Appendix:

Code: Feature engineering:

```
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 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')

# 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)

# 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)


Naïve bayes Model

sklearn.naive_bayes import GaussianNB

target = y_train.values

features = X_train.values

classifier_nb = GaussianNB()

model_nb = classifier_nb.fit(features, target)

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)

XGBoost model

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) model_xgb = classifier_xgb.fit(X_train, y_train) prediction_

xgb = model_xgb.predict(X_test) acc_score_xgb = accuracy_score(prediction_xgb,y_test)

print("Accuracy:", acc_score_xgb*100)

Neural Network

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 X_train, X_test, y_train, y_test = train_test_split(X_scale, y, test_size =0.20, random_state =100)