CAPSTONE PROJECT

On

HEALTHCARE ANALYTICS

By,

Akash Bhor

Contents **Page No.**

1. Introduction

Healthcare organizations are under increasing pressure to improve patient care outcomes and achieve better care. While this situation represents a challenge, it also offers organizations an opportunity to dramatically improve the quality of care by leveraging more value and insights from their data. Health care analytics refers to the analysis of data using quantitative and qualitative techniques to explore trends and patterns in the acquired data. While healthcare management uses various metrics for performance, a patient's length of stay is an important one.

Being able to predict the length of stay (LOS) allows hospitals to optimize their treatment plans to reduce LOS, to reduce infection rates among patients, staff, and visitors.

2. Project Goal

The goal of this project is to accurately predict the Length of Stay for each patient so that the hospitals can optimize resources and function better.

3. Hypothesis Generation

Understanding the problem in detail by assuming different factors that impact the outcomes of Length of Stay before any data exploration or analysis. Here the variables can be divided into two levels: Patient-Level and Hospital-Level.

Patient-Level:

- Type of Admission – Patients can be admitted in three levels Urgent, Emergency, and Trauma. Patients admitted to urgent care are likely to stay fewer days. Whereas Trauma patients usually stay longer because they must be monitored until they are qualified to be discharged.
- Severity of Illness – Severity can be classified as Minor, Moderate, and Extreme. A patient recorded as minor will stay fewer days than a patient recorded as extreme.
- Visitors with Patient – Patients with more visitors are like to stay longer in the hospital.
- Age – Infants and older Patients usually take a longer time to recover so they stay longer than younger Patients.
- Admission Deposit – Patients who are likely to deposit a high amount of money at the time of admission might have severe conditions and stay longer.

Hospital-Level:

- Ward Type – Patients allocated in ICU might stay longer than the general ward as their condition is more severe.
- Department – Patients under surgery are likely to stay longer than gynecology as their recovery time is longer.

4. Data Exploration

4.1 Overview of Data

The train data consist of 318438 observations for which patient length of stay can be predicted from 17 variables. The description of all variables is shown in Table 4.1.

Table 4.1 Dataset Overview

Variables	Description
case_id	Case_ID registered in Hospital
Hospital_code	Unique code for the Hospital
Hospital_type_code	Unique code for the type of Hospital
City_Code_Hospital	City Code of the Hospital
Hospital_region_code	Region Code of the Hospital
Available Extra Rooms in Hospital	Number of Extra rooms available in the Hospital
Department	Department overlooking the case
Ward_Type	Code for the Ward type
Ward_Facility_Code	Code for the Ward Facility
Bed Grade	Condition of Bed in the Ward
patientid	Unique Patient Id
City_Code_Patient	City Code for the patient
Type of Admission	Admission Type registered by the Hospital
Severity of Illness	Severity of the illness recorded at the time of admission
Visitors with Patient	Number of Visitors with the patient
Age	Age of the patient
Admission_Deposit	Deposit at the time of Admission
Stay	Patient Length of Stay

In this data, the target variable "stay" is divided into 11 different classes ranging from 0 days to more than 100 days. **Figure 4.1** shows different levels of the "Stay" variable.

```
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)
```

Figure 4.1 Unique values in Stay Column

4.2 Data Cleaning and Preparation

In this data set, variables "City_code_patient" and "Bed Grade" have missing values. These missing values must be treated before feeding to the algorithm as they distort the model performance. So, the missing values are replaced using the "mode" imputation technique.

Since most of the variables in the dataset have ordinal data, we transformed them into levels by using a label encoder to perform further analysis on the data. Table 4.2 shows the number of distinct observations of ordinal data in the dataset.

Table 4.2 Distinct Observations of Ordinal Data

Variables	**Number of distinct observations**
Hospital_type_code	7
Hospital_region_code	3
Department	5
Ward_Type	6
Ward_Facility_Code	6
Type of Admission	3
Severity of Illness	3
Age	10
Stay	11

4.3 Feature Engineering

Once the data is cleaned and prepared, we grouped patientid and case_id to extract the new column "count_id_patient". This variable contains the count of multiple admits of a patient under different case_id. Further two more columns "Hospital_region_code" and "ward_facility_code" were grouped to patientid and case_id. These two new variables "count_id_patient_hospitalCode" and "count_id_patient_wardfacilityCode" contain the count of multiple admissions in a hospital region and the count of multiple wards allocated to a patient. [Appendix - 10.1 Feature Engineering]

Before getting into analysis, the train data must be split into two parts, the first part with all the feature variables and the second part with a target variable ("Stay"). Then preprocessed into train and validation sets. So, here we are portioning the train set with 80% and validation set with 20% of the data for Naïve Bayes and XGBoost models.

