Predictive Model for Patient Appointment Cancellations

Group 23:Adam Chin, Colleen Kim, Lesley Liu, Sarah M'Saad
{archin,ryuc,lliu,sarms}@bu.edu

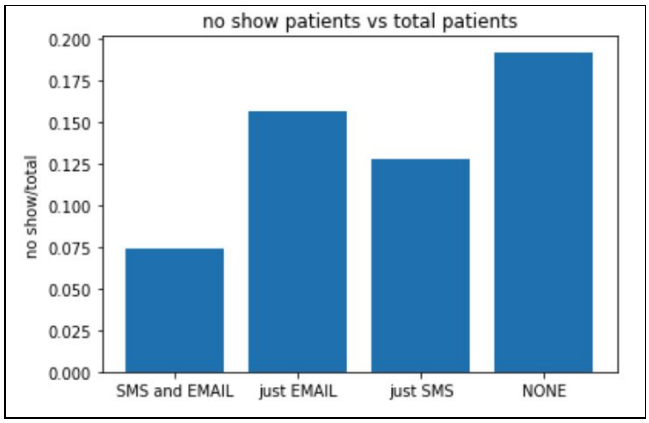

Figure 1. Graph demonstrating a possible correlation between user email and sms activity, and appointment absence.

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.

Dataset and Evaluation Metric

Our dataset has four different tables: Enrollment Data, Engagement Events, Train, and Test. Enrollment data lists various data about the patients including hospital ID, patient ID, the date they registered, the date of their procedure, whether they turned on email notifications or sms notifications, their date of birth, and their gender. Engagement Events lists all the interactions involving the patients. These events include the patient's ID, the date and time it occurred, the event name, and possibly a short description of the event. Some events include a notification sent, a user viewed the node, the hospital rescheduled the user's appointment, etc. For each patient, these events are all chronologically dated up until their appointment date.

Our train data set includes example data for our program to train on. 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. (See Figure 2). Our training data set has 1996 samples while our test data set has 222 samples, but in order to see how our model was doing, we split the original training data into training, validation, and testing sets.

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

Metric

Our plan for data processing was to extract relevant UI clicks, relevant messages reminders, age 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.

Results

In the beginning we focused on the history and accuracies of the training, and validation sets we built from the original training data. We decided to use an MLP after finding a probable correlation between user email and sms activity, and hoped that an MLP would be able to detect more data trends. (See Figure 1.) Using a MLP with different activation functions, we received high but similar accuracies.

Our MLP is a fully connected two layer neural network. The network has an input size based on the training data and a hidden layer of 10 neurons. Using train.csv, we split the data into train, and validation data; a 70:30 ratio, respectively. For the weights and biases we initialized weights to a very small random number and biases to 0. In our loss function, we input our train and validation data that we had split and we begin to calculate the layers of the neural net.

For the first layer activation we compute the dot product of X (the training data) and our weight, then add our bias. Next we push it through the first layer's activation function, which we choose and pass as a parameter. The activation functions that we tested were sigmoid, RELU, tanh, and leaky RELU. After the activation function, is the next layer.

The next layer is where we used cross-entropy loss with log-sum-exp. This loss function is similar to logistic regression but it is more preferred over other loss functions for MLPs. This is because cross-entropy is faster in converging by quickly learning through gradient descent, compared to mean squared error in 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, and we use the 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.

The first activation function we used and tested in our model was a sigmoid function. (See Figure 3.) We had a training accuracy of 88%, and a validation accuracy of 89%.

The next test we did was with a rectified linear unit (RELU) activation function. (See Figure 4.) Similar to the sigmoid function, the RELU function also had high training and validation accuracies, 88% for both. Because of the similar accuracies and loss histories, we wanted to test other activation functions.

