December 8, 2019

**50.038 Computational Data Science**
Project Report

Professors

*Prof. Dorien Herremans*
*Prof. Soujanya Poria*

# Analysis of Positive and Negative Syndrome Scale (PANSS) Assessments of Schizophrenic Patients

***Prediction of Erroneous PANSS Assessments***

Authors
*Singapore University of Technology and Design*

*Lu Jiankun, 1002959*
*Nashita Abd Guntaguli, 1003045*
*Peng Shanshan, 1002974*

Abstract

The objective of this paper is to explore various data mining tools while finding the best method to predict which of the PANSS assessments will be erroneous (either flagged for review or assigned to a CS). We consider five different traditional classifiers: logistic regression, naïve bayes, support vector machines k-nearest neighbor and random forest. Following this, we implemented 7 different neural network architectures: simple and bidirectional RNN, bidirectional LSTM, bidirectional GRU, CNN-biLSTM, CNN-biGRU and transformer based architecture. Based on our experiments, RNN architectures perform the best.

## Introduction

Positive and Negative Syndrome Scale(PANSS) Assessment is an assessment that aims to measure symptom severity of patients with schizophrenia. For instance, these symptoms include depression, anxiety, tension etc. Based on the increasing severity of positive, negative and general symptoms, patients are scored on a standardized scale from 1-7 for each symptom. [1] These seven points representing increasing levels of psychopathology are as follows:

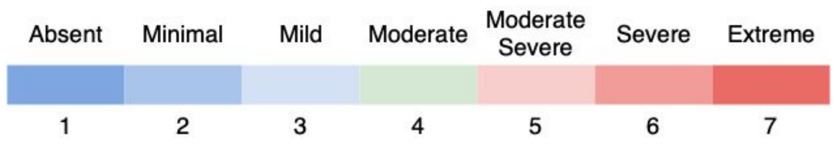


Fig. 1. Symptom severity scale in relation to PANSS Score

Interviewers are trained to standardized reliability, allowing them to measure the PANSS score during a brief 50 minute interview. The patient is rated a score on 30 different symptoms based on the interview as well as reports from family members or primary care hospital workers.

The PANSS has three classes of items that are assessed:

1. Positive symptoms (7 items) which refer to an excess or distortion of normal functions (P1-P7)
2. Negative symptoms (7 items) which represent a diminution or loss of normal functions (N1-N7)
3. General Psychopathology symptoms (16 items). (G1-G16)

Each PANSS item is rated on an ordinal scale from 1 (i.e. absent) to 7 (i.e. extreme), hence the minimum score a patient can receive is 30 and maximum is 210.

Following is a list of all the 30 symptoms being measures in a PANSS assessment:

Table 1. PANSS Symptoms measured on a scale of 1-7 in PANSS Assessments

| **Positive Symptoms** | **Negative Symptoms** | **General Psychopathology Symptoms** |
|---|---|---|
| P1: Delusions | N1: Blunted affect | G1: Somatic concern |
| P2: Conceptual disorganisation | N2: Emotional withdrawal | G2: Anxiety |
| P3: Hallucinations | N3: Poor rapport | G3: Guilt feelings |
| P4: Excitement | N4: Passive/apathetic social withdrawal | G4: Tension |
| P5: Grandiosity | N5: Difficulty in abstract thinking | G5: Mannerisms & posturing |
| P6: Suspiciousness/persecution | N6: Lack of spontaneity & flow of conversation | G6: Depression |
| P7: Hostility | N7: Stereotyped thinking | G7: Motor retardation |
| | | G8: Uncooperativeness |
| | | G9: Unusual thought content |
| | | G10: Disorientation |
| | | G11: Poor attention |
| | | G12: Lack of judgement & insight |
| | | G13: Disturbance of volition |
| | | G14: Poor impulse control |
| | | G15: Preoccupation |
| | | G16: Active social avoidance |

PANSS is called the "golden standard", indicating that all assessments of psychotic behavioral disorders must follow it. It provides balanced representation of positive and negative symptoms and gauges their relationship to one another and to global psychopathology.

**Total G score**: sum of all 16 general psychopathology symptom scores (G1-G16)
**Total N score**: sum of all 7 negative symptom scores (N1-N7)
**Total P score**: sum of all 7 positive symptom scores (P1-P7)

### Problem Statement and Analysis Question

Experts ingrain their full efforts to maintain high reliability, however, there are often noticeable occurrences of erroneous ratings. Examples include the patient assessment (as a whole) not making any sense, assessments that are inconsistent with previous ratings, and an outcome assessment trajectory that is infeasible. Therefore, clinical auditing firms are typically hired to validate the collected patient assessments. Assessments that are potentially erroneous are either flagged for review or assigned to a clinical specialist for follow up and confirmation.

Our goal in this data science project is to analyse the given data (PANSS assessments of patients) and audit those assessments that are potentially erroneous. Hence, we will be using data analysis and machine learning techniques to answer the question:

Is this particular assessment valid or is it erroneous (needs review)?

Having humans audit all of the PANSS assessments can be time consuming and expensive. It is therefore reasonable to wonder if a machine learning algorithm can reliably predict which of the assessments are erroneous. Having a reliable algorithm would allow the clinical auditor to focus only on the subset of assessments that have issues, instead of all of the assessments uniformly. Hence, digitalising the task of auditing assessments will be an effective and efficient addition to the current PANSS process in the field of psychology.

**Binary Classification Problem**
**Positive Class**: Potentially Erroneous Assessments
**Negative Class**: Passed Assessments

## Dataset

### Dataset Source

We sourced for our dataset for this project from the **STATS 202 course** at Stanford University.
Here is the link:
https://sites.google.com/site/stats202/class-project-2012

### Data Description

The dataset contains 20,947 rows and 40 feature columns.

1. *Study* - A character indicating which of the five studies the data represents.
2. *Country* - The country where the assessment was conducted.

3. *PatientID* - An identification number given to each unique patient.
4. *SiteID* - An identification number given to each unique assessment site.
5. *RaterID* - An identification number given to each unique rater.
6. *AssessmentiD* - An identification number given to each unique assessment conducted.
7. *TxGroup* - A string corresponding to the patient's (randomly) assigned treatment group.
8. *VisitDay* - An integer corresponding to the number of days that have passed since the baseline assessment.
9. *P1-P7* - The scores corresponding to each of the 7 positive symptoms of the assessment.
10. *N1-N7* - The scores corresponding to each of the 7 negative symptoms of the assessment.
11. *G1-G16* - The scores corresponding to each of the 16 general psychopathology symptoms of the assessment.
12. *PANSS_Total* - The sum of the ratings across the 30 PANSS items.
13. *LeadStatus* - A string indicating whether the assessment's audit passed, was flagged, or was assigned to a clinical specialist.

Our **label** will be **LeadStatus**, with the following possible values or classes:

1: Potentially Erroneous Assessments (flagged for review or assigned to clinical specialist)

OR

0: Passed Assessments

## Exploratory Analysis

To understand our data better, we conducted data cleaning to check null or error values and skewness of the data, followed by some exploratory analysis including looking at distribution of data columns and correlation among columns. We also observed some interesting trends within the dataset which influenced our decision on which models to use.

### Distribution of Data

The following are some examples of visualising the distributions of certain PANSS scores over all assessments for our 2 different target labels.

#### a. Distribution of PANSS assessments by Positive, Negative and General Score Categories

The 30 features (scores) can be grouped into 3 categories; Negative symptoms, Positive symptoms and General psychopathology symptoms. The following plots (Fig. 2) display standardized feature scores of positive, negative and general symptoms groups.

*Fig. 2 description:*

1. Scatter Plots (Top Right): Linear Regression of data sample and Linear Dependencies between
   a. positive and negative (correlation coefficient $r = 0.45$)
   b. negative and general (correlation coefficient $r = 0.63$)
   c. positive and general (correlation coefficient $r = 0.78$)
2. Density Plots (Bottom Left): Density relations between the 3 symptom groups. Features were standardized for comparability.
3. Diagonal: Individual distributions of positive, negative and general symptom scores (features).

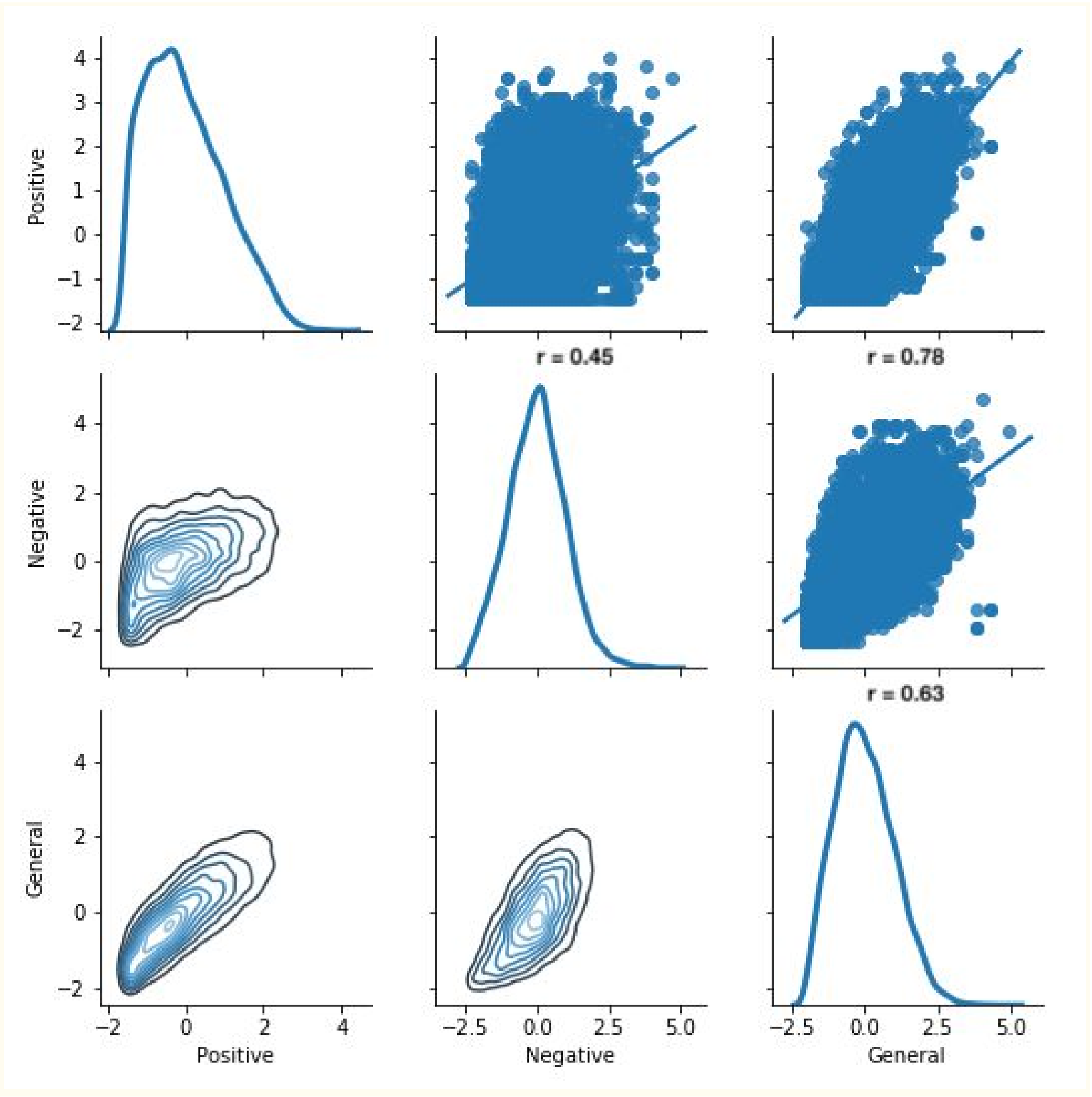


Fig. 2.