5. Modelling Strategy

5.1 Model 1 - Naïve Bayes

Naïve Bayes is a classification technique that works on the principle of Bayes theorem with an assumption of independence among the variables. Here the goal is to predict Length of Stay i.e., "Stay" column (Target Variable) and it is classified into 11 levels. We must find the probability of each patient's length of stay using feature variables, which contain the patient's condition and hospital-level information. These feature variables are ordinal and naïve Bayes is a perfect multilevel classifier.

In Bayes theorem, given a Hypothesis H and Evidence E, it states that the relation between the probability of Hypothesis P(H) before getting Evidence and probability of hypothesis after getting Evidence P(H|E)

$$\mathbf{P(H \mid E) = [\frac{\square(\square \mid \square)}{\square(\square)}\ ]\ P(H)}$$

When we apply Bayes Theorem to our data it represents as follows.

- P(H) is the prior probability of a patient's length of stay (LOS).
- P(E) is the probability of a feature variable.
- P(E|H) is the probability of a patient's LOS given that the features are true.
- P(H|E) is the probability of the features given that patient's LOS is true.

Model is trained using Gaussian Naïve Bayes classifier, partitioned train data is fed to the model in array format then the trained model is validated using validation data. This model gives an accuracy score of **34.55%** after validating. [Appendix – 10.2.1 Naïve Bayes]

5.2 Model 2 – XGBoost

Boosting is a sequential technique that works on the principle of an ensemble. At any instant T, the model outcomes are weighed based on the outcomes of the previous instant (T -1). It combines the set of weak learners and improves prediction accuracy. Tree ensemble is a set of classification and regression trees. Trees are grown one after another, and they try to reduce the misclassification rate. The final prediction score of the model is calculated by summing up each and individual score.

Before feeding train data to the XGB Classifier model, booster parameters must be tuned. Tunning the model can prevent overfitting and can yield higher accuracy.

In this XGBoost model, we have used the following parameters for tunning,

- **learning_rate = 0.1** - step size shrinkage used to prevent overfitting. After each boosting step, we can directly get the weights of new features, and eta shrinks the feature weights to make the boosting process more conservative.

- **max_depth = 4** – Maximum depth of the tree. This value describes the complexity of the model. Increasing its value results in overfitting.
- **n_estimators = 800 –** Number of gradient boosting trees or rounds. Each new tree attempts to model and correct for the errors made by the sequence of previous trees. Increasing the number of trees can yield higher accuracy but the model reaches a point of diminishing returns quickly.
- **objective = 'multi:softmax' –** this parameter sets XGBoost to do multiclass classification using the softmax objective because the target variable has 11 Levels.
- **reg_alpha = 0.5 -** L1 regularization term on weights. Increasing this value will make the model more conservative.
- **reg_lambda = 1.5 -** L2 regularization term on weights and is smoother than L1 regularization. Increasing this value will model more conservative.
- **min_child_weight = 2 -** Minimum sum of instance weight needed in a child.

Once the model was trained and validated, it yields an accuracy score of **43.04%**. When compared to the Naïve Bayes model that's an 8.5% improvement. [Appendix – 10.2.2 XGBoost]

