Deep learning: Predicting Length of Stay using Healthcare Analytics II Kaggle dataset

Jesus Lopez
jgl5@hood.edu
Hood College
Frederick, MD, USA

ABSTRACT

Of increasing importance in healthcare management in the United States is the efficient allocation of hospital resources in order to increase patient outcomes. One particular area of interest is accurately predicting patients' Length of Stay at a hospital; based on presumed Length of Stay estimates, healthcare providers may then make choices that may ultimately lead to decisions which increase positive care outcomes for patients, especially those who may be at risk of long hospitalization stays. This project explored whether a simple deep learning model could be made to make accurate predictions about the Length of Stay of a patient given some categorical features and data from Kaggle's Healthcare Analytics II data-set. Two models were developed for the purpose of the project; the first, engineered towards a multi-class classification prediction in terms of Length of Stay intervals, while the second was engineered towards a binary classification between an estimated Short or Long Stay. Ultimately, both models resulted in low-accuracy predictions which showed high amounts of inaccuracy during validation. The results may be explained by fundamental flaws in the data-set that potentially included too few categories of sample data, which led to the model being unable to find any patterns of significance to use for making predictions of any utility to healthcare providers. Importantly, the results of this experimental project could serve as a useful evaluation of the sort of feature engineering and/or curated data that would be needed to make much better models in the future.

CCS CONCEPTS

• **Applied computing** → *Consumer health*; *Health informatics*; • **General and reference** → **Experimentation**.

KEYWORDS

healthcare analytics II dataset, length of stay, deep learning, binary classification, multiclass classification, categorical classification

ACM Reference Format:
Jesus Lopez. 2021. Deep learning: Predicting Length of Stay using Healthcare Analytics II Kaggle dataset . In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Conference'17, July 2017, Washington, DC, USA
© 2021 Association for Computing Machinery.
This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in *Proceedings of ACM Conference (Conference'17)*, https://doi.org/10.1145/nnnnnnn.nnnnnnn.

1 INTRODUCTION

The potential of deep learning models to find a place as aids to the management of healthcare in the United States is indisputable. With the rising costs of healthcare there is pressure on healthcare facilities in the United States to optimize the efficient use of resources to cut healthcare costs while maintaining or improving patient outcomes [2]. The fact that deep learning is most suitable for finding patterns among large amounts of data and that healthcare records are often large and have countless pieces of information means that the domain of healthcare is ripe for the application and experimental use of deep learning. One area which deep learning would be a boon to healthcare managers would be in the prediction of patients' Length of Stay (LOS). Research shows a correlation between patient outcomes and patients' length of stay [4, 5] . Investigations by [1] support the idea that healthcare workers make different decisions about patient care if they expect a patient will have longer stays. However, the ability to make predictions about patients' Length of Stay remain bound to the experience of the healthcare professional interacting with the patient [1] and there is currently a lack of any definitive industry tool. However, the area is one which is receiving attention from groups focused on applied deep learning [6]. Sharing in the pursuit, this project experimented with the use of some simple deep learning models to generate predictions of a high enough accuracy to potentially be useful in a clinical setting. Specifically, one simple model was four-layer sequential model that would accept the categorical data that included important pieces of information about a patient's severity of illness, their age, and other potentially useful categories which a model might discover a pattern to base predictions which would be outputted as a probability of eleven possible intervals of Length of Stay. The second model would be similar but would instead attempt to reduce the problem into a binary classification problem by generating a prediction that only stated whether a patient would be expected to have a hospital stay greater or less than ten days. This approach, will hopefully have more success as a similar approach, albeit with an entirely different and more sophisticated model has shown to have over 77 percent accuracy[6].

2 RESULTS

On training and validation the best achieved metrics, as shown in Figure 1 was an accuracy of 29 percent, with all features, except the Admission Deposit feature taken as input. Meaning that in 29 percent of the times, model one achieved a probability distribution that accurately predicted the correct LOS interval. Effectively, this meant that even the best version of model one would not be able to solve the problem. For model two, the results were much higher than in model one, but still poor. Accuracy ranged from 81 percent

