Predicting Depression using RNA

Ziv Lautman (lautman@stanford.edu), Harvey Wang (hawang97@stanford.edu), Shannon Xiao (sxiao1@stanford.edu)

Project Video: https://youtu.be/MG-gLSZiFMg

1. Introduction

In 2020, one in every five adults and one in every three young adults in the US (ages 18-25) experienced mental illness. Combined, more than 15 million people have had serious thoughts about suicide [1]. Mental health issues, such as depression, are a serious concern in our society today and a key factor in tackling this issue lies in its diagnosis. Depression can manifest itself in many ways [2]. However, there are currently very limited diagnostic tools to assess mental health, which hinders the treatment process. In this project, we were looking to investigate the use of machine learning models to help with mental illness diagnoses, using RNA sequencing data. At sample time, each individual answered the Beck Depression Inventory-II (BDI-II) questionnaire, which measures the existence and severity of symptoms of depression. Thus, the inputs to our algorithm are individual RNA sequencing results of 11,000 genes (i.e. which genes were expressed and a numerical estimate of their expression magnitude), we then experimented with PCA, SVM, a neural network, logistic regression, and anomaly detection to provide a classifier for depressed and non-depressed classes.

2. Related Work

We started off by reading the research paper "Identification of Diagnostic Markers for Major Depressive Disorder Using Machine Learning Methods" by Zhao et al. [2], which was published in the *Frontiers in Neuroscience* journal, in order to analyze existing attempts at our problem. We compared the features (genes) that Zhao et al. considered in their study to be highly correlated with depression, and identified 98 of the 137 genes in our dataset. Hence, we pruned our original dataset to match the genes from Zhao et al.'s study and resulted with a total of 124 samples with 98 genes for the feature columns and one output column for the depression score.

3. Dataset

As part of one of our group member's ongoing research at Dr. Michael Snyder's lab, we have obtained a proprietary dataset, containing 124 samples of RNA sequences, 26 of which are classified as depressed while the others are not depressed. The data came from 45 individuals, sampled at various times in the course of six months. Figure 1 provides a snippet of the dataset. Since the BDI values (the depression scores) of our proprietary data set were on a scale from 0-60, we adjusted the output values to fit a binary scale of 0/1. Samples with a BDI at or above 14 were considered depressed and hence considered 1.0, while the rest of the samples were 0.0.

	KLRB1	WWC3	MAFG	AKR1C3	DTYMK	ACTR8	FKBP2	MKNK1	bdi_total
0	5.927547	4.906848	3.200140	0.000000	4.158720	4.570252	5.829772	0.000000	1.0
1	5.831458	4.899783	5.039019	3.282688	3.050514	0.000000	5.101140	1.532666	1.0
2	7.000639	4.920201	4.117801	0.000000	2.281300	3.152017	5.117801	3.509569	0.0
3	4.242291	3.146076	1.873057	0.000000	1.095450	0.000000	2.129397	3.194985	0.0
4	7.038439	3.632118	3.700022	0.000000	2.542481	3.998546	3.427004	3.683344	0.0
...	...	...	...	...	...	...	...	...	...
119	4.140027	2.520299	0.970102	0.162747	0.000000	0.000000	3.589011	2.520299	0.0
120	7.556936	3.400761	1.997405	2.743648	0.000000	0.000000	5.093057	0.000000	0.0
121	6.390697	3.659828	0.744220	0.000000	0.284789	1.454714	4.532716	3.588570	0.0
122	6.435180	2.986174	3.827809	2.282159	1.649891	1.851525	4.252775	2.737354	1.0
123	4.390128	2.956033	2.530727	0.000000	0.000000	2.101883	4.893297	3.355640	0.0

Figure 1: Project dataset, rows are individual samples; columns are genes expressed alongside the BDI score of each individual sample

We normalized the input data with feature scaling and mean normalization. To split our data into training, cross-validation, and test sets, we used the train_test_split method from the scikit-learn Python library. The training set was 60% of our overall dataset with 60 not depressed and 14 depressed samples, and the cross-validation and testing sets were each 20% of the overall dataset each with 19 not depressed and 6 depressed samples. With that, we started to train our learning models.