### b. Imbalance of Dataset in terms of Target Feature (Labels)

We noticed that our dataset is highly imbalanced in terms of target variable distribution. Target class 1 is only 24.4% and target class 0 is 75.6%. This means that a model that predicts only target class zero will achieve a reasonably high accuracy of 75% (Fig. 3). Therefore, this insight allowed us to tackle the problem of an imbalanced dataset using class weight and up-sampling, which will be discussed further in this report.

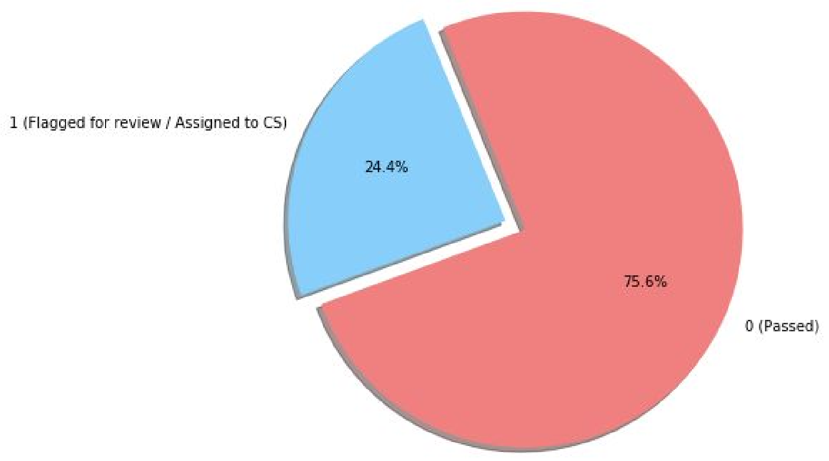
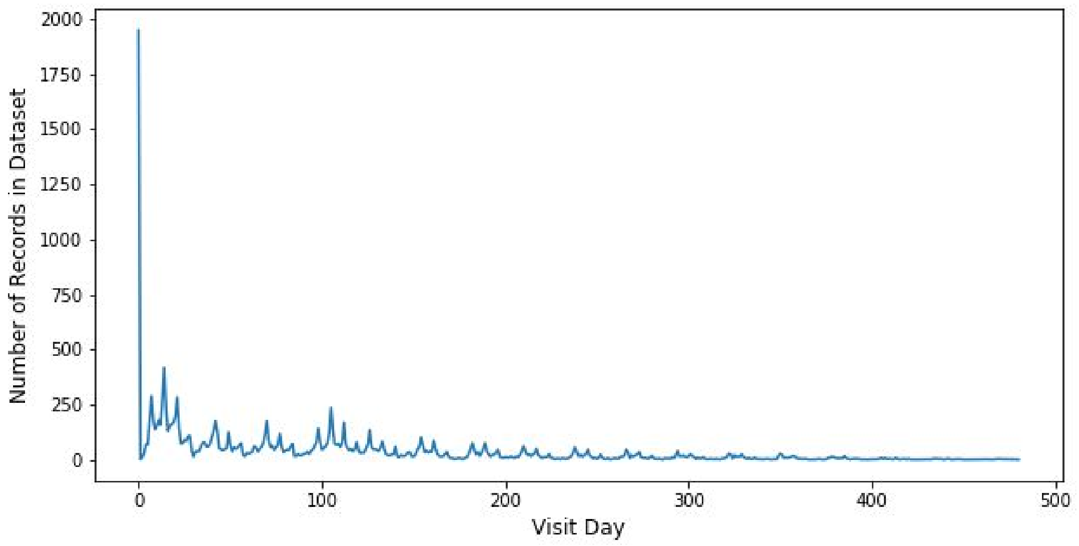


Fig. 3. Pie chart shows our imbalanced dataset distribution

### c. Understanding *VisitDay* feature and Binned Visit Days

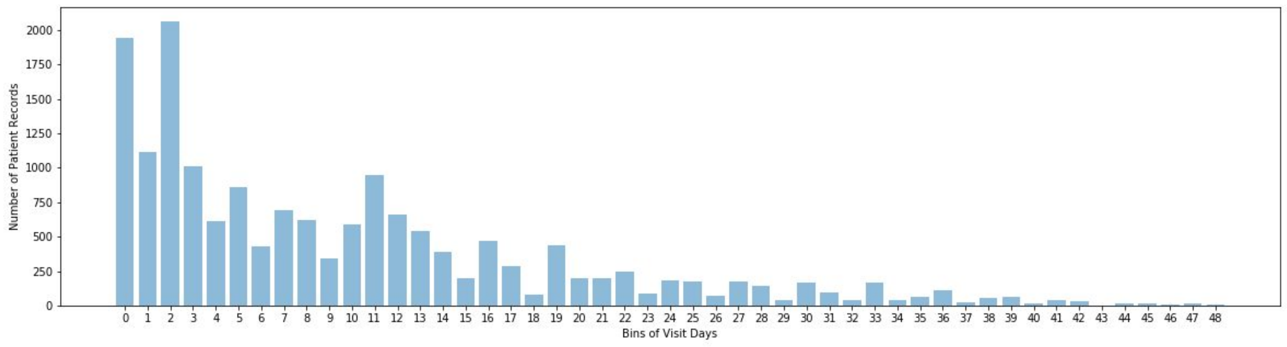
*VisitDay* is an integer corresponding to the number of days that have passed since the baseline assessment. The following graph (Fig. 4) helps us understand the distribution of *VisitDay* feature as it displays the number of patient records available for any given visit day. Since this Data is too discrete, we decided to bin the *VisitDay* feature.

Fig. 4. Distribution of *VisitDay* feature



The following bar graph (Fig. 5.) shows the binned visit days, where visit days have been binned on 10 day intervals.

Fig. 5. Number of Patient Records for each *VisitDay* Bin



## d. Failure Rate over Country and PANSS Scores

It is also interesting to see rate of failure across countries and PANSS scores. As we can see, in our dataset (Fig. 6, Fig. 7), China has the largest population, followed by the USA and Russia, while China, India and the USA have the highest failure rates for PANSS assessment. There is no significant correlation observed between average PANSS total scores and rate of failure.

Fig. 6. Probability of Target Variable being 1 for each Country

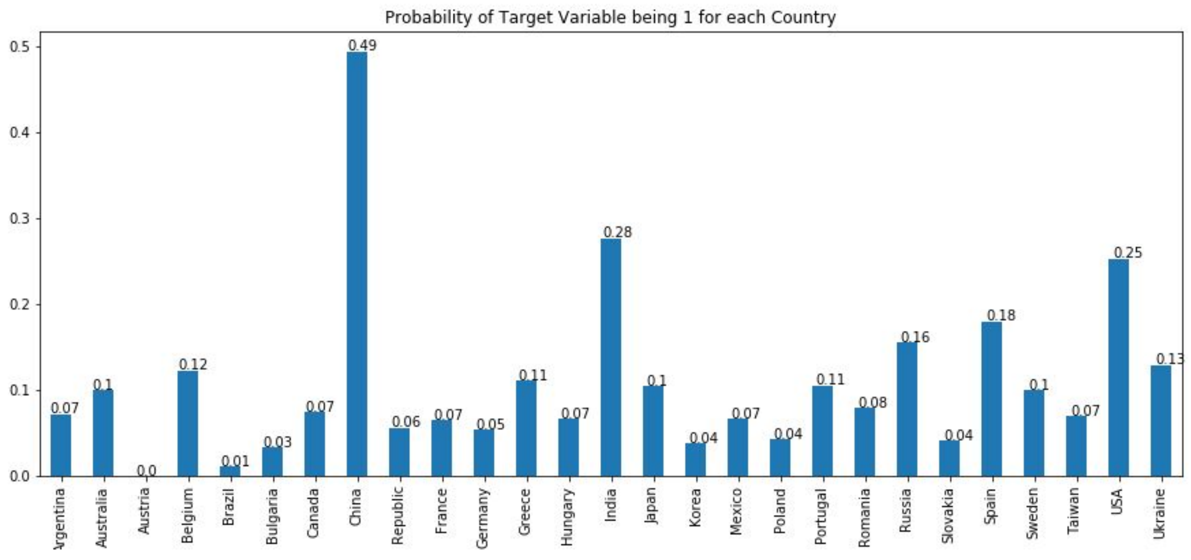
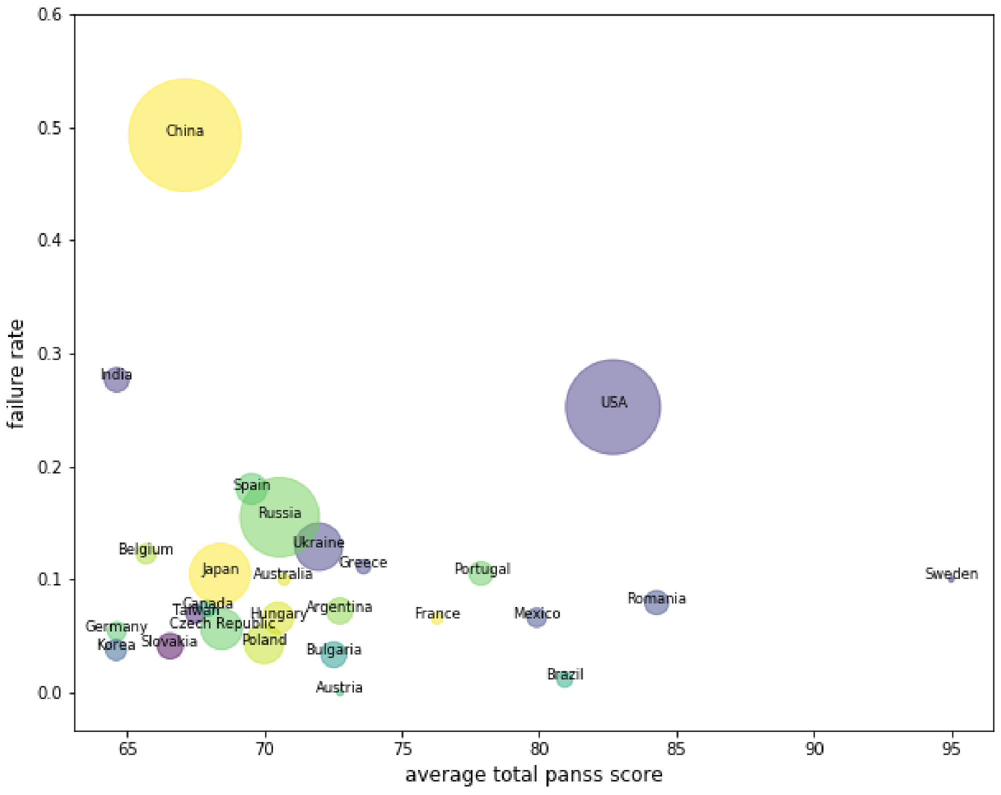