99 percent, but loss scores were even worse than in model one. Like in model there was an experimentation cycle of removing features, adding layers, changing input dimensions, and optimizer functions. However, many of the changes resulted in degrading accuracy and only moderate improvement to loss scores. In the end, the best results for model two, shown in Figure 2, was an accuracy of 100 percent, but with an atrocious loss score. As a result of the loss score, it was clear that model could not be considered practical for clinical settings. As shown by the visualization the results throughout the project, as shown on Figure 3, no model was able to satisfy the objectives of the project.

```
2021-12-07 13:48:21.882349: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:185] None of the MLIR Op
red 2)
Epoch 1/5
781/781 - 5s - loss: 1.8774 - accuracy: 0.2934 - val_loss: 1.8736 - val_accuracy: 0.2971
Epoch 2/5
781/781 - 5s - loss: 1.8705 - accuracy: 0.2956 - val_loss: 1.8706 - val_accuracy: 0.2937
Epoch 3/5
781/781 - 4s - loss: 1.8691 - accuracy: 0.2967 - val_loss: 1.8663 - val_accuracy: 0.2975
Epoch 4/5
781/781 - 4s - loss: 1.8685 - accuracy: 0.2968 - val_loss: 1.8665 - val_accuracy: 0.2985
Epoch 5/5
781/781 - 4s - loss: 1.8680 - accuracy: 0.2965 - val_loss: 1.8658 - val_accuracy: 0.2959
<keras.callbacks.History at 0x7fb6dfc69150>
```

Figure 1: best Results of model one

```
 1/15
28 [==============================] - 4s 5ms/step - loss: 7.1512 - accuracy: 0.9998
 2/15
28 [==============================] - 3s 5ms/step - loss: 7.1410 - accuracy: 1.0000
 3/15
28 [==============================] - 3s 4ms/step - loss: 7.1410 - accuracy: 1.0000
 4/15
28 [==============================] - 3s 4ms/step - loss: 7.1410 - accuracy: 1.0000
 5/15
28 [==============================] - 3s 4ms/step - loss: 7.1410 - accuracy: 1.0000
 6/15
28 [==============================] - 3s 4ms/step - loss: 7.1410 - accuracy: 1.0000
 7/15
28 [==============================] - 3s 4ms/step - loss: 7.1410 - accuracy: 1.0000
 8/15
28 [==============================] - 3s 5ms/step - loss: 7.1410 - accuracy: 1.0000
 9/15
28 [==============================] - 3s 5ms/step - loss: 7.1410 - accuracy: 1.0000
 10/15
28 [==============================] - 3s 5ms/step - loss: 7.1410 - accuracy: 1.0000
 11/15
28 [==============================] - 3s 5ms/step - loss: 7.1410 - accuracy: 1.0000
 12/15
28 [==============================] - 3s 5ms/step - loss: 7.1410 - accuracy: 1.0000
 13/15
28 [==============================] - 3s 5ms/step - loss: 7.1410 - accuracy: 1.0000
 14/15
28 [==============================] - 3s 5ms/step - loss: 7.1410 - accuracy: 1.0000
 15/15
28 [==============================] - 3s 4ms/step - loss: 7.1410 - accuracy: 1.0000
s.callbacks.History at 0x7f966c6e6cd0>
```

Figure 2: Best results of Model two

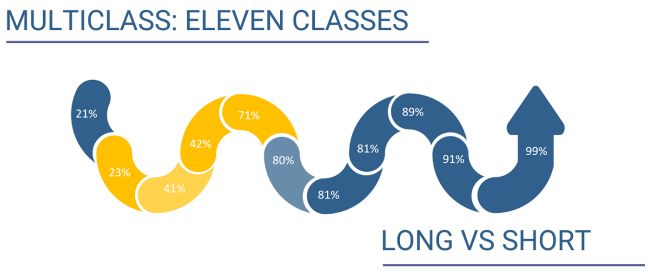

Figure 3: Shows moderate illness admissions are strongly prevalent across LOS intervals