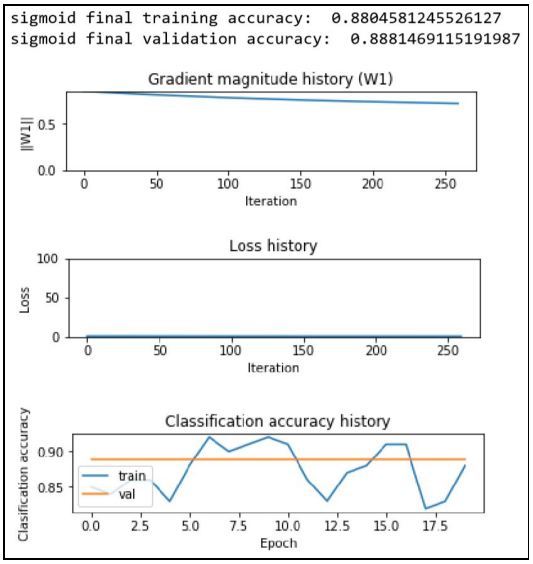

Figure 3. Histories for the gradient magnitude, loss, and accuracies for training and validation using sigmoid.

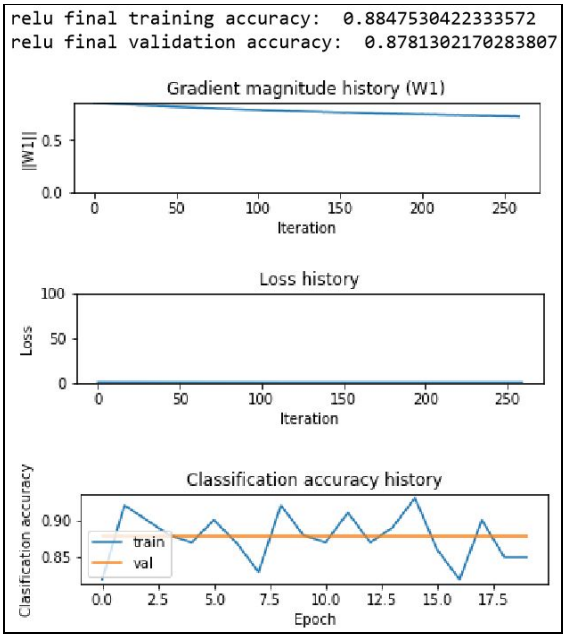

Figure 4. Histories for the gradient magnitude, loss, and accuracies and validation using RELU.

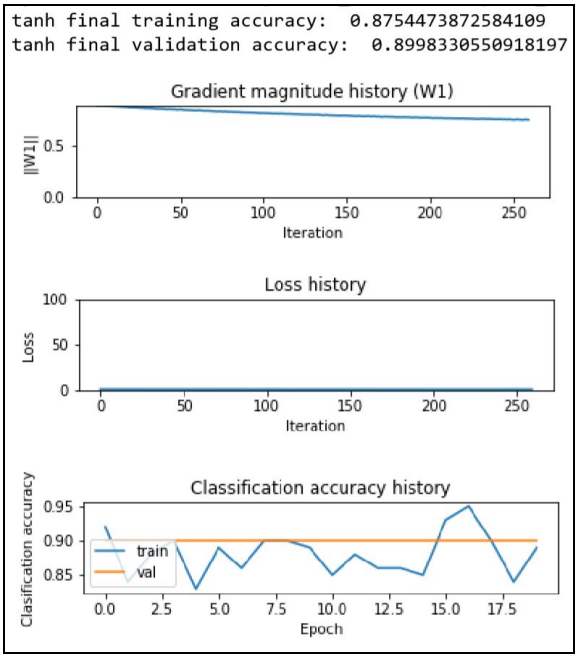

Figure 5. Histories for the gradient magnitude, loss, and accuracies and validation using tanh.

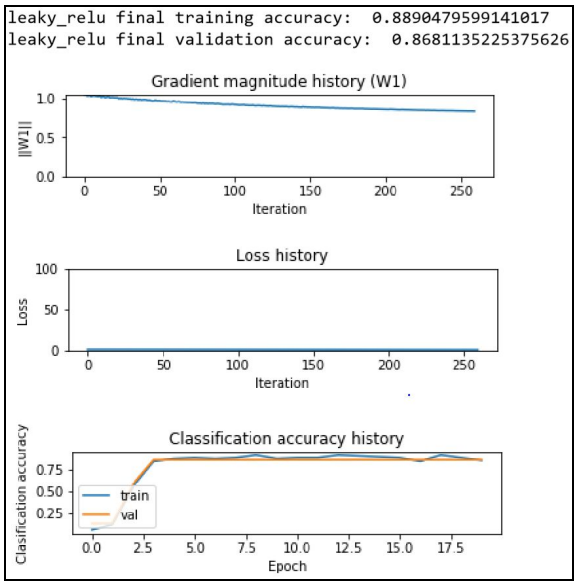

Figure 6.Histories for the gradient magnitude, loss, and accuracies and validation using leaky RELU.