4. Methods

We chose to implement a few algorithms, starting with PCA for unsupervised learning in hopes of spatially separating the depressed samples from the non-depressed. Then, we explored supervised learning methods such as logistic regression, SVM, and neural networks in order to produce classification models for our data. Since we had a skewed dataset, we also experimented with anomaly detection.

4.1. Principal Component Analysis (PCA)

PCA is an unsupervised method that enables visualization and the identification of patterns in high-dimensional data based on the correlation between the different features. Using the covariance matrix of the data, PCA essentially finds the directions of maximum variance represented by eigenvectors and their magnitude of variance (eigenvalues) explained in the data. The number of eigenvectors is equal to or less than the number of features (dimensions) of the data. Since PCA is sensitive to data scaling, which affects the magnitude of the variance, we first used mean normalization of the features prior to performing PCA.

4.2. Logistic Regression

Logistic regression is useful for binary classification, allowing us to predict discrete values. Because our problem concerns classifying depressed versus non-depressed cases, we found this algorithm fitting. Here, our hypothesis function involves both θ and the sigmoid/logistic function, $h_\theta(x)$.

$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$

θ is the fitted parameters that will map x to a value that is in turn plugged into the sigmoid function. $h_\theta(x)$ constrains the output to the (0,1) interval, which is desired for binary problems. The hypothesis function determines a decision boundary, which separates the data from each class. The value of $h_\theta(x)$ provides a probability of the output belonging to the class represented by $y = 1$; to get a discrete classification, an output ≥ 0.5 (when $\theta^T x \geq 0$) is considered $y = 1$, else $y = 0$.

The cost function J measures the accuracy of our function. m is the number of training samples and n is the number of features; i represents each training sample and j represents each weight :

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)} log\left(h_\theta\left(x^{(i)}\right)\right) + \left(1 - y^{(i)}\right) log\left(1 - h_\theta\left(x^{(i)}\right)\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

The term $\frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$ is the regularization term.

Changing the value of λ allows us to penalize the coefficients to prevent overfitting.

4.3. Support Vector Machine (SVM)

We also tried SVM, as it is also typically used for classification. SVM focuses on finding a decision boundary that maximizes its distance from the data of each class, and provides a prediction of which class the example belongs to. Logistic regression provides the likelihood of an example belonging to a class and tries to maximize that probability. As such, logistic regression is more sensitive to outliers. SVM also makes use of kernels, which are a type of similarity function. These kernels map the features to a higher-dimension, allowing the data to be able to be separated more easily. The cost function of an SVM is similar to the logistic regression cost function, with some modifications.

$$J(\theta) = C\sum_{i=1}^{m}\left[y^{(i)} cost_1\left(\theta^T x^{(i)}\right) + \left(1 - y^{(i)}\right) cost_0\left(\theta^T x^{(i)}\right)\right] + \frac{1}{2}\sum_{j=1}^{n}\theta_j^2$$

By convention, a regularization parameter C replaces λ, and C is added in front of the first term; this changes the term that is prioritized. The $\frac{1}{m}$ term is removed and the logarithmic functions are replaced by functions specific to this algorithm.

4.4. Neural Network

Artificial neural network is a supervised learning method that is set to mimic a biological neural network by presenting a set of mathematical operations through layers of computation nodes (neurons). The layers in an artificial neural network are fully connected layers (by matrix multiplications) followed by a non-linear transform

(ReLU, sigmoid, and others). There are various hyperparameters to choose when building a neural network, such as: number of layers, number of nodes in each layer, learning rate, regularization, and others.

4.5. Anomaly Detection

In this generative learning algorithm, the dataset is modeled as a Gaussian distribution, with a mean μ and a standard deviation σ. To use this method, we assume that the data is distributed as a Gaussian distribution. Each example in the dataset can be modeled with the following function, with p being the probability that the sample is "normal":

$$p(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

For datasets with multiple features, like ours, the probability of each feature was calculated and multiplied together, with j representing each feature:

$$p(x) = \prod_{j=1}^{n} p\left(x_j; \mu_j, \sigma_j\right) = \frac{1}{\sqrt{2\pi}\sigma_j} exp\left(-\frac{\left(x_j-\mu_j\right)^2}{2\sigma_j^2}\right)$$

If the probability of an input sample is very small, below a threshold ε, then it is flagged as an anomaly. To determine an accurate probability threshold for flagging samples, we can compare the calculated probabilities with a labeled dataset and adjust ε accordingly.