5.3 Model 3 – Neural Networks

Neural Networks are built of simple elements called neurons, which take in a real value, multiply it by weight, and run it through a non-linear activation function. The process records one at a time and learns by comparing their classification of the record with the known actual classification of the record. The errors from the initial classification of the first record are fed back into the network and used to modify the network's algorithm for further iterations.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 254750, 64)        1344
_________________________________________________________________
dense_1 (Dense)              (None, 254750, 128)       8320
_________________________________________________________________
dense_2 (Dense)              (None, 254750, 256)       33024
_________________________________________________________________
dense_3 (Dense)              (None, 254750, 512)       131584
_________________________________________________________________
dense_4 (Dense)              (None, 254750, 512)       262656
_________________________________________________________________
dense_5 (Dense)              (None, 254750, 11)        5643
=================================================================
Total params: 442,571
Trainable params: 442,571
Non-trainable params: 0
_________________________________________________________________
```

Figure 5.1 – Neural Network Model Summary

In this neural network model, there are **six** dense layers as shown in Figure 5.1, the final layer is an output layer with an activation function "**SoftMax**". SoftMax is used here because each patient must be classified in one of the 11 levels in the Stay variable. In this model, increasing the number of neurons from each layer to the other layer, will increase the hypothetical space of the model and try to learn more patterns from the data. There are a total of **442,571** trainable parameters, this can be observed in Figure 5.1. Every layer is activated using "**relu**" activation function because it overcomes the vanishing gradient problem, allowing models to learn faster and perform better.

Before training the model, data were scaled, converted into a sparse matrix, and portioned into 80% as a train set and 20% as a test set. This neural network model was compiled using "**categorical_crossentropy"** as a function of loss because the target variable is categorical and "S**GD"** as an optimizer argument. Initially, the model was trained using portioned train data with 20 epochs and validation set argument set at 20%. In Figure 5.2 showing TensorBoard, we can observe that the model is overfitting from the 4th epoch. So, the model is retrained by setting epochs to 4.

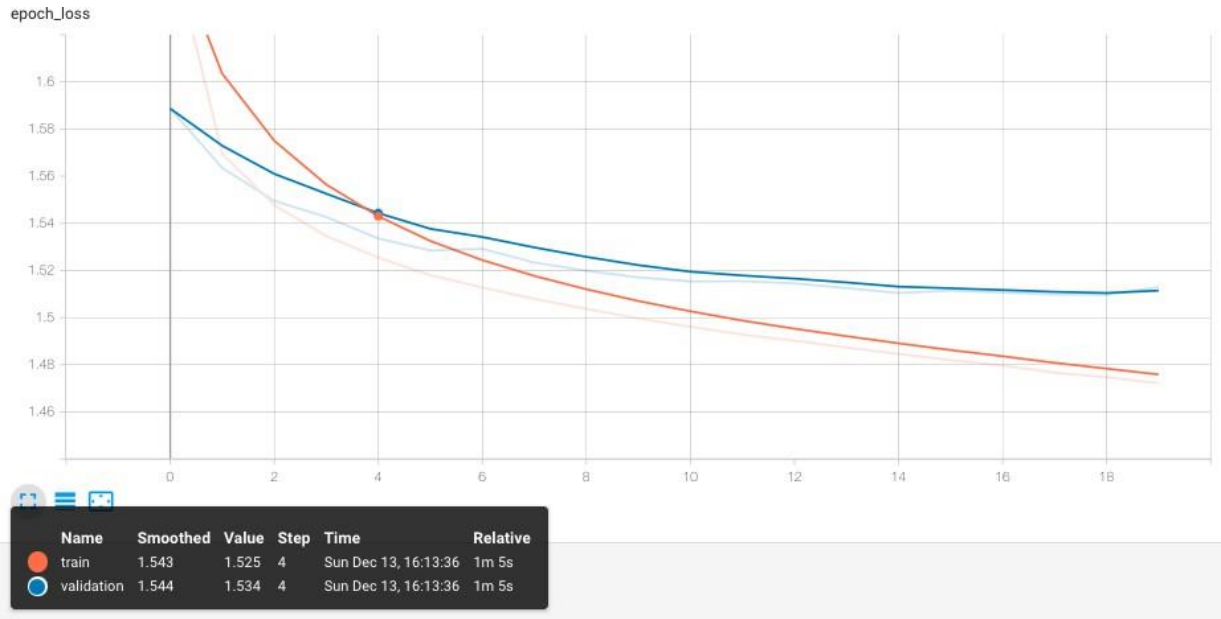


Figure 5.2 – Epoch Loss Graph

Finally, evaluating the model with a test set yields an accuracy score of **42.05%**. Neural Networks supposedly performs better than any other models. But because of the smaller dataset, it was not able to learn more accurately than the XGBoost model. [Appendix – 10.2.3 Neural Network]

6. Prediction and Results

In the Naïve Bayes model, patients are more likely to be misclassified. This model is biased towards the duration of 21-30 days, it has classified 72,206 patients for this level (can be observed in Table 6.1).

Table 6.1 – Number of observations classified into different levels of Length of Stay from all models

Length of Stay	Predicted Observations from Naïve Bayes	Predicted Observations from XGBoost	Predicted Observations from Neural Network
0-10 Days	2598	4373	4517
11-20 Days	26827	39337	35982
21-30 Days	**72206**	58261	61911
31-40 Days	15639	12100	8678