Fig. 7. Visualization of Failure Rate in each country

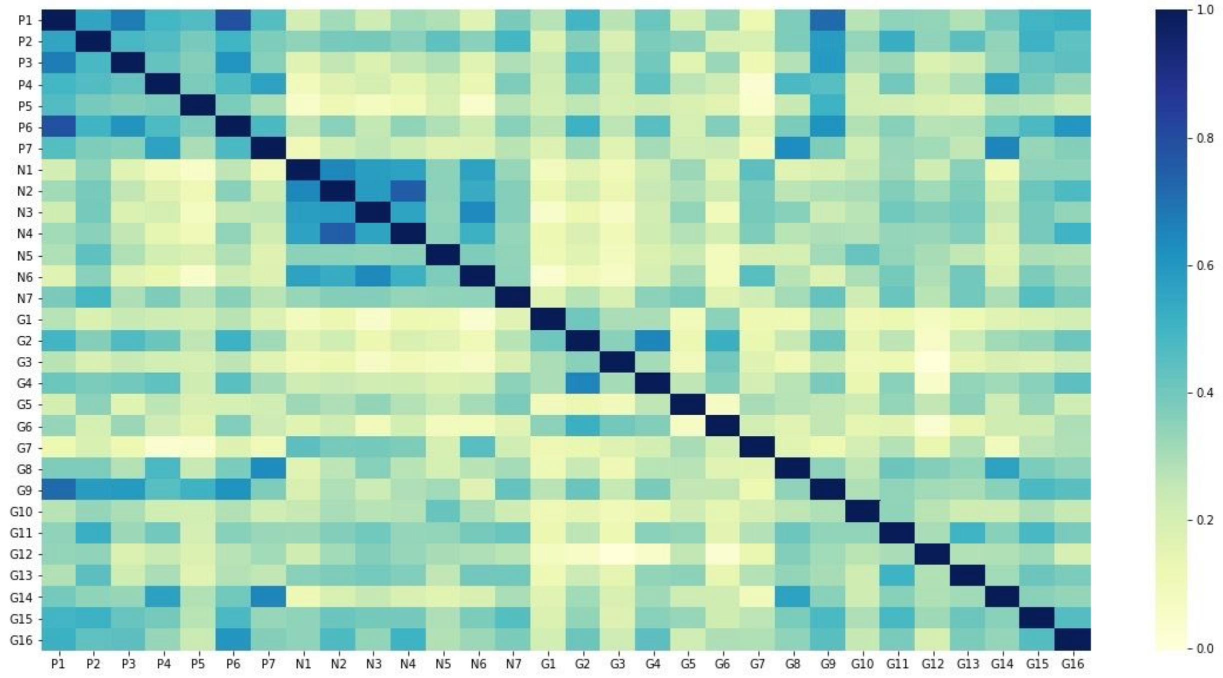


These 2 plots indicate a higher probability of assessments being flagged for review or assigned to CS for certain countries like USA, India and China. This reason for this result could however be the very large population of USA, India and China. Hence, this stipulates that including these features in our dataset will introduce a bias. Hence, we decided to drop this column of *Country*.

## Correlation Among PANSS Scores

We also visualized correlation among the 30 PANSS scores, so that we could drop highly correlated ones in our classification task , as more independency among features usually gives better performance in machine learning models. However, this heatmap (Fig. 8.) indicates that there is no significant correlation among the PANSS scores.

Fig. 8. Heatmap of correlation amongst 30 PANSS symptom scores



## Principal Component Analysis

Next, in order to understand the hidden item stratification, we applied PCA. In previous studies, PCA was the most applied data analysis model to find out hidden factors of variation involving PANSS assessments. Previous studies used five-factor models for evaluation.[2, 3, 4] Therefore, we also evaluated our PCA to extract 5 model components.

PCA helped us comprehend the PANSS items scores by combining them into 5 representative variables. Based on the amount that patient scores vary with each dimension, these five principal components collectively explained the variability across the set of assessments and features. The following heatmap (Fig. 9) summarises the 5 principal components we obtained.

Fig. 9. HeatMap of correlation between 5 PCAs and 30 PANSS features

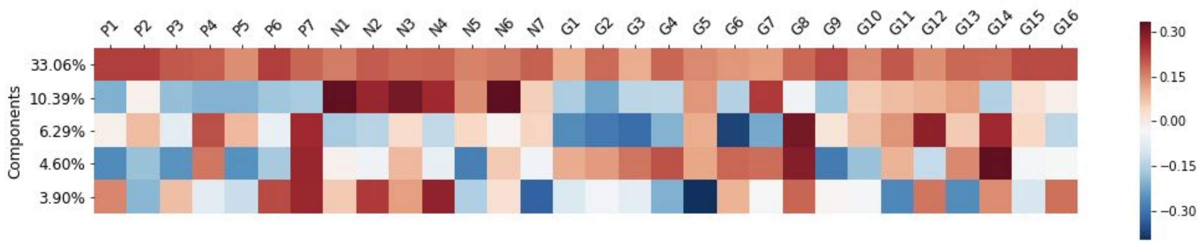


Fig. 9. The first principal component captures most of the information of patient profiles and their variation, hence it contributes most to variation in the dataset, indicating a close correlation between all the original features and itself. It provides a linear fit closest to a patient's assessment profile, as indicated by the heatmap. The following 4 components are uncorrelated to each other. Each component and its ratio of explained variance is shown. The result is each component roughly corresponding to each group of symptoms.

Our results were aligned with previous research results, and hence it indicated that our data is not biased in terms of distribution or variance. Most of previous research reported five-component PCA solutions for PANSS assessments.

This PCA decomposition indicated 58.24% of variance across PANSS scores. This result is comparable to previous studies results: 52% [3], 51% [4], up to 57% of variance explained [2].

1. The component that captured most variance (33%) was most associated with certain positive and negative symptoms, and is consequently a representative of the average patient profile. The first principal component from our dataset was consistent with the first component in other studies [2-4], indicating the highest correlation with certain symptoms like delusions, preoccupation, emotional withdrawal and disorganization.
2. The second principal component from our dataset displayed a 10% variance with highest correlation between negative symptoms like blunted affect, poor rapport and poor spontaneity. It also shows a certain correlation with general symptoms like poor attention and lack of judgement and insight. This was in contrast to the second component in the previous study where the component correlates most with positive symptoms rather than negative symptoms.
3. Our third principal component contributes to 6% variance and is comprised most with general cognitive symptoms like poor attention, mannerisms and posturing, poor impulse control and hostility. This is aligned with the third cognitive component results of previous studies.
4. The fourth component represents a variance of 4.6%and correlates with symptoms like excitement, hostility and uncooperativeness. In previously reported components, the fourth component also relates most with excitement related features
5. And finally, our fifth component is a representative of depression symptoms encompassing anxiety, guilt, emotional withdrawal and tension with 3.9% variance. This is also consistent with previous results.

In summary, only our second component correlates more with negative instead of positive symptoms as compared to previous study results. Hence the 5 PCAs extracted from our patient sample are virtually identical to the components found in a series of previous hidden factor decompositions in schizophrenic patient samples (2-4). This indicates a reasonable distribution of our patient samples.

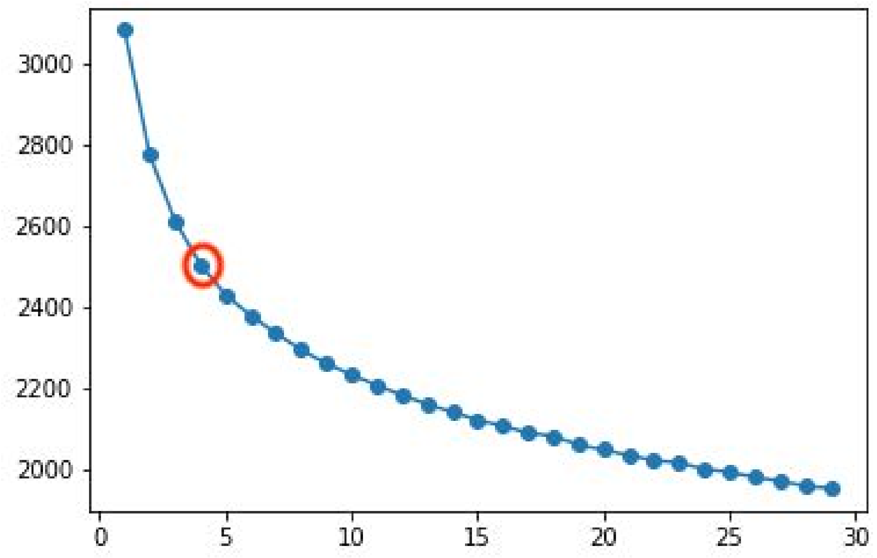
## K-means

To understand the hidden group structure of the patients in the data, we applied the k-means clustering algorithm to partition the patients based on similar symptom profiles. Our goal for using K-means is to divide the patients into similar salient symptom profiles and understand relationships between varying patients in different groups. K-means will helps us determine symptom patterns and relationships among schizophrenia patients.

In order to capture the functional status of patients prior to experiencing any treatment or interview effects, we used the baseline measurements of each patient(i.e. their 0th visit day scores) when segmenting patients into maximally different groups.

To decide the best value of k, we performed clustering using up to 30 clusters and used the Elbow Method to indicate the right number of clusters that best fits the data. The best number of clusters is 4. (Fig. 10)

Fig. 10. Total error vs k value



**Result:** Four clusters that were determined exposed four distinct symptom clusters of patients. Bar graph (Fig. 11) shows a comprehensive outlook on the four groups and their symptom severity distribution.