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 128)               2560
_________________________________________________________________
dense_1 (Dense)              (None, 128)               16512
_________________________________________________________________
dense_2 (Dense)              (None, 128)               16512
_________________________________________________________________
dense_3 (Dense)              (None, 128)               16512
_________________________________________________________________
dense_4 (Dense)              (None, 11)                1419
=================================================================
Total params: 53,515
Trainable params: 53,515
Non-trainable params: 0
_________________________________________________________________
```

Figure 4: Model One Summary, Text Display

3 DISCUSSION

3.1 Model design

The data-set used, Healthcare Analytics II, found on Kaggle, has approximately 415,000 samples in total, with two comma-separated value (CSV) files splitting the total data-set into a training and testing set. The former had a column with samples for patient Length of Stay, while the test set did not have a column for Length of Stay. Therefore, only the CSV file containing the 318,000 samples were used for the training of the two models used for this project. The data-set had a total of eighteen columns, of which one column was the target column (Length of Stay), one column, "case id" would be useless due to being unique. During the initial phase of Exploratory Data Analysis, there appeared to be some useful categories to serve as features, in particular the Severity of Illness category and Age ; even the rest of the categories seemed to have potential as features to better tune and train the models. In the initial rounds of model design, all the possibly useful features, shown in Figure 5 were at first utilized to train the first model, shown in Figure 6, which is a sequential model built with Keras using four layers. At the final layer, an output would contain a probability of the most likely interval under which a patient's Length of Stay would fall under. In other words, because the data-set expressed Length of Stay in intervals, such as zero to ten days, eleven to twenty-one days, and so on, up until a period of more than one hundred days, the model would have to answer in terms of percentage; for example, the LOS for a patient would be 10 percent likely to be zero to ten days, 15 percent to be twenty to thirty-one days, and so on. The outcomes realized by model one was initially accuracy ranging from 17 to 20 percent, with stable, but high loss values. Subsequently, additional layers of different input dimensions and activation functions were tried in a trial-and-error fashion, but ultimately had worse accuracy, sometimes ranging as low as 10 percent. Then, different features were subtracted from the input. For example, the Admission Deposit, the City-Patient-Code, and Bed Grade, were removed because their potential could be ambiguous since the categorical values for them did not have a self-contained method for evaluating what was best. Specifically, the values for Bed Grade were given from 1.0

to 4.0 but the data did not express whether 1.0 stood for the best bed grade, the worst bed grade, or vice-versa. Therefore, from a feature engineering point, it seemed best to remove feature that might be adding confusion to the model's attempt to finder a useful pattern for generating better representations. Even before actual

```
 #   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
```

Figure 5: Data-set Columns

Training the Neural Network

```
# define model to tackle this single-label, binary classification problem
model = Sequential()
model.add(Dense(128, input_dim = X_train.shape[1] , activation = 'relu'))
model.add(Dense(128, activation = 'relu'))
model.add(Dense(128, activation = 'relu'))
model.add(Dense(128, activation = 'relu'))
model.add(Dense(11, activation='softmax'))

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

Figure 6: Model One Raw Code

testing there was an expectation that the second model would be more fruitful since it would work under the theory that simplifying the problem space would generate better results. Specifically, by turning what was, with model one, a multi-class classification problem,into a binary classification problem by having the model only attempt to predict whether a patient would have a LOS greater than ten days or not. For the training of model two, the encoding of the Length of Stay did not use hot-one-encoding, as in model one, but instead used simple integer encoding where all the values in the Stay column were turned to zero if the categorical value corresponded to an interval of less than ten days, and an integer of one if the categorical value corresponded to an interval of more than ten days. This has the benefit of reducing the overall program space and allow for more resource efficient training epochs. For this model, shown in Figure 7 the same number of layers were applied at first, though different numbers of layers and dimensions were tried just as in the experimentation with model one.

Model

```
def get_basic_model():
    model = tf.keras.Sequential([
    normalizer,
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(2, activation='relu' )
  ])
    model.compile(loss='binary_crossentropy', optimizer='rmsprop', metri
cs=['accuracy'])
    return model
```

```
model = get_basic_model()
model.summary()
model.fit(numeric_features, target, epochs=15, batch_size=500)
```

Figure 7: Model Two Raw Code

3.2 Analysis of the data

Given the results, investigation of the underlying data seemed appropriate. Using the functionality of the Python programming language's Pandas library, visualizations for the data showed some potential flaws in the data-set itself. As shown by Figure 8 there seems to be no statistically significant correlation between the various categories which served as features for the model to train on. This was surprising given that one would expect at least a relationship between LOS and Severity of Illness, as shown in Figure 9. While this does not conclusively prove that the data-set's categories could not be used to generate a useful pattern - after all, the power of deep learning is the fact that the models can use mathematical operations to create abstract representations that a human may not be able to create themselves[?] – it does seem that this data-set, despite being used for a Kaggle competition for the same task as this project, may not be ideal. Further issues occur with the imbalance in the data. The representation between the admission categories, meaning the departments there are data for is very skewed toward Gynecology. Consequently, the danger of over-fitting was present and thus might explain the high accuracy score, but abysmal loss score of model two. However, in model one, given the very poor accuracy, but better loss score, the opposite case, under-fitting, might be occurring if the model could not capture the proper relationship between the other admission types and the other features.

