**2021 Summer**

**Xuewei Zhang; Antonio Reybol Jr.; Colin Brence; Yilei Zhang**

PREDICTING OUTCOME OF FIRST-EPISODE PSYCHOSIS

**CIS9660 Final Project**

**Contents**

[**1.** **Introduction** 2](#_Toc79603649)

[**2.** **Data Exploration, Cleaning and Visualization** 2](#_Toc79603650)

[**2.1 Data Cleaning** 2](#_Toc79603651)

[**2.2 Data Visualization and Exploration** 3](#_Toc79603652)

[**3.** **Overview of Models** 4](#_Toc79603653)

[**3.1 Logistic Regression** 4](#_Toc79603654)

[**3.2 Decision tree** 5](#_Toc79603655)

[**3.3 Gradient Tree Boosting** 8](#_Toc79603656)

[**3.4 Neural Networks – Perceptron** 9](#_Toc79603657)

[**4.** **Model Evaluation** 10](#_Toc79603658)

[**5.** **Model Implementation** 11](#_Toc79603659)

[**6.** **Postmortem** 11](#_Toc79603660)

# **Introduction**

**Background of the data used for the project – where did it come from?**

Our dataset comes from a study of Psychosis patients in Glasgow, Scotland. The data was used to predict the outcome of Psychosis, a mental illness characterized by a disconnect from reality, with symptoms including hallucinations, delusions, and disorganized thought patterns. Psychosis has about a 3% lifetime incidence globally, and Schizophrenia, the most prevalent psychotic disorder, forms an estimated 1.6- 2.6% of healthcare costs in developed countries. It is a global disease with immense costs to healthcare systems and patient quality-of-life.

**What does each observation in the data represent?**

Each observation represents a patient and their associated information, including their PANSS Scale, housing status, relationship status, education, employment, and other demographic data.

**Why did your group choose this data?**

Psychosis is often a chronic illness that causes lifelong impairment, but emerging research offers hope that some patients may suffer from only a single episode without any relapse; however, features that predict relapse or remission are still a matter of ongoing research.

Thus, our problem essentially boils down to how accurately we can predict relapse and remission in psychosis patients, and the implications for treatment. In doing so, we can reduce the economic costs of psychosis and improve patient quality of life.

In the paper we reference, the authors achieved an AUC of 0.652 using an elastic-net regression model. We wanted to improve on the models the authors used and provide a basis for discussion of a recommender system for use in creating prognoses and treatment plans in the treatment of psychosis.

# **Data Exploration, Cleaning and Visualization**

## **2.1 Data Cleaning**

In the data cleaning stage, we first dropped values unrelated to our project, then we imputed missing values, and finally we created labels for each observation. Column Cohort was dropped because it is a tag column used by data collectors themselves. We also removed the observations where the target variable M6\_Rem, Y1\_Rem or Y1\_Rem\_6 is NaN.

Almost every column in the dataset contains missing values and the missing rates are from 1% up to 30%. Considering our dataset has a few observations, we decided not to remove the missing values. Instead, for numerical variables, we filled in null values by K-nearest neighbor method because usually the medical records are not missing randomly. And for categorical variables, we replaced missing values with the mode of each feature given the data is heavily skewed.

Since there are three target variables and we want to build a multiclass classifier based on them, we used the below chart to combine them into 7 labels. N means no, Y means yes. Out of 7 labels, label 4 and label 7 don’t exist in our dataset. Label 5 only has 3 observations, so we removed them for modeling purposes. Thus, the final labels for the classifier are label 0, 1, 2, 3 and 6.

Table

Description automatically generated

## **2.2 Data Visualization and Exploration**

Through data exploration we realized that we had many variables and not so many observations. Once we cleaned and prepared our data to run all our models, the data frame contained only 131 observations and 53 columns. We soon became encouraged to try dimension reduction because we anticipated the potential of our model’s overfitting. There were two types of graphs that we decided to create: a histogram and a correlation matrix.

Diagram

Description automatically generated

The histogram uncovered a few patterns such as uneven distributed data among the variables, particularly right skewed data. At this point, we immediately saw this as an opportunity to conduct feature engineering. Since a lot of the data distribution is right skewed, we took the cube root of those variables. We used these transformed variables to train our multi-class logistic model to reduce overfitting, which it did by almost 1% by assessing the AUC scores.

# **Overview of Models**

To predict the outcome of psychosis, we tried four different methods to build prediction models:

1. Logistic Regression
2. Decision Tree
3. Gradient Tree Boosting
4. Neural Networks

## **3.1 Logistic Regression**

**What issues did you encounter in the data?**

The first attempt at running a logistic regression model was trying to understand how to build one for multi-classification and what the difference is between “over-vs-rest” or “over-vs-one.” As I ran the model using the Logistic Regression classifier, I kept getting errors because I was not aware to set the hyperparameter “multi\_class” to “ovr” to indicate how we wanted to train the model. Once this parameter was set, the model was overfitting extremely. Therefore, we realized we would need to conduct feature engineering and dimension reduction.