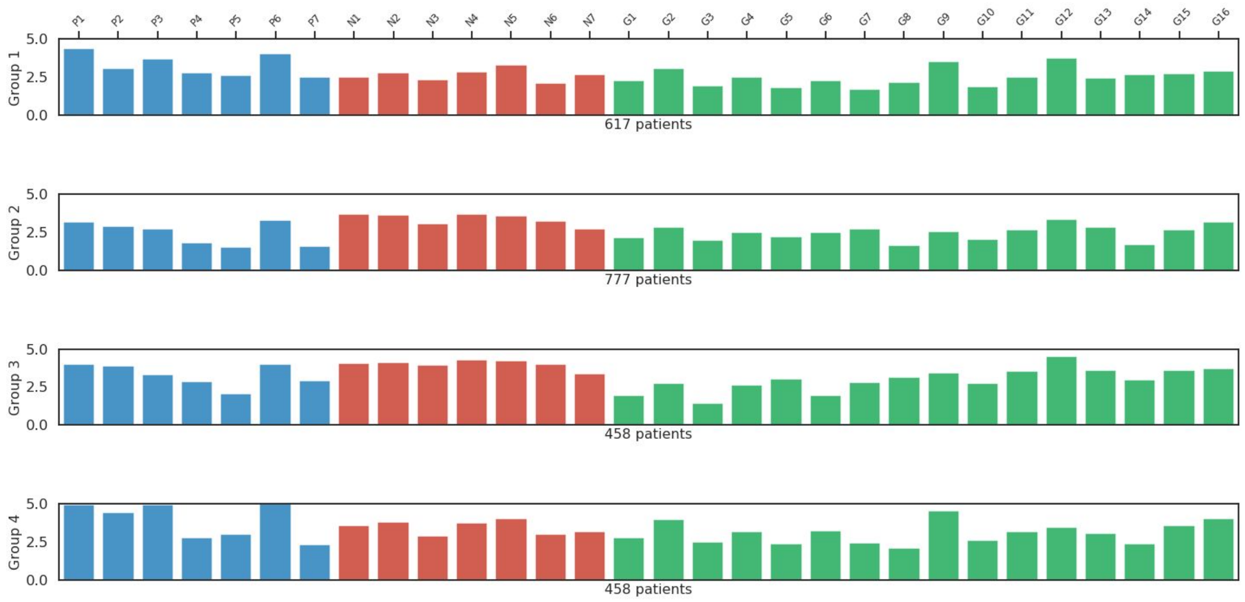


Fig. 11.

*Fig 11 description:*

The importance (score) of each PANSS item determines the weight of each bar.
Blue bars: Positive symptom scores
Red bars: Negative symptom scores
Green bars: General psychopathology symptom scores

Consequently, graphs (Fig. 12) show a correlation between either total Positive, total Negative and total General psychopathology scores of each group.

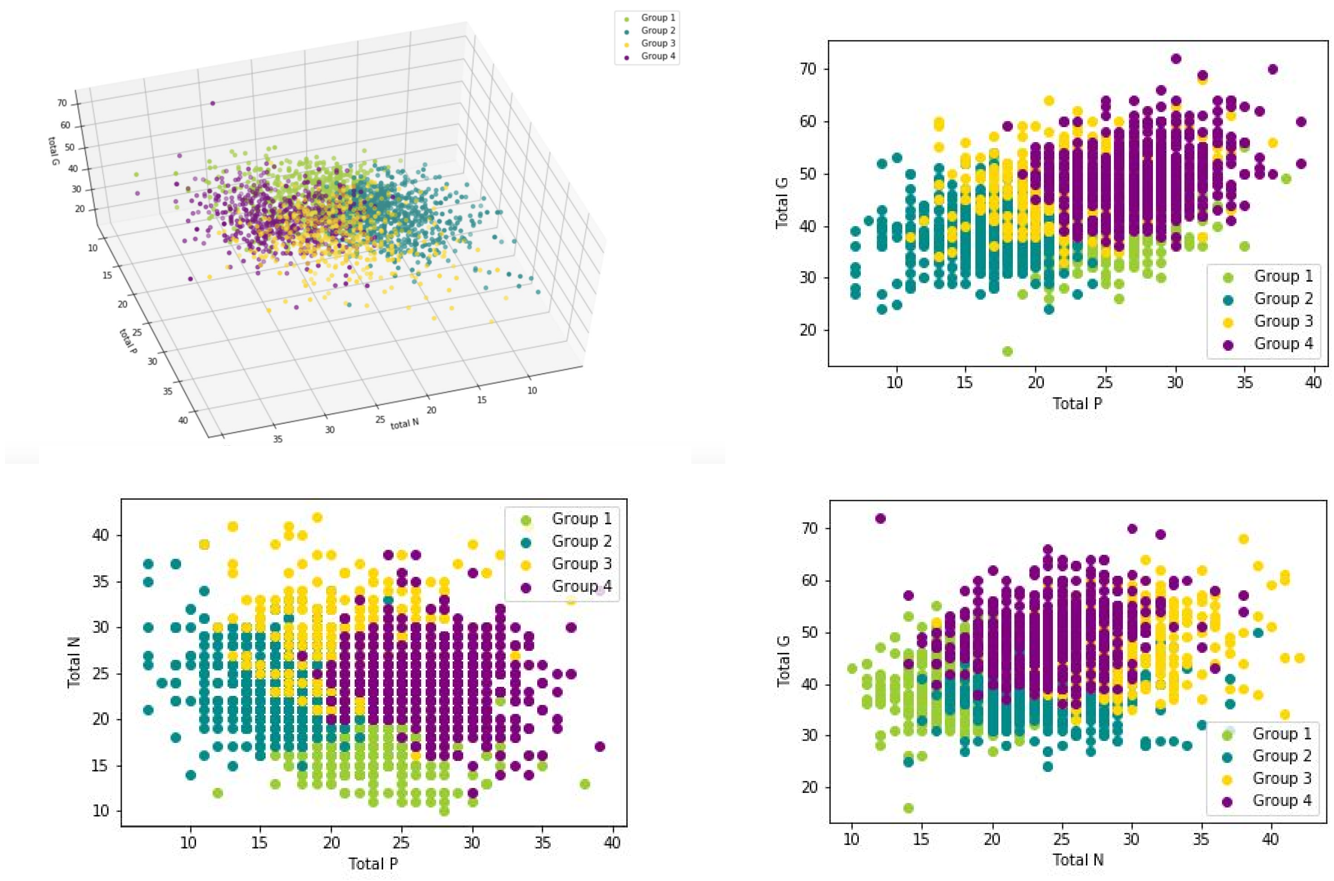


Fig. 12.

*Fig 12 description:*

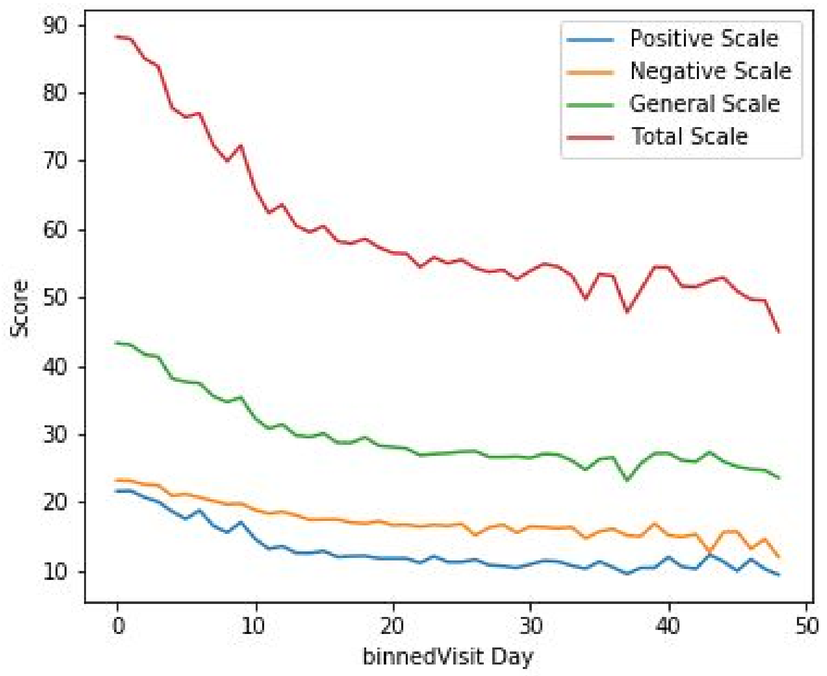
Significant patterns are seen in the structured relationship between varying patient groups:
Group 1: relatively low total G and total N scores, average total P score
Group 2: relatively low total G and total P score, average total N score
Group 3: high total N score, average total G score and total P score
Group 4: high total P and total G, average total N score

Therefore, to summarise our results, we discovered a hidden structure within our dataset not only in terms of dimension (using PCA) but also a discriminative hidden structure in the PANSS scores based on the categorical aspect of PANSS.

### PANSS Score Trend over Time

Looking at past assessments of a patient and determining if the new assessment record is consistent with past records is one of the important criteria to evaluate the validity of an assessment. Therefore, we further investigated our dataset based on time series (i.e. binnedvisitDay) of patients and discovered a generally decreasing trend of PANSS scores over different visit times, conforming with the intuition that the patients' symptoms get better and better with effective treatment. It also inspires us to consider one patient as a data point with time series instead of using individual assessment records that does not contain any temporal information, which were used in our LSTM models as input.

Fig. 13.



## Data Preprocessing & Feature Engineering

### Binary Label

We will be performing a Binary Classification task on the dataset, hence we first converted the label column, i.e. *LeadStatus* column as follows:

a. Replaced "Flagged" and "Assigned to CS" with 1
b. Replaced "Passed" with 0

### Train Test Split

In order to ensure the same distribution in both train and test sets, we merged and shuffled the 4 datasets. The train-test was split as 80/20. Furthermore, we made sure all the patient records concerning one patient were either in train or in test set, not in both.