41-50 Days	469	61	26
51-60 Days	13651	19217	21709
61-70 Days	92	16	1
71-80 Days	955	302	248
81-90 Days	296	1099	1165
91-100 Days	2	78	21
More than 100 Days	4322	2213	2799

Whereas the other two models XGBoost and Neural Networks are predicting mostly similar Length of Stay for the patient, we can see this similarity for the first five cases in Table 6.2. In Table 6.1, we can see that the observations classified by both these models are marginally similar.

Table 6.2 – Predicted Length of Stay for first five cases from different models

case_id	Length of Stay predicted from Naïve Bayes	Length of Stay predicted from XGBoost	Length of Stay predicted from Neural Networks
318439	21-30	0-10	0-10
318440	51-60	51-60	51-60
318441	21-30	21-30	21-30
318442	21-30	21-30	21-30
318443	31-40	51-60	51-60

Examining these predictions, many of the patients are staying in the hospital for 21-30 days and very few people are staying for 61-70 days. As far as the distribution of Length of Stay is concerned, 13% of the patients are discharged from the hospital within 20 days and 1% of the overall patients are staying in the hospital for more than 60 days.

7. Future Insights

- **Smart Staffing & Personnel Management:** having a large volume of quality data helps health care professionals in allocating resources efficiently. Healthcare professionals can analyze the outcomes of checkups among individuals in various demographic groups and determine what factors prevent individuals from seeking treatment.
- **Advanced Risk & Disease Management:** Healthcare institutions can offer accurate, preventive care. Effectively decreasing hospital admissions by digging into insights such as drug type, conditions, and the duration of patient visits, among many others.
- **Real-time Alerting: Clinical Decision Support (CDS):** applications in hospitals analyzes patient evidence on the spot, delivering recommendations to health professionals when they make prescriptive choices. However, to prevent unnecessary in-house procedures, physicians prefer people to stay away from hospitals
- **Enhancing Patient Engagement:** Every step they take, heart rates, sleeping habits, can be tracked for potential patients (who use smart wearables). All this information can be correlated with other trackable data to identify potential health risks.

8. Conclusion

In this project, different variables were analyzed that correlate with Length of Stay by using patient-level and hospital-level data.

By predicting a patient's length of stay at the time of admission helps hospitals to allocate resources more efficiently and manage their patients more effectively. Identifying factors that associate with LOS to predict and manage the number of days patients stay, could help hospitals in managing resources and in the development of new treatment plans. Effective use of hospital resources and reducing the length of stay can reduce overall national medical expenses.

9. References

- **Janatahack: Healthcare Analytics II** - *Analytics Vidhya* - Link
- **What Is Naive Bayes Algorithm in Machine Learning?** - *Rohit Dwivedi* - Link
- **Naïve Bayes for Machine Learning – From Zero to Hero** - *Anand Venkataraman* - Link
- **XGBoost Parameters** - *XGBoost Documentation* - Link
- **Predicting Heart Failure Using Machine Learning, Part 2**- *Andrew A Borkowski* - Link
- **How to Tune the Number and Size of Decision Trees with XGBoost in Python** - *Jason Brownlee* - Link
- **Big Data Analytics in Healthcare That Can Save People** - *Sandra Durcevic* - Link
- **Learning Process of a Neural Network** – *Jordi Torres* - Link

10. Appendix – Code

10.1 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')
```