**How did you tune your models?**

By referencing the histogram of variables, we knew there was a need for variable transformation. We calculated the cube root of the variables that had right skewed data as a starting point to reduce overfitting. However, the model was not performing as well as we had expected, so we looked into how other parameters of the Logistic Regression classifier could help reduce overfitting. By setting the “penalty” parameter to “l1”, to shrink the coefficients, resulted in the model performing somewhat better, although not by much. Eventually, we discovered cross-validation was an option to enhance the model performance by using the Logistic Regression CV classifier, which we set the “cv” parameter to 10. After multiple attempts of tuning our models and not having the bandwidth to try the PCA method, for dimension reduction, the AUC scores before and after tuning and feature engineering is follows:

|  |  |  |
| --- | --- | --- |
|  | **Weighted Average AUC** | **Average AUC** |
| **Train** | 0.9798 | 0.9497 |
| **Validation** | 0.7827 | 0.7490 |
| **After Feature Engineering** | | |
| **Train** | 0.9800 | 0.9511 |
| **Validation** | 0.7891 | 0.7588 |

## **3.2 Decision tree**

For decision tree models, we built two models using grid search cross validation, together with the decision tree classifier function to tune the parameters and get the best performing model with the highest weighted AUC.

First off, we used on-hot encoding to convert all features with string values into numeric ones, ended up with 51 new generated features with dummy values. Adding on the original 35 numeric features, we ended up with 87 features in total.

For the first round of decision tree model building and parameter tuning, we decided to use all the features to get a baseline decision tree model. Then we set the parameters and “scoring” for GridSearchCV:

**Parameters:**



**Scoring:**



One of the challenges we faced was to pick the model evaluation metric that we can use to get the optimal model using each method. Since we were to solve a multiclass problem, we couldn’t use the regular AUC score to evaluate models. After discussing it within the group and consulting the professor, we decided to use weighted AUC as the key metric to get the optimal model, therefore we set the scoring = ‘roc\_auc\_ovr\_weighted’ to get the best performing model as Model 1.

A screenshot of a text message

Description automatically generated with low confidence

When running the cross-validation, we kept getting an error saying the certain class was not present. After certain digging, we realized it was due to that class 5 only has three observations and the cross-validation process couldn’t get at least one observation in each group of the data. Therefore, we had to remove class 5 for this project and hopefully when more data is collected later on, the model can be refreshed.

Graphical user interface, text

Description automatically generated

Text

Description automatically generated

We got Model 1 and got the weighted AUC score from train and validation data.

**Decision Tree Model 1:**

**Parameters:**



**Decision Tree Visualization:**

Diagram

Description automatically generated

**AUC Performance:**

Graphical user interface, text, application

Description automatically generated

Although the Model 1 AUC performance is acceptable, it still requires 87 features to get this level of performance. If we were to deploy this model, we had to improve the model efficiency. Therefore, we rank all the features by feature importance in Model 1 and took all non-zero importance features and included all of them into the building of Model 2.

Model 1 Feature Importance:

Graphical user interface, text

Description automatically generated

Non-Zero Importance Features:

Text, letter

Description automatically generated

For Model 2, we used the same set of parameters and potential values and setting for GridSearchCV and got a new optimal model as Decision Tree Model 2.

**Decision Tree Model 2**

**Parameters:**

A picture containing chart

Description automatically generated

**Decision Tree Visualization:**

Diagram

Description automatically generated

**AUC Performance:**

Graphical user interface, text, application

Description automatically generated

Turns out Decision Tree Model 2 has yielded the same level of weighted AUC for training data. However, for validation data, Model 2 showed slight improvement. Meanwhile, Model 2 only requires 5 numeric features. Based on model performance and efficiency, we decided to use Model 2 as the optimal model for the Decision Tree method.

Table

Description automatically generated

## **3.3 Gradient Tree Boosting**

We also trained a gradient boosting tree model as a candidate classifier since our dataset is highly dimensional, heavily skewed and only has a few observations. We expected gradient boosting can decrease the potential high variance and negative impact from unbalanced data.

The stratified shuffle split was used in training to prevent sampling bias, making sure that validation dataset contains observations from all classes. Thus, the evaluation score can be more accurate. After testing several features combinations, we decided to use predictors having an importance score higher than .015 for training and used AUC weighted score as randomized search evaluation metric. F1 weighted score was also tested during the process, but the result was not as good as AUC.