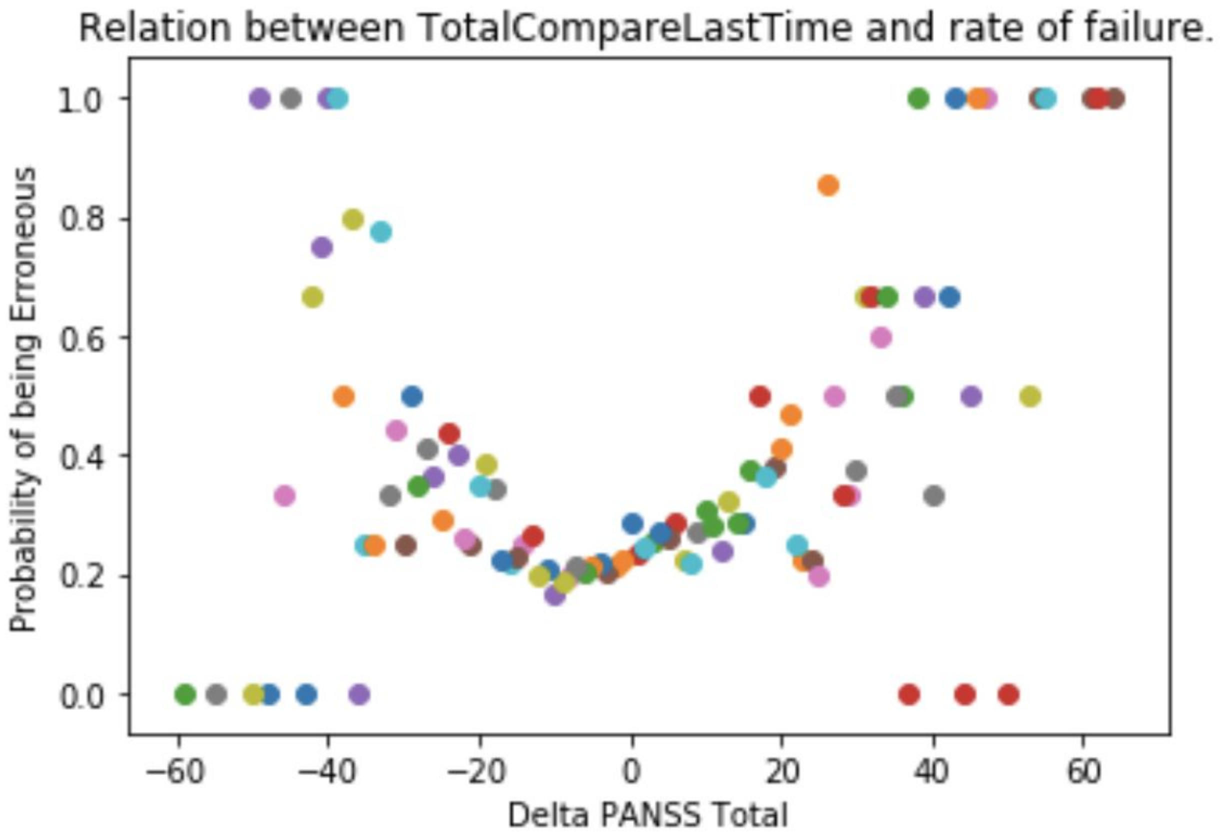
## Upsampling

One issue of our dataset is that it is heavily biased, with more than three quarters having 0 labels. Trained on an imbalanced dataset, a classifier can produce suboptimal models which are biased towards the majority class and have low performance on the minority class. To solve the problem, we implemented up-sampling for class 1 and trained the models using the resulted new balanced dataset.

## New feature that encompasses time series

Inspired by the exploratory where we found the trend of PANSS scores over time, we constructed a new feature PANSS_Score_Comp_Last, by subtracting PANSS_Total by that of the same patient's last assessment record. Through this feature, we are able to extract the change of PANSS scores over time for each patient, though only between two consecutive visits in this case, which can be further modified in the future to represent a longer time span. We observe (Fig. 14) a roughly convex relation between this new feature and the assessment failure (having a 1 label) rate, indicating that bigger changes tend to be erroneous.

Fig. 14.



# Models and Evaluation

## Evaluation Methodology

As mentioned in Data Preprocessing, 20% of our dataset is split as test set. While splitting, we ensured that all records related to each patient are part of either train or test set.

In the training process we also used 4 fold cross validation.

We want to find erroneous ones, but we cant predict too many erroneous because that defeats the purpose

We used F1 score as our evaluation metrics, as it measures both the recall and precision, which are important indicators for our model's reliability and efficiency. Our model needs to avoid Type 2 errors as predicting "erroneous" assessments as "non-erroneous" will be dangerous to the patients and psychiatrists. In order to not overlook any erroneous assessments a high recall is required. At the same time, we cannot predict too many "erroneous" assessments as well, as that would defeat the purpose of our data science task. Hence, a high precision is needed as well to ensure minimized false positive (Type 1 error), so that human labour for auditing is saved at an optimum level.

Therefore, we chose F1 score as our evaluation metric as it determines a balance between both precision and recall. Furthermore, since our data set is imbalanced, F1 score is a better measure of our model's performance.

## 1. Traditional ML Models

We first try a few traditional machine learning models. In particular, we tested five models to determine which one had the best fit. Table 2 gives the results for each of the models after determining the best fit for that model.

Table 2. Overall Results of Traditional Models

| Model | Precision | Recall | F1 | Accuracy |
|---|---|---|---|---|
| Logistic Regression | 0.35 | **0.67** | 0.45 | 0.63 |
| Naïve Bayes | 0.35 | 0.54 | 0.43 | 0.64 |
| Support Vector Machine | 0.44 | 0.61 | **0.51** | 0.71 |
| Random Forest | **0.55** | 0.38 | 0.45 | **0.77** |
| KNN | 0.37 | 0.56 | 0.45 | 0.66 |

## a. Logistic Regression

Logistic regression is a relatively simple and quite effective algorithm to estimate the probability of a binary response based on one or more predictor independent variables. Logistic Regression simply models probability of output in terms of input and does not perform statistical classification. It can be used to make a classifier by choosing a cut-off value and classifying inputs with probability greater than the cut-off as one class, below the cutoff as the other.

In this project, we used linear_model.LogisticRegression() from sklearn library. As we can see from the results, our model is affected seriously by the imbalanced dataset and the predictions on test set are heavily biased towards label 0, resulting in a poor F1 score of 0.17 for label 1.

To solve the problem of imbalanced dataset, we assign class weights to our model:

0.75 for class 1 and

0.25 for class 0.

With class weights, the F1 score boosts up from 0.17 to 0.44.

In order to get better results, we conducted a random search followed by a grid search in the hyper parameter space and optimized the hyper parameters including penalty choice (L1 or L2) and C value (the regularization strength). After the optimization, we were able to increase the F1 score by 0.01. Table 3 displays the results of this model.

Table 3. Results of Logistics Regression Model

| Model | Precision | Recall | F1 | Accuracy |
|---|---|---|---|---|
| **Default** | **0.62** | 0.10 | 0.17 | **0.76** |
| **With class weights** | 0.33a. | **0.67** | 0.44 | 0.59 |
| **After fine tuning** | 0.35 | **0.67** | **0.45** | 0.63 |

## b. Naïve Bayes

The Naive Bayesian classifier is based on Bayes' theorem with independence assumptions between predictors. A Naive Bayesian model is easy to build, with no complicated iterative parameter estimation. We used naive_bayes.GaussianNB() from sklearn library in our implementation.

Because of the independence assumption, we performed PCA on our data to convert the slightly correlated data to a set of linearly uncorrelated components before feeding it into the model. However, no significant performance improvement is observed after applying PCA, which conforms with our correlation analysis result, that there is no highly correlated features within our dataset.

To deal with imbalanced data, we upsampled the minority class. With upsampling, the model's F1 score increases from 0.40 to 0.43. The following Table 4 shows the result of fine tuning for the model.

Table 4. Results of Naive Bayes Model

| Model | Precision | Recall | F1 | Accuracy |
|---|---|---|---|---|
| **Default** | 0.40 | 0.41 | 0.40 | 0.70 |
| **After PCA** | **0.44** | 0.20 | 0.28 | **0.74** |
| **With upsampling** | 0.35 | **0.54** | **0.43** | 0.64 |

## c. Support Vector Machine

Support Vector Machines is another very popular classification technique. It can be considered a perceptron at optimal stability, which finds the optimal decision boundary by maximizing the margin (the distance between support vectors and the decision boundary). It is relatively robust to outliers as the decision boundary is affected only by the support vectors. We used svm.SVC() from sklearn library. Same issue of imbalanced dataset affects SVM models as well and we applied class weights during training process, improving the F1 score from 0.00 to 0.44. After that we tried different kernels to obtain a more complicated decision boundary instead of a linear boundary, and RBF kernel gave us the best performance. Table 5 summarizes the results of this model.

Table 3. Results of SVM

| Model | Precision | Recall | F1 | Accuracy |
|---|---|---|---|---|
| **Linear Kernel without class weights** | 0.00 | 0.00 | 0.00 | **0.75** |
| **Linear Kernel with class weights** | 0.36 | 0.56 | 0.44 | 0.65 |
| **RBF kernel with class weights** | **0.44** | **0.61** | **0.51** | 0.71 |

## d. Random Forest

Random Forest classifier is a very useful ensemble method based on Decision Tree classifier. It randomly chooses a subset of the available attributes and runs a decision tree classifier. It has many advantages which makes it a good fit for our dataset. For example, it is robust to outliers and non-linear data. It also handles unbalanced data well because it tries to minimize the overall error rate. When trained on an unbalanced data set, the larger class will get a lower error rate while the smaller class will have a larger error rate. In this project, we used ensemble.RandomForestClassifier() from scikit learn library.

We did up-sampling during the training process to rectify the issue of imbalanced data, however it did not improve the model's performance by a lot. In order to get better results, we conducted a random search followed by a grid search in the hyper parameter space and optimized the hyper parameters including maximum depth of the tree, the number of features to consider during best split, the minimum number of samples required to split an internal node and the number of trees in the forest. Table 6 shows how after the optimization, we were able to increase F1 score to 0.45.

Table 6. Results of Random Forest

| Model | Precision | Recall | F1 | Accuracy |
|---|---|---|---|---|
| **Default** | 0.55 | 0.20 | 0.30 | 0.76 |
| **With upsampling** | 0.49 | 0.23 | 0.31 | 0.75 |
| **After fine tuning** | **0.55** | **0.38** | **0.45** | **0.77** |

## e. KNN

K-nearest neighbors is another useful classifier. It counts the number of cases where relevance is 1 and the cases where relevance is 0 at a given point, and then assigns the value with a higher frequency to that point. In this project, we used sklearn.neighbors.KNeighborsClassifier() from scikit learn library.

The key issue is to define distance and scale variables. If the features have different dimension, the result of K-Nearest Neighbors classifier could be inaccurate. Therefore, all the data was standardized when preprocessing the data to improve the results. We did up-sampling during the training process to rectify the issue of imbalanced data, improving the F1 score from 0.30 to 0.40.

As it is not clear what value k should take before running the model, we tested all the values from 1 to 50. The ROC-AUC and F1 score are evaluated using test data. The maximum value for both ROC-AUC and F1 score occurs at k=38, with ROC-AUC of 0.63 and F1 score of 0.45. Below are graphs (Fig. 15) showing the trends of ROC-AUC and F1 score as the value of k increases.