We ended up testing our model with a tanh activation function and a variant of the RELU, the leaky RELU function (See Figures 5 and 6, respectively.) Both activation functions also had high training and validation accuracies. Tanh had a training accuracy of 88%, and a validation accuracy of 90%, while leaky_relu had a training accuracy of 89%, and a validation accuracy of 87%. Both of their gradient magnitude histories and loss histories were similar to each other and to the histories we got from sigmoid and RELU.

With these accuracies, we were very excited to try running our model on the testing data set and submit it to Kaggle, the platform being used to communicate with a representative from Medumo, and test models on the testing data. We first ran our model with the sigmoid activation function, and got a score of 50% as we submitted the results to Kaggle. We ran the model on the testing data set again, with RELU, tanh, and leaky RELU activations, and for each function we got a testing accuracy of 50%. This made us question what was going wrong exactly with our model. 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 the total number of patients since our models were always predicting the total number of patients. (See Figures 7 and 8.) This made us realize that all of our data is highly imbalanced and we simply ran into Accuracy Paradox.

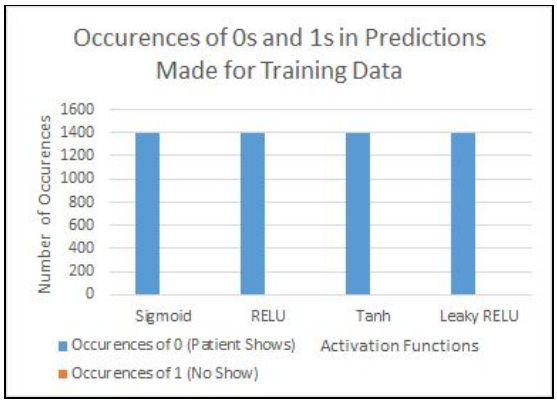

Figure 7. A table made from collecting the predictions our MLP model made for the training data.

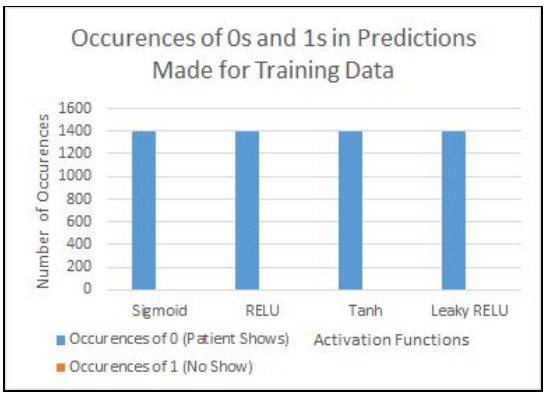

Figure 8. A table made from collecting the predictions our MLP model made for the training data.

So our first approach was to make our data as balanced as possible. From online sources we found that the method SMOTE (Synthetic Minority Over-sampling Technique) is an effective technique to fight data imbalanced. Next, we changed our training model to a SVC with non-linear kernel (**RBF**) and a linear one (SVM) since both of them are used with unbalanced data and high bias. Also, we changed the way we analyze our model. We no longer rely on the accuracy metric but we mainly use recall, precision and balanced accuracy metrics for evaluation. We also put emphasis on the results of the confusion matrix to better understand the distribution of our predictions.

After normalizing our features/scaling to a range of (0, 1), we get a mean balanced accuracy of 0.6 by using SVC with a 'radial basis function' which is considerably low. This encouraged us to use CARTs and specifically **Decision Trees** as our next training model. By weighting our dataset to a ratio 2:1 show and no show patients, and using a testing size of 0.3, the decision tree output gave a balanced accuracy with a mean of 0.82, a precision of 0.74 on the no show patients and a well distributed confusion matrix (see Table 1).

n = 529	Predicted 1	Predicted 0
Actual 1	316	45
Actual 0	39	129

Table 1. A confusion matrix using Decision Tree Classifier

This also gave our highest score on Kaggle (see Table 1) which definitely encouraged our decision that choosing a decision tree classifier as our training model is somewhat an effective choice.