5. Results

5.1. PCA

We used the PCA module from the scikit-learn Python library. Applying PCA on our dataset did not seem to reveal a large explained variance that could help classify our dataset. Figure 3a shows the cumulative explained variance for each PC component, the first PC explained ~17%, the second ~5% and then drops sharply as can be seen from the figure. Figure 3b shows the projections of PC1 and PC2, with color encoding depressed vs non-depressed, with no apparent spatdial difference.

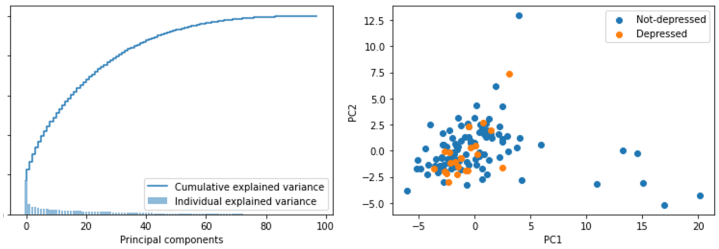

Figure 2 & 3: (a) cumulative explained variance for each PC (b) Projection of PC1 and PC2 of all samples

5.2. Logistic regression

To train our logistic regression classifier, we first added a column of 1s to the beginning of our input (X) matrix for the bias term. Our initial weights (θ) were randomized from a uniform distribution over [0, 1). We optimized the weights using the fmin_tnc function from the SciPy Python library with our cost and gradient functions, training X- and y-data, and regularization constant (λ) as parameters. To choose our parameter λ, we created a list of 12 values from 0 to 10.24 with each value incremented by a power of 2.

For each λ, we learned some θ and calculated F1 scores for our error metric. We then chose the λ that corresponded to the highest F1 score.

After training the model with all 98 features, we discovered that the model would oftentimes classify the validation and test samples as all 0. Given the large number of features, we determined that we were overfitting to our training set, so we decided to limit our input to only 20 features (not including bias). We chose the features by randomly sampling 20 genes at a time to train the model with and selected the set of genes that yielded the highest F1 score from the validation set. After determining a subset of 20 gene features, we iterated through all the λ values and found that a regularization parameter of $\lambda = 0.32$ had the highest validation F1 score of 0.8.

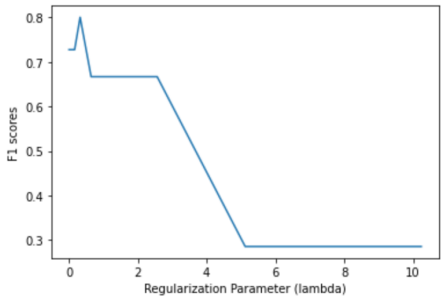

Figure 4: Plot of the F1 scores of the validation set (using the subset of features) against lambda values

We then tested the weights learned from the training set (with the subset of 20 gene features) and the optimal lambda value of 0.32 on the test set and calculated an F1 score of 0.44.

5.3. SVM

Using scikit-learn's SVM module, we trained an SVM model. We iterated over a list of C-values from 0.005 to 0.1, incremented by 0.005, to determine the optimal regularization parameter. In each iteration, the SVC function, with a linear kernel and a class weight of 50 for the data with a y-value of 1.0, and the fit function were used. The use of class weight also prevented divisions by zero when calculating the precision and recall; like in logistic regression, the algorithm predicted zeros because of our skewed data set. Precision and recall functions were used, and the F1 score was calculated from the precision and recall values. A C-value of 0.005 gave the highest scores. So, the algorithm with a C-value of 0.005 was used on the test set, giving a precision score of 0.29, a recall score of 0.83, and an F1 score of 0.43. Additionally, an ROC curve was made using the test data. The AUC was 0.64.