Fig. 15. Trends of ROC-AUC and F1 score with value of k

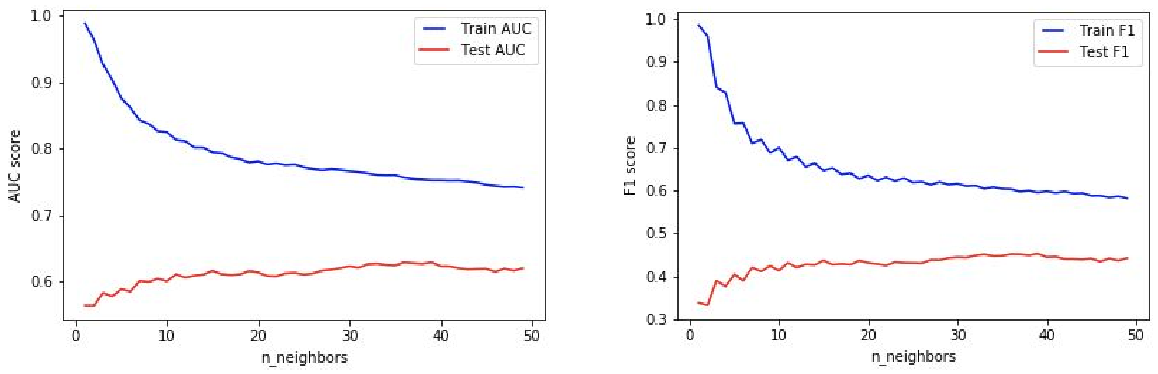


Table 7. Results of KNN

| Model | Precision | Recall | F1 | Accuracy |
|---|---|---|---|---|
| K = 5 | 0.43 | 0.23 | 0.30 | **0.73** |
| **K = 5 with upsampling** | 0.33 | 0.52 | 0.40 | 0.62 |
| **K = 38 with upsampling** | **0.37** | **0.56** | **0.45** | 0.66 |

## 2. Neural Networks

We used Keras for the implementation of our neural network models.
**All model summaries are available in Annex (pg. 26)*

### Input Preprocessing

As we observed during exploratory data analysis, it is important to capture the temporal information within the dataset. Each patient has their assessment taken at least more than once, resulting in a "visit day" feature that specifies which day the patient's assessment was taken. According to this feature and PatientID, we constructed sequential data from the original dataset, with input dimension transformed from 2 dimensional (number of assessments, feature size) to 3 dimensional (number of patients, number of patients' visits, feature size). Padding is applied during the transformation as patients have different number of visits. Later a masking layer is applied in the neural networks architecture to ignore the noise introduced by padding. Table 8 gives the results for each of the models.

Table 8. Overall results of Neural Network Models

| **Model** | **Precision** | **Recall** | **F1** | **Accuracy** |
|---|---|---|---|---|
| Simple RNN | 0.69 | 0.53 | 0.60 | 0.83 |
| biSimpleRNN | 0.70 | 0.57 | 0.63 | 0.84 |
| biLSTM | 0.74 | 0.54 | 0.63 | **0.85** |
| biGRU | 0.73 | 0.56 | 0.64 | **0.85** |
| CNN-biLSTM | 0.73 | **0.58** | **0.65** | **0.85** |
| CNN-biGRU | 0.75 | 0.56 | 0.64 | **0.85** |
| Attention | **0.83** | 0.45 | 0.60 | **0.85** |

### a. Simple RNN

We implement a simple Recurrent Neural Network(RNN), because it is a popular architecture used extensively with use cases consisting of sequential or contextual data. RNNs are designed to retain a memory of previous sequences and their result. We expect the RNN model to capture the sequential memory of each patient visit and learn long-term dependencies in the sequences. It makes use of the temporal information to predict based on the context of each patient's gradual visits and scores. As expected, table x shows that this simple RNN model's outcome is a significant improvement from previous traditional models implemented, increasing F1 score from 0.51 to 60.
**Refer to Annex a. for model summary*

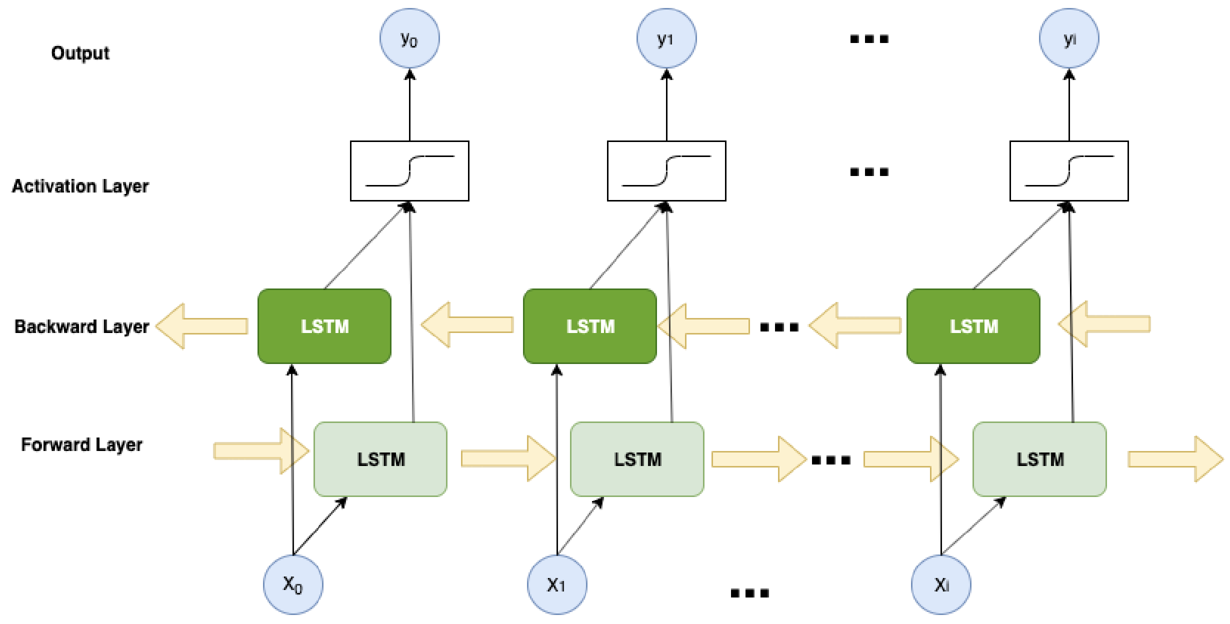
### b. Bidirectional Simple RNN

The input sequence in a Bidirectional RNN is fed in normal time order for one network, and in reverse time order for another. This structure enables the network to have both backward and forward information about the sequence at every time step. This implementation improves our F1 score further from 0.60 to 0.63 because information about forward and backward patterns are captured effectively.
**Refer to Annex b. for model summary*

### c. Bidirectional LSTM (Long Short-term Memory) and GRU (Gated Recurrent Unit) Neural Network

As one of the most popular RNN models, LSTM networks are well-suited to tasks based on time series data and addresses the vanishing gradient issue of simple recurrent networks. Figure 16 summarizes our model structure. As expected, the model outperforms all traditional machine learning models largely, as well as the fully connected feed forward network.

Fig. 16. Bidirectional LSTM Model Architecture



GRU is a less complex variation of LSTM. We also implemented a Bi-directional GRU network and gained similar results as the Bi-directional LSTM.
**Refer to Annex c. and Annex d. for model summary*

## d. CNN-Bidirectional LSTM/GRU

As CNNs are great feature extractors that allow the model to extract information from smaller subregions of the input matrix, we thought it might be helpful to use CNN layers to extract features from smaller period of visit days. We used 4 layers of CNNs with1D kernels applied along the visit days. The operation here is analogous to constructing n-grams in text. After that, we add a BiLSTM layer (or BiGRU layer) and the output layer.

Despite some minor enhancement, there is no significant breakthrough of performance observed after using CNN layers. The reason could be that our sequence is too short for a meaningful sub-sequence feature extraction with an average of length 8.

**Refer to Annex e. and Annex f. for model summary*

## e. Attention Architecture

The encoder blocks of the Transformer architecture uses multi-headed self attention, which effectively captures the dependency or similarity between parts of the sequential value. In our case, with the use of multi-head self attention, the model would attend to assessment records that have similar or relevant PANSS scores on other visit days of the same patient when it predicts the label of a certain assessment. We found a keras transformer library online [5] and implemented a Transformer based model using the library.

The model achieves the highest precision score across all (significantly higher than the second highest). However, in terms of F1 score, it does not perform as well as the recurrent models. There are 2 possible reasons for this. Firstly, rather than a few similar or relevant assessment records, the trend observed from the full sequence of records could be more helpful for auditing. Secondly, while processing long and far dependencies is a common difficulty for recurrent models where attention mechanism shows its advantage, our data sequence may be too short that the recurrent neural networks would handle it perfectly without any information loss from visit days far away.

**Refer to Annex g. for model summary*

## Conclusion

Table 9 gives a summary of results for all the models we implemented. As shown in the table, CNN bidirectional LSTM model gives the highest F1 score.

Table 9. Summary of Results from all models

| **Model** | **Precision** | **Recall** | **F1** | **Accuracy** |
|---|---|---|---|---|
| Logistic Regression | 0.35 | **0.67** | 0.45 | 0.63 |
| Naïve Bayes | 0.35 | 0.54 | 0.43 | 0.64 |
| Support Vector Machine | 0.44 | 0.61 | 0.51 | 0.71 |
| Random Forest | 0.55 | 0.38 | 0.45 | 0.77 |
| KNN | 0.37 | 0.56 | 0.45 | 0.66 |
| Simple RNN | 0.69 | 0.53 | 0.60 | 0.83 |
| biSimpleRNN | 0.70 | 0.57 | 0.63 | 0.84 |
| biLSTM | 0.74 | 0.54 | 0.63 | **0.85** |
| biGRU | 0.73 | 0.56 | 0.64 | **0.85** |
| CNN-biLSTM | 0.73 | 0.58 | **0.65** | **0.85** |
| CNN-biGRU | 0.75 | 0.56 | 0.64 | **0.85** |
| Attention | **0.83** | 0.45 | 0.60 | **0.85** |

## References

[1]. Opler, Mark G.A.; Yavorsky, Christian; Daniel, David G. (2017-12-01). "Positive and Negative Syndrome Scale (PANSS) Training". *Innovations in Clinical Neuroscience*. **14** (11–12): 77–81. ISSN 2158-8333. PMC 5788255. PMID 29410941.

