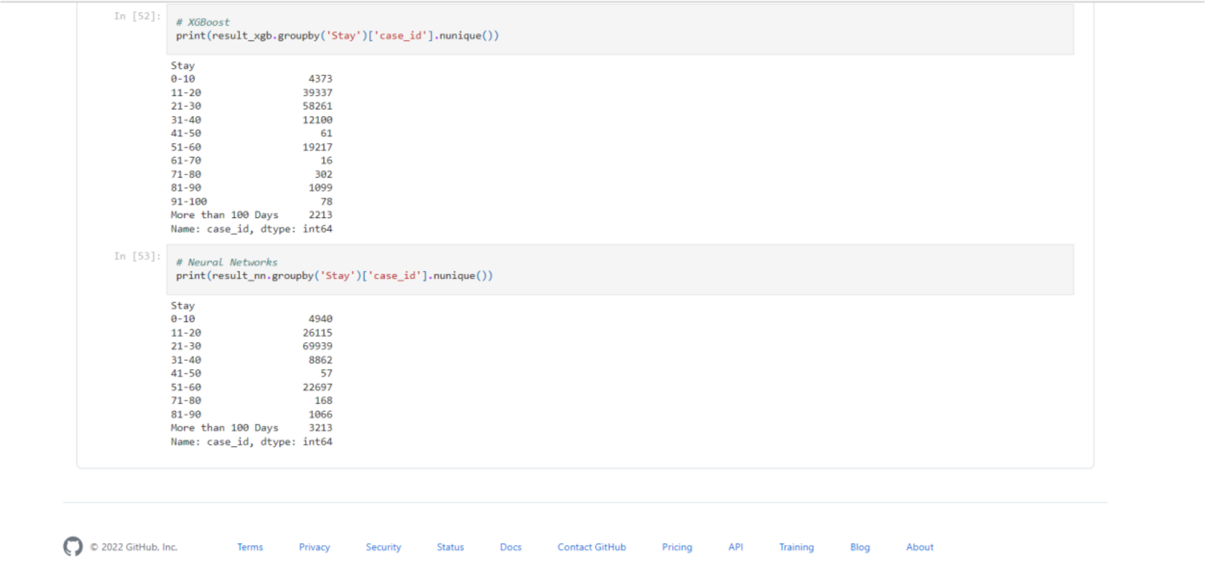
In [49]:
```
result_nn = pd.DataFrame(pred, columns=['Stay'])
result_nn['case_id'] = test['case_id']
result_nn = result_nn[['case_id', 'Stay']]
```

In [50]:
```
result_nn['Stay'] = result_nn['Stay'].replace({0:'0-10', 1: '11-20', 2: '21-30', 3:'31-40', 4: '41-50', 5: '51-60', 6: '61-70', 7: '71-80', 8: '81-90'
result_nn.head()
```

Out[50]:

	case_id	Stay
0	318439	0-10
1	318440	51-60
2	318441	21-30
3	318442	21-30
4	318443	51-60

RESULTS



In [52]:
```
# XGBoost
print(result_xgb.groupby('Stay')['case_id'].nunique())
```

```
Stay
0-10                  4373
11-20                39337
21-30                58261
31-40                12100
41-50                   61
51-60                19217
61-70                   16
71-80                  302
81-90                 1099
91-100                  78
More than 100 Days    2213
Name: case_id, dtype: int64
```

In [53]:
```
# Neural Networks
print(result_nn.groupby('Stay')['case_id'].nunique())
```

```
Stay
0-10                  4940
11-20                26115
21-30                69939
31-40                 8862
41-50                   57
51-60                22697
71-80                  168
81-90                 1066
More than 100 Days    3213
Name: case_id, dtype: int64
```

© 2022 GitHub, Inc.   Terms   Privacy   Security   Status   Docs   Contact GitHub   Pricing   API   Training   Blog   About