Based on the AUC score for each class and overall data, we can see the gradient tree boosting model performs pretty well on class 0, which is that patients‘ symptoms didn’t get remission. When predicting class 1 and class 2, the model has relatively the same performance. And the model performs worst at predicting class 3 and class 6. The reason is that we don’t have enough observations from these two classes, which also makes the model very unstable when predicting them.  
Diagram

Description automatically generated with low confidence

One difficulty in training gradient tree boosting is that randomized search cv cannot always bring the best parameters for the model. Because, for example, we only have 7 observations for class 6 in the training dataset, so during the cross validation, the model may calculate the best evaluation metric from a subset of data without class 6. So the parameters selected by these kinds of metrics might not result in the best performance of the model in terms of the real data or validation data. In addition, the balance between overfitting and under-fitting is difficult to control.

## **3.4 Neural Networks – Perceptron**

**What issues did you encounter in the data?**

The data first had to be converted into a TensorFlow dataframe rather than a pandas dataframe in order to work with Keras. Additionally, the small number of observations likely affected the performance of the neural network adversely, as neural networks tend to train and make predictions better when given a larger dataset to work with.

**How did you tune your models?**

Model tuning was done through research into neural networks built with the keras package. The efficacy of different activation and optimizer functions for multiclass classification problems was researched. Due to difficulties in creating a search grid, the results from several different types of the model were compared, with the best results selected. An L2 regularization function, along with a learning rate of 0.01, reLu activation function for the input nodes and hidden layers, and softmax activation function for the output were selected. Additionally, sparse categorical accuracy was selected as the performance metric, as this selection allows the model to work with multiclass classification data.

# **Model Evaluation**

Because it is multi-class classification and the dataset is unbalanced, we focused on the weighted AUC score to assess models. We can see that every model suffers an overfitting issue in terms of the overall performance. Logistic regression has the most severe overfitting while regular tree has the mildest overfitting issue. Both tree based models have better performance than logistic regression model. Gradient tree boosting performs slightly better than regular decision tree.

Chart, bar chart

Description automatically generated

For each class, every model shows a very high AUC score when predicting class 0, which is around 0.95.  They also perform good at predicting class 6. Gradient tree boosting performs best for class 1 and class 2. Regular decision tree beats other models in terms of class 3. Logistic regression has the largest variance of AUC scores, which means its prediction result is very unstable.

Chart, bar chart

Description automatically generated

We also look at accuracy for each model as the perceptron model doesn’t have AUC for comparison. As we can see the perceptron model has the highest accuracy on the validation dataset, followed by regular decision tree model. The overfitting issues of perceptron and regular decision tree are not as severe as logistic regression model and gradient boosting tree.

A picture containing chart

Description automatically generated

Based on the AUC and accuracy metrics, we believe the gradient boosting tree is the best classifier to predict whether a patient's symptoms are remission within a specific time period.

# **Model Implementation**

**How would you monitor its performance if deployed?**

One way to monitor the performance of the model is to cross - reference the prediction and the actual outcome of the patient. For example, if the prediction for a particular patient were they will be in remission for at least 1 year and they actually stayed in remission for at least one year - this would be a success story.

**If given the chance, what changes would you need to make in order to implement your model into production?**

It would be interesting to have the opportunity to identify how to implement a Positive and Negative Syndrome Scale (PANSS) tool in electronic health records (EHRs) that can predict patients’ mental health outcomes. Health care providers would be able to assess their patients by using the PANSS tool and input their observations into the tool and predict the outcome of the patients’ mental health.

# **Postmortem**

The whole process of finding the data, identifying the problem, building different models and evaluating model performance was a great practice for everyone team member and we did learn so much from a practical standpoint. However, if we had more time, we would love to improve on these steps:

* Try different approach on the data cleaning
  + After feature engineering, where we took the cube root of variables with right skewed data, to enhance our logistic regression model. However, this resulted in a lower performing AUC compared to our initial model prior to any featuring engineering. Since our initial dataset has many dimensions with fewer observations, if the team had more time, we would have the opportunity to reduce dimensionality by performing PCA and possibly achieving a better performing model.
  + With more time, the team could have identified which features were most important to each model and use those features to run the model again - increasing the possibility of enhancing the models’ performance.
* We did try four different modeling methods, but if we had more time we would love to try more methods like Naive Bayes model, support vector machine etc.
* Even though we got a model with great performance on predicting all labels, but if we had more time, we could spend more time on hyperparameter tuning to improve the performance on class 3 and overall performance as well.
* We had a great idea of building a small recommender system, with at least one pre-defined recommendation per predicted relapse or remission outcome. Would love to test it out if we had more time.