10.2 Models

10.2.1 Naive Bayes Model

```
from 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)

Acurracy: 34.55439015199096
```

10.2.2 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)
```

```
Accuracy: 43.047355859816605
```

10.2.3 Neural Network

```
from keras.utils import to_categorical
#Sparse Matrix
a = to_categorical(y_train)
b = to_categorical(y_test)

model = Sequential()
model.add(Dense(64, activation='relu', input_shape = (254750, 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'))
```

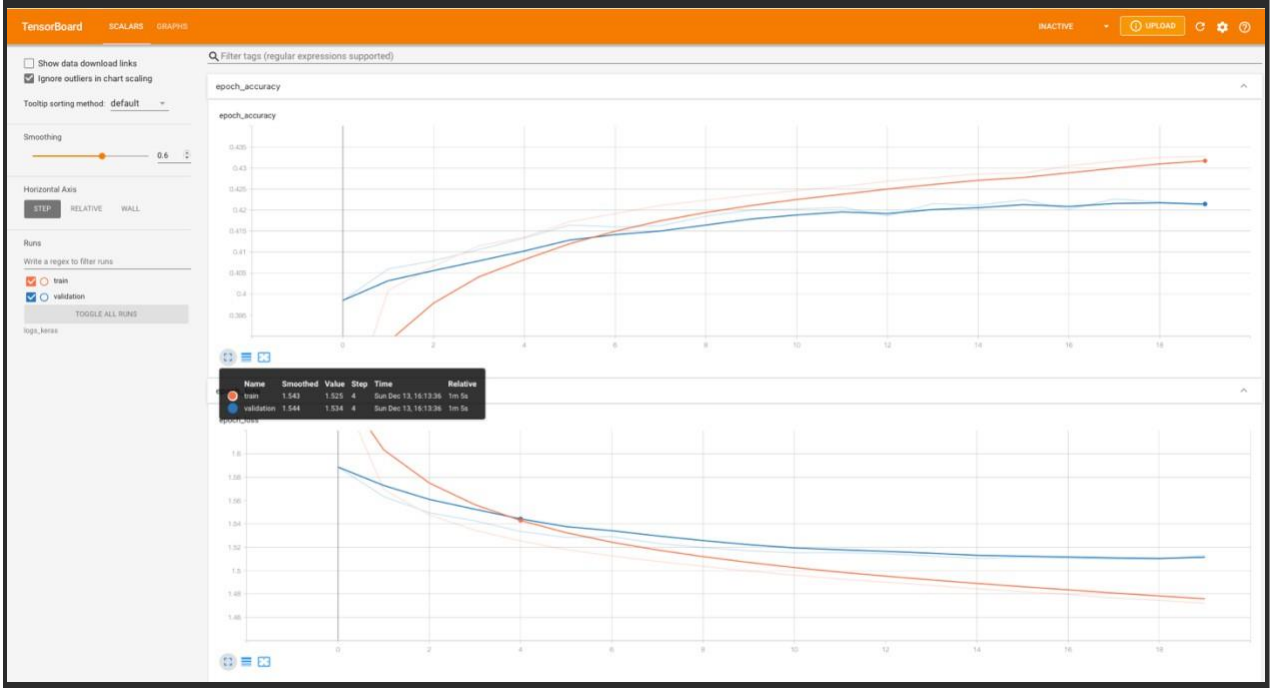
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 254750, 64)        1344
_________________________________________________________________
dense_1 (Dense)              (None, 254750, 128)       8320
_________________________________________________________________
dense_2 (Dense)              (None, 254750, 256)       33024
_________________________________________________________________
dense_3 (Dense)              (None, 254750, 512)       131584
_________________________________________________________________
dense_4 (Dense)              (None, 254750, 512)       262656
_________________________________________________________________
dense_5 (Dense)              (None, 254750, 11)        5643
=================================================================
Total params: 442,571
Trainable params: 442,571
Non-trainable params: 0
```

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

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

# Genrating tensorboard
!tensorboard --logdir logs_keras
```




```
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)

Model Evaluation
1991/1991 [==============================] - 2s 1ms/step - loss: 1.5071 -
accuracy: 0.4205

[1.5071101188659668, 0.4204716682434082]
```

Below GitHub link navigates to project repository containing python notebook file, markdown file with outputs (in HTML format), TensorBoard screenshot, and datasets.
GitHub Link - https://github.com/akashbhor1356/Healthcare_Analytics.git