[2]. Bell MD, Lysaker PH, Beam-Goulet JL, Milstein RM, Lindenmayer JP. Five-component model of schizophrenia: assessing the factorial invariance of the positive and negative syndrome scale. Psychiatry Res. 1994;52:295–303. doi: 10.1016/0165-1781(94)90075-2. [PubMed] [CrossRef] [Google Scholar]

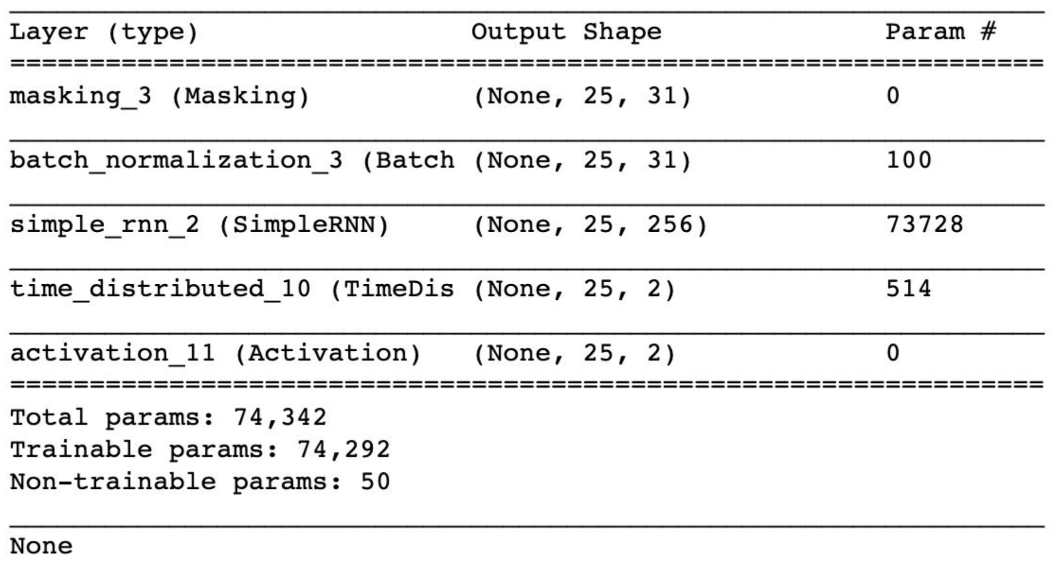
[3]. Lindenmayer JP, Grochowski S, Hyman RB. Five factor model of schizophrenia: replication across samples. Schizophr. Res. 1995;14:229–234. doi: 10.1016/0920-9964(94)00041-6. [PubMed] [CrossRef] [Google Scholar]

[4]. White L, Harvey PD, Opler L, Lindenmayer J. Empirical assessment of the factorial structure of clinical symptoms in schizophrenia. Psychopathology. 1997;30:263–274. doi: 10.1159/000285058.[PubMed] [CrossRef] [Google Scholar]

[5] https://github.com/kpot/keras-transformer

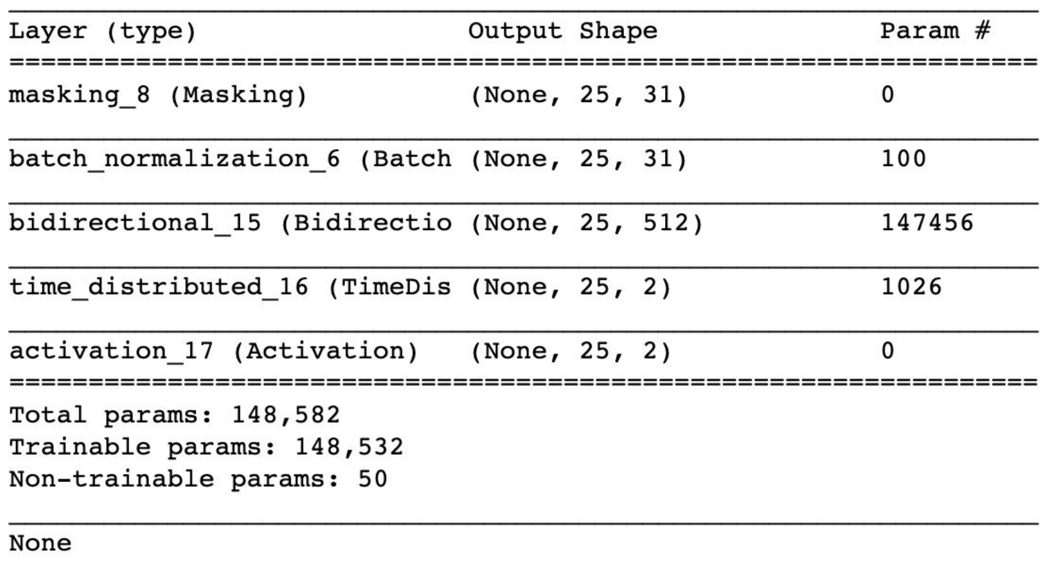
## ANNEX (Model Structures for Neural Networks)

### a. Simple RNN



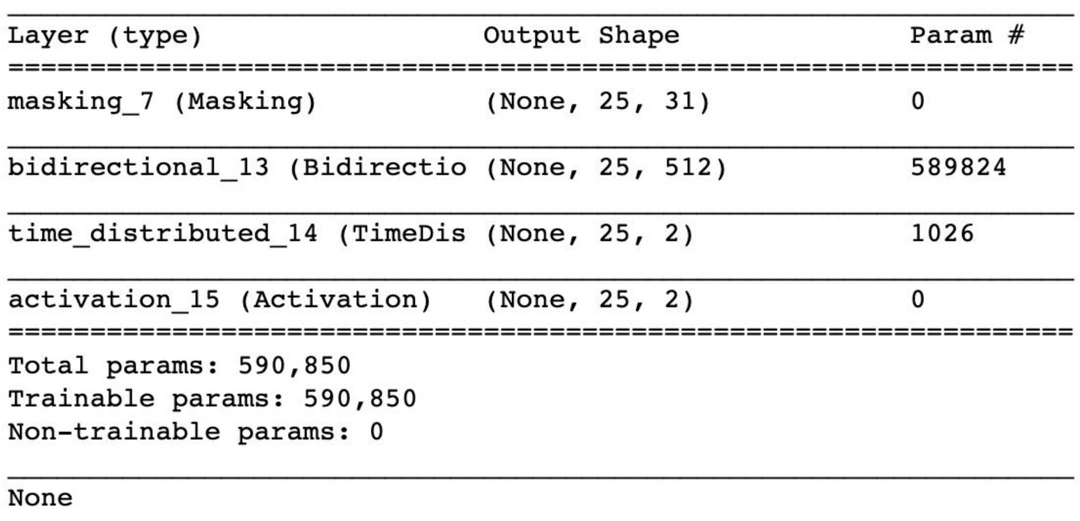
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
masking_3 (Masking)          (None, 25, 31)            0
_________________________________________________________________
batch_normalization_3 (Batch (None, 25, 31)            100
_________________________________________________________________
simple_rnn_2 (SimpleRNN)     (None, 25, 256)           73728
_________________________________________________________________
time_distributed_10 (TimeDis (None, 25, 2)             514
_________________________________________________________________
activation_11 (Activation)   (None, 25, 2)             0
=================================================================
Total params: 74,342
Trainable params: 74,292
Non-trainable params: 50
_________________________________________________________________
None
```

### b. Bidirectional Simple RNN



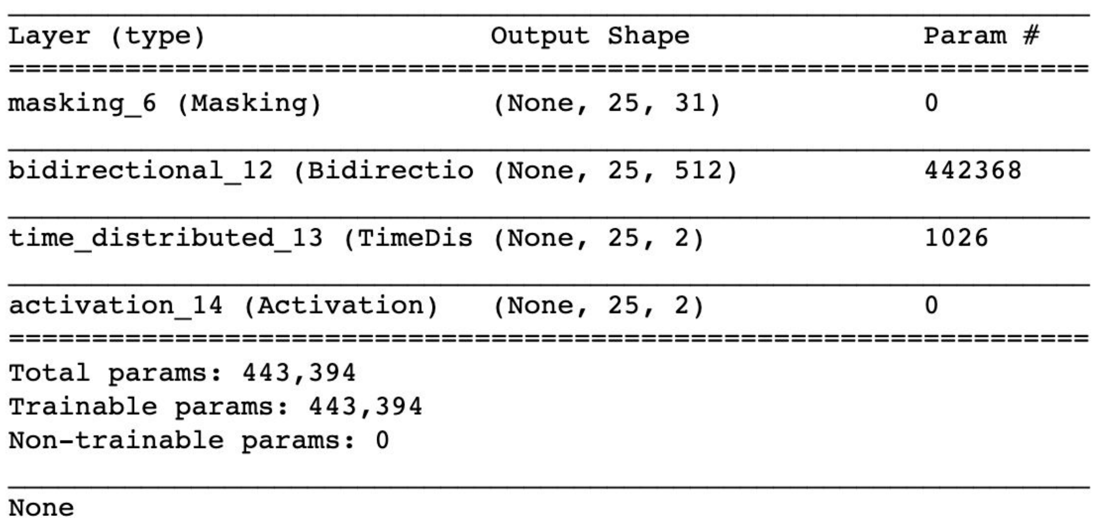
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
masking_8 (Masking)          (None, 25, 31)            0
_________________________________________________________________
batch_normalization_6 (Batch (None, 25, 31)            100
_________________________________________________________________
bidirectional_15 (Bidirectio (None, 25, 512)           147456
_________________________________________________________________
time_distributed_16 (TimeDis (None, 25, 2)             1026
_________________________________________________________________
activation_17 (Activation)   (None, 25, 2)             0
=================================================================
Total params: 148,582
Trainable params: 148,532
Non-trainable params: 50
_________________________________________________________________
None
```