3.3 Future Work

Any continuation of the work with this data-set would either require far more extensive manipulation of the data-set, such as using technique of data augmentation, or entirely different models, such as a model such as used the proposed Autoencoder+DNN model used by Zebin et al. [6] in their work on this very task, but using full medical records. In addition, an altogether new data-set might

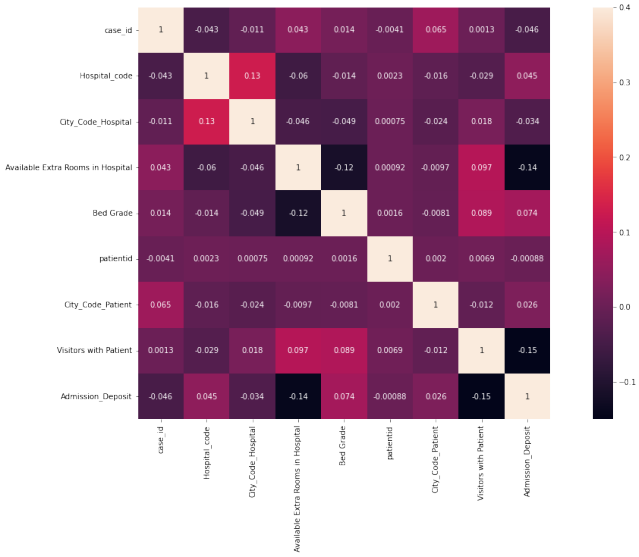

Figure 8: Matrix shows a lot of negative correlation and very little, if any, statistically significant correlation between the various categories

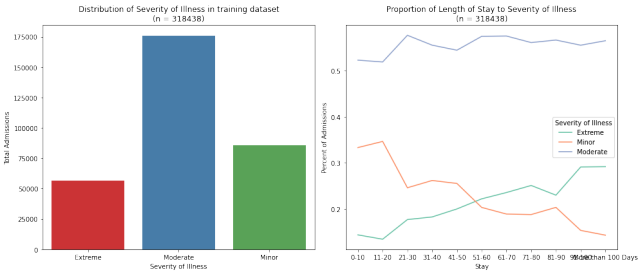

Figure 9: Shows moderate illness admissions are strongly prevalent across LOS intervals

be worth serious consideration, such as MIMIC-III, which has been used by other deep learning groups [3].

ACKNOWLEDGMENTS

Very special thanks goes to fellow Hood College student, Ragi Ginu, a graduate student workings towards completion of a masters degree at the College. She was instrumental to this author's efforts to have working code for the multi-classification model. Without her efforts, the author would have spent countless more hours trying to code a solution to address an error where the data was not being properly accepted by the input layer of the model.

CODE AVAILABILITY

The source code including both training and testing of our networks, with pre-trained model and demo data, are available ats https://github.com/Jgl5/Deep-Learning-Course

REFERENCES

[1] Stoopes S et al Grover CA, Sughair J. 2018. Case management reduces length of stay, charges, and testing in emergency department frequent users. *West J Emerg Med* (March 2018). https://doi.org/10.5811/westjem.2017.9.34710

[2] Lavezza A et al Hoyer EH, Friedman M. 2016. Promoting mobility and reducing length of stay in hospitalized general medicine patients: a quality-improvement project. *J Hosp Med* (May 2016). https://doi.org/10.1002/jhm.2546

[3] Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-Wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. MIMIC-III, a freely accessible critical care database. *Scientific data* 3, 1 (2016), 1–9.

[4] Sun R. McDermott KW, Elixhauser A. 2017. Trends in hospital inpatient stays in the United States. *HCUP Statistical Brief 225* (June 2017). https://hcup-us.ahrq.gov/reports/statbriefs/sb225-Inpatient-US-Stays-Trends.pdf

[5] Pizzo E et al Rojas-García A, Turner S. 2018. Impact and experiences of delayed discharge: a mixed-studies systematic review Link to Exit Disclaimer. Health Expect. 54, 2, Article 1 (Feb. 2018), 41 pages. https://doi.org/10.1111/hex.12619

[6] Tahmina Zebin, Shahadate Rezvy, and Thierry J. Chaussalet. 2019. A deep learning approach for length of stay prediction in clinical settings from medical records. In *2019 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*. 1–5. https://doi.org/10.1109/CIBCB.2019.8791477