Predictive Model for Patient Appointment Cancellations

Group 23: Adam Chin, Colleen Kim, Lesley Liu, Sarah M'saad

Project Task

The goal of our project is to identify patient engagement patterns on an application used by hospitals, and created by Medumo. By doing so, we aim to reduce the number of unused appointments, and enable hospitals to efficiently accommodate patients without overbooking or underbooking.

Related Work

In a similar study also concerned with cancelled hospital appointments, a model was built based on the classification algorithm Gradient Boosting Machine in [1]. Other studies have tried to address cancellations in other industries such as the hotel industry using linear regression in [2].

Approach

The model we used was a neural network, specifically a Multilayer Perceptron (MLP) neural network, with cross entropy as our loss function. We classified patient outcomes as Show/No-Show dependending on inputs from events 4 days before the milestone date (appointment date). The input includes: relevant notification messages, number of clicks, Node_Viewed, rescheduling. The output corresponds to the flag 0 for 'Show' and 1 for 'No Show'. The models for computing the loss were: cross entropy, and backpropagation for neural networks.

Metric

Our plan for data processing was to extract relevant UI clicks, relevant messages reminders and so forth to be seen as events that affect the decision taken by the patient to show up to their appointment. The relevant data was represented in a matrix, with each row representing a patient, and each column representing a feature such as the total number of UI clicks.

Dataset

Our dataset has four different tables: Enrollment Data, Engagement Events, Train, and Test. Enrollment data lists various data about the patients including their procedure date, and whether or not they opted in to email or sms notifications among other information. Engagement Events list all the interactions involving the patients. These events include the patient's id, and a short description of the event. Our train data set includes example data for our program to train on and has 1996 samples, but we split the training data into training, validation, and testing sets to test our model. This set includes the hospital ID, patient ID, registration date, procedure date, and a No Show/Late Cancel flag of whether or not they successfully completed their procedure. The No Show/Late Cancel flag is set to 0 if they successfully completed their procedure, and set to 1 if they didn't show up or cancelled within 3 days. The test dataset is formatted the same as the train data, but all the No Show/Late Cancel flags are initialized to -1, and there are 222 samples.

Index Id	Hospital Id	Patient Id	Registration Date	Procedure Date	No Show/Late Cancel Flag
0	18	12345	02/15/2018	02/15/2018	0
1	18	12346	03/02/2018	07/09/2018	0
2	21	54321	02/12/2018	03/16/2018	1

Figure 1. An example of how our Test and Train data are formatted.

Preprocessing Data

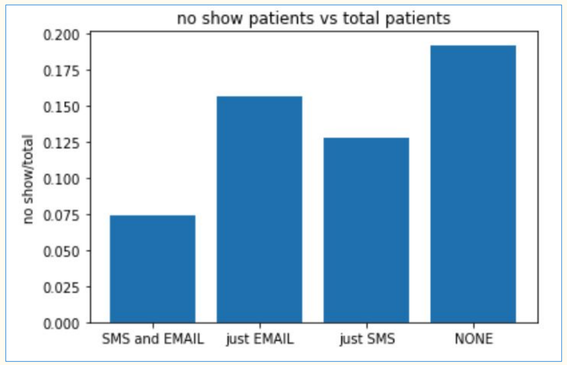


Figure 2. Graph demonstrating a possible correlation between user email and sms activity, and appointment absence

Initial Tests

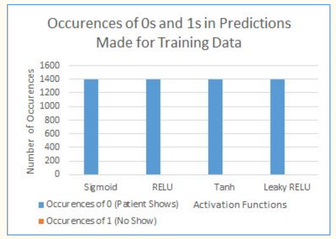
In the beginning we focused on the history and accuracies of the training and validation sets we built. Using our MLP with different activation functions, we received high but similar accuracies. The activation functions we tested were sigmoid, RELU, tanh, and leaky RELU.

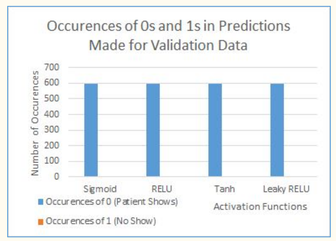
Our MLP is a fully connected two layer neural network. For the first layer activation we compute the dot product of the training data and our weight, then add our bias. Next we push it through the first layer's activation function.

The next layer is where we used cross-entropy loss with log-sum-exp. This loss function is similar to logistic regression, but more preferred over other loss functions for MLPs since it's faster in converging than linear regression. After we complete calculating the loss, we begin the backwards pass. For this, we compute the derivatives of the selected activation function and we return the found loss and gradients based on the derivatives.

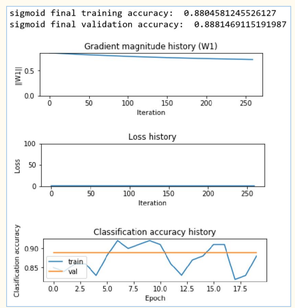
To train our model, we passed our X and y data that we get from splitting the data using stochastic gradient descent to do so. We chose a learning rate of 0.01 and a decay of 0.95. Then we print out the accuracy of each activation function to compare how well our model was trained. Finally, after we trained the model, we passed in our test.csv into the predict function to see how well our model was trained.

Initial Predictions





Figures 3, 4, 5. To the immediate left, a bar graph made from collecting the predictions our MLP model made for the training data, and under it the predictions for the validation data. To the right of that graph histories for the gradient magnitude, loss, and accuracies from using sigmoid



Final Test

Our initial results seemed very successful, so we were excited to try running our model on the testing data set on Kaggle. We first ran our model with sigmoid activation and got a score of 50% on Kaggle. We then tried running it again with RELU, tanh, and leaky RELU, but still only got a score of 50%.

After a quick inspection of predicted results, it turns out our model predicts that every patient will show up and our accuracy was just the ratio of total number of show patients over total number of patients. This made us realize that our data was highly imbalanced.

Our first approach was to make the data more balanced. We tried to increase the number of patients that didn't show up, and we also tried decreasing the number of patients that did show up just to get rid of some bias for our MLP. However, after much trial and error, we were not doing much better.

Towards the end, we discovered that using one class SVM with a non-linear kernel (RBF) is best for classifying unbalanced data, so we made sure to set the weight_class parameter to balanced to keep every input as important as the rest. This definitely improved our results on Kaggle with the testing data set, and our SVM scored 0.56931, and had an accuracy of 70%.

Confusion Matrix

As you can see from our confusion matrix of the SVM model, the number of true positives is pretty high. This means that there are 253 correct predictions from our model. However, the number of true negatives are pretty low so our model couldn't guess as many No shows as we would like it to have predicted.

```
[[253 105]
 [ 16  26]]
```

Looking Forward

Looking forward, the most ideal step would be gathering more data on instances of patients missing appointments. We would also want to explore more methods being used currently to help with accuracy paradox. Our SVM model does better than our MLP model, but we believe its performance can be improved with the above.