## c. Bidirectional LSTM



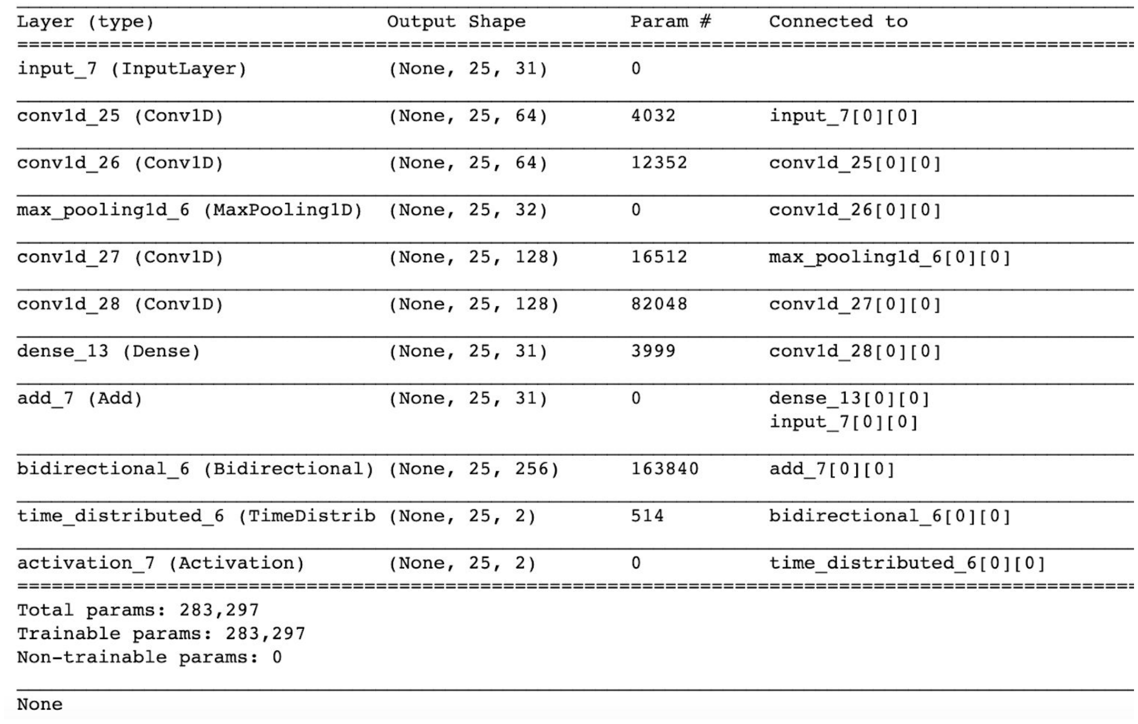
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
masking_7 (Masking)          (None, 25, 31)            0
_________________________________________________________________
bidirectional_13 (Bidirectio (None, 25, 512)           589824
_________________________________________________________________
time_distributed_14 (TimeDis (None, 25, 2)             1026
_________________________________________________________________
activation_15 (Activation)   (None, 25, 2)             0
=================================================================
Total params: 590,850
Trainable params: 590,850
Non-trainable params: 0
_________________________________________________________________
None
```

## d. Bidirectional GRU



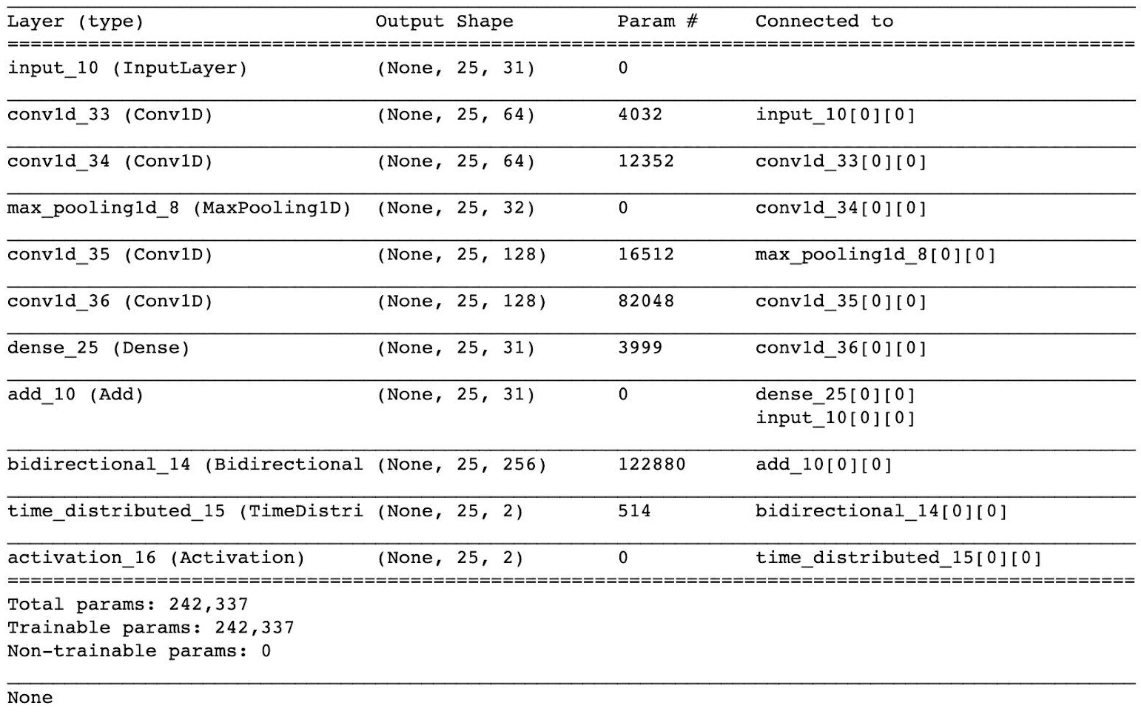
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
masking_6 (Masking)          (None, 25, 31)            0
_________________________________________________________________
bidirectional_12 (Bidirectio (None, 25, 512)           442368
_________________________________________________________________
time_distributed_13 (TimeDis (None, 25, 2)             1026
_________________________________________________________________
activation_14 (Activation)   (None, 25, 2)             0
=================================================================
Total params: 443,394
Trainable params: 443,394
Non-trainable params: 0
_________________________________________________________________
None
```

## e. CNN-LSTM



```
________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
================================================================================================
input_7 (InputLayer)            (None, 25, 31)       0
________________________________________________________________________________________________
conv1d_25 (Conv1D)              (None, 25, 64)       4032        input_7[0][0]
________________________________________________________________________________________________
conv1d_26 (Conv1D)              (None, 25, 64)       12352       conv1d_25[0][0]
________________________________________________________________________________________________
max_pooling1d_6 (MaxPooling1D)  (None, 25, 32)       0           conv1d_26[0][0]
________________________________________________________________________________________________
conv1d_27 (Conv1D)              (None, 25, 128)      16512       max_pooling1d_6[0][0]
________________________________________________________________________________________________
conv1d_28 (Conv1D)              (None, 25, 128)      82048       conv1d_27[0][0]
________________________________________________________________________________________________
dense_13 (Dense)                (None, 25, 31)       3999        conv1d_28[0][0]
________________________________________________________________________________________________
add_7 (Add)                     (None, 25, 31)       0           dense_13[0][0]
                                                                 input_7[0][0]
________________________________________________________________________________________________
bidirectional_6 (Bidirectional) (None, 25, 256)      163840      add_7[0][0]
________________________________________________________________________________________________
time_distributed_6 (TimeDistrib (None, 25, 2)        514         bidirectional_6[0][0]
________________________________________________________________________________________________
activation_7 (Activation)       (None, 25, 2)        0           time_distributed_6[0][0]
================================================================================================
Total params: 283,297
Trainable params: 283,297
Non-trainable params: 0
________________________________________________________________________________________________
None
```

## f. CNN-GRU



```
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_10 (InputLayer)           (None, 25, 31)       0
__________________________________________________________________________________________________
conv1d_33 (Conv1D)              (None, 25, 64)       4032        input_10[0][0]
__________________________________________________________________________________________________
conv1d_34 (Conv1D)              (None, 25, 64)       12352       conv1d_33[0][0]
__________________________________________________________________________________________________
max_pooling1d_8 (MaxPooling1D)  (None, 25, 32)       0           conv1d_34[0][0]
__________________________________________________________________________________________________
conv1d_35 (Conv1D)              (None, 25, 128)      16512       max_pooling1d_8[0][0]
__________________________________________________________________________________________________
conv1d_36 (Conv1D)              (None, 25, 128)      82048       conv1d_35[0][0]
__________________________________________________________________________________________________
dense_25 (Dense)                (None, 25, 31)       3999        conv1d_36[0][0]
__________________________________________________________________________________________________
add_10 (Add)                    (None, 25, 31)       0           dense_25[0][0]
                                                                 input_10[0][0]
__________________________________________________________________________________________________
bidirectional_14 (Bidirectional (None, 25, 256)      122880      add_10[0][0]
__________________________________________________________________________________________________
time_distributed_15 (TimeDistri (None, 25, 2)        514         bidirectional_14[0][0]
__________________________________________________________________________________________________
activation_16 (Activation)      (None, 25, 2)        0           time_distributed_15[0][0]
==================================================================================================
Total params: 242,337
Trainable params: 242,337
Non-trainable params: 0
__________________________________________________________________________________________________
None
```

## g. Attention Architecture

```
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
panss_scores (InputLayer)       (None, 25, 32)       0
__________________________________________________________________________________________________
transformer0_self_attention (Mu (None, 25, 32)       4096        panss_scores[0][0]
__________________________________________________________________________________________________
transformer0_dropout (Dropout)  (None, 25, 32)       0           transformer0_self_attention[0][0]
                                                                 transformer0_transition[0][0]
__________________________________________________________________________________________________
transformer0_add (Add)          (None, 25, 32)       0           panss_scores[0][0]
                                                                 transformer0_dropout[0][0]
                                                                 transformer0_normalization1[0][0]
                                                                 transformer0_dropout[1][0]
__________________________________________________________________________________________________
transformer0_normalization1 (La (None, 25, 32)       64          transformer0_add[0][0]
__________________________________________________________________________________________________
transformer0_transition (Transf (None, 25, 32)       8352        transformer0_normalization1[0][0]
__________________________________________________________________________________________________
transformer0_normalization2 (La (None, 25, 32)       64          transformer0_add[1][0]
__________________________________________________________________________________________________
transformer1_self_attention (Mu (None, 25, 32)       4096        transformer0_normalization2[0][0]
__________________________________________________________________________________________________
transformer1_dropout (Dropout)  (None, 25, 32)       0           transformer1_self_attention[0][0]
                                                                 transformer1_transition[0][0]
__________________________________________________________________________________________________
transformer1_add (Add)          (None, 25, 32)       0           transformer0_normalization2[0][0]
                                                                 transformer1_dropout[0][0]
                                                                 transformer1_normalization1[0][0]
                                                                 transformer1_dropout[1][0]
__________________________________________________________________________________________________
transformer1_normalization1 (La (None, 25, 32)       64          transformer1_add[0][0]
__________________________________________________________________________________________________
transformer1_transition (Transf (None, 25, 32)       8352        transformer1_normalization1[0][0]
__________________________________________________________________________________________________
transformer1_normalization2 (La (None, 25, 32)       64          transformer1_add[1][0]
__________________________________________________________________________________________________
transformer2_self_attention (Mu (None, 25, 32)       4096        transformer1_normalization2[0][0]
__________________________________________________________________________________________________
transformer2_dropout (Dropout)  (None, 25, 32)       0           transformer2_self_attention[0][0]
                                                                 transformer2_transition[0][0]
__________________________________________________________________________________________________
transformer2_add (Add)          (None, 25, 32)       0           transformer1_normalization2[0][0]
                                                                 transformer2_dropout[0][0]
                                                                 transformer2_normalization1[0][0]
                                                                 transformer2_dropout[1][0]

__________________________________________________________________________________________________
transformer2_normalization1 (La (None, 25, 32)       64          transformer2_add[0][0]
__________________________________________________________________________________________________
transformer2_transition (Transf (None, 25, 32)       8352        transformer2_normalization1[0][0]
__________________________________________________________________________________________________
transformer2_normalization2 (La (None, 25, 32)       64          transformer2_add[1][0]
__________________________________________________________________________________________________
transformer3_self_attention (Mu (None, 25, 32)       4096        transformer2_normalization2[0][0]
__________________________________________________________________________________________________
transformer3_dropout (Dropout)  (None, 25, 32)       0           transformer3_self_attention[0][0]
                                                                 transformer3_transition[0][0]
__________________________________________________________________________________________________
transformer3_add (Add)          (None, 25, 32)       0           transformer2_normalization2[0][0]
                                                                 transformer3_dropout[0][0]
                                                                 transformer3_normalization1[0][0]
                                                                 transformer3_dropout[1][0]
__________________________________________________________________________________________________
transformer3_normalization1 (La (None, 25, 32)       64          transformer3_add[0][0]
__________________________________________________________________________________________________
transformer3_transition (Transf (None, 25, 32)       8352        transformer3_normalization1[0][0]
__________________________________________________________________________________________________
transformer3_normalization2 (La (None, 25, 32)       64          transformer3_add[1][0]
__________________________________________________________________________________________________
time_distributed_3 (TimeDistrib (None, 25, 2)        66          transformer3_normalization2[0][0]
__________________________________________________________________________________________________
activation_3 (Activation)       (None, 25, 2)        0           time_distributed_3[0][0]
==================================================================================================
Total params: 50,370
Trainable params: 50,370
Non-trainable params: 0
__________________________________________________________________________________________________
None
```