#	Change	Team	Score	Entries
1	-	Group 6	0.87386	7
2	-	Colleen (ours)	0.57077	11

Table 2. Kaggle Leaderboard

Not to mention, we chose a sampling ratio of 2:1 and from Figure 9 we can see the balanced accuracy drops as we increase the ratio of majority to minority. We also decided on 0.6 ratio since it won't lead the Decision Tree to overfit our data which would eventually lead to faulty predictions.

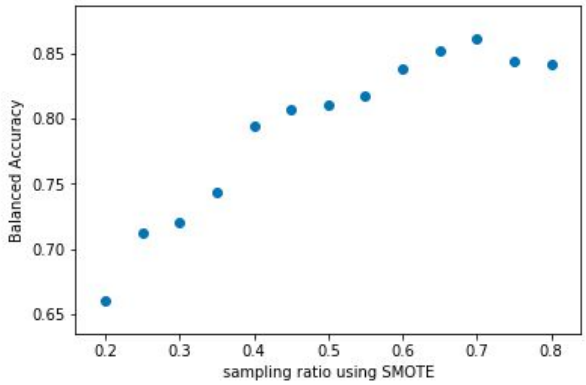

Figure 9. A scatter plot of the sampling ratio vs the resulting balanced accuracy of Decision Tree model

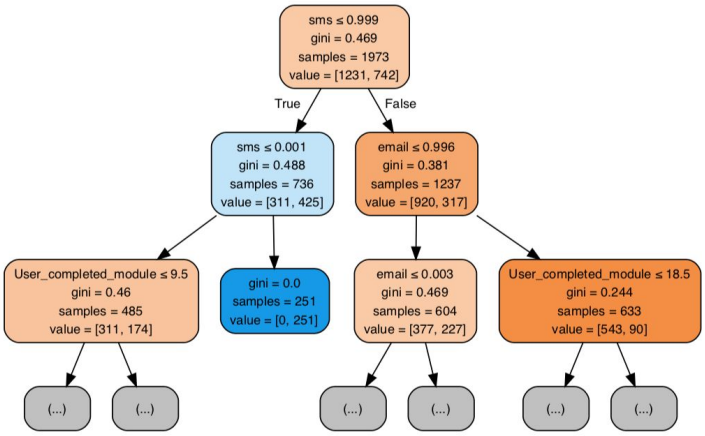

Figure 10. Decision Tree Graph visualization

The decision for our model is also highly related to the best score we've received from Kaggle. We wanted higher precision/recall metrics; however, when we achieve those results, the resulting Kaggle score is low which discouraged any further adjustments.

In conclusion, after testing with all these different models, we have noticed that SMS is weighted higher in deciding whether or not the patient will come in or not. On the other hand, features like email, age, the time between registration also affect the outcome. However, when it comes to the interaction with the application there doesn't seem to be much of an effect on the outcome of the patient's appointment. In the future, to further improve our predictions, collecting more data about patients that missed an appointment would help the problem of high bias that we've encountered. Another thing to note is that maybe we're missing key features to predict our data, so an improvement to our model would be extracting more information about the patients and see if other patterns emerge.

Timeline and Roles

Task	Deadline	Lead
Preprocess Data	11/10/18	Sarah
Implement Neural Network's (MLP)	11/14/18	Colleen
Implement Neural Network (MLP)	11/14/18	Lesley
Add Option for Logistic Regression Loss	12/4/18	Adam
Compare different loss results	11/27/18	Sarah
Try tanh activation layer	11/28/18	Colleen
Build SVM model	11/28/18	Sarah
Try a 3rd layer in MLP	12/8/18	Colleen
Try Leaky RELU activation layer	12/8/18	Lesley
Try to counter high bias	12/9/18	Sarah
In depth analysis of data	12/9/18	Lesley
Build Decision Tree Model	12/9/18	Sarah
Prepare Poster	12/10/18	All

References

1) C. Elvira, A. Ochoa, J. C. Gonzálvez, F. Mochó. Machine-learning-based no show prediction in outpatient visits. IJIMAI. 4(7):29-34, 2017.
2) N. Antonio, A. Almeida, L. Nunes. Predicting hotel booking cancellation to decrease uncertainty and increase revenue. Tour. Manag. Stud. 13(2);25-39, 2017.