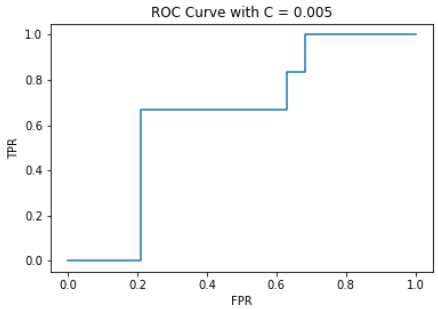

Figure 5: The ROC curve

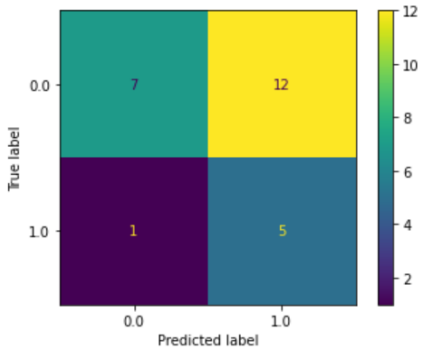

Figure 6: The confusion matrix of the test data

5.4. Neural Network

We trained a neural network with various different hidden layers and tested several learning rates alongside regularization parameter λ. As previously, the data was split into 60% training examples, 20% cross-validation examples, and 20% test examples. The best performance hyperparameters were: 3 hidden layers (98, 48, 1), learning rate of 0.01, lambda of 0.1, and 10,000 iterations (full analysis including learning curves figures are in the source code [3]). The obtained performance had an F1 score of 0.53, a precision of 0.44, and a recall of 0.66. Additionally, an ROC curve was made using the test data and the AUC was 0.69 (Figure 7). Figure 8 shows the learning curve for the parameters above. It can be seen that our model suffers from overfitting. Unfortunately, only at the regime of the highest overfitting (high variance) does the model output values other than 0.

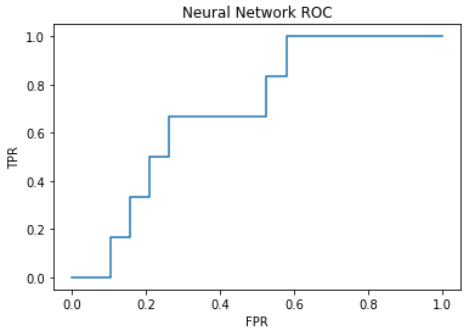

Figure 7: The ROC curve from of the Neural Network

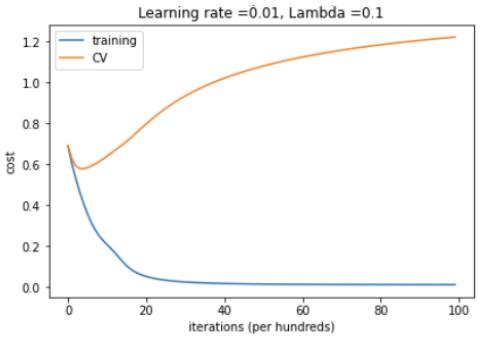

Figure 8: Neural Network learning curve example

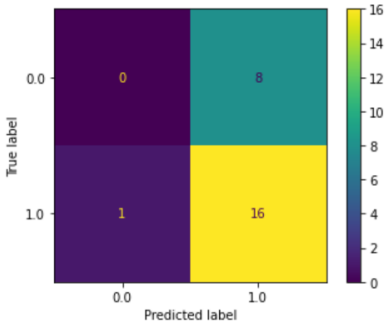

Figure 9: The confusion matrix of the test data

5.5. Anomaly Detection

Because of the large disparity between the amount of depressed and non-depressed cases in our data, we also tried an anomaly detection algorithm. Depressed cases were considered to be anomalies. Like previous algorithms, the data was split into 60% training examples, 20% cross-validation examples, and 20% test examples. However, the training set consisted of only non-depressed examples in order for the algorithm to learn the probabilities of normal cases. So, we could not apply the train_test_split method for the training examples. Instead, we ordered the dataset by ascending values based on the BDI score converted into binary. The data with 0.0-values were separated from the dataset and randomized. The split dataset was then reconstituted. The top 60% were taken to form the training dataset; the remaining 40% was randomized through train_test_split. The cross-validation set had 16 0.0-values and 9 1.0-values; the test set had 8 0.0-values and 17-1.0 values. As can be noted, the cross-validation and test sets are uneven.

The multivariate_normal method from the SciPy library was used to model the Gaussian distribution. The threshold was chosen by training on the cross-validation set. Probability values from the minimum value to the maximum value were tried, in increments of the range divided by 1000. The threshold with the best F1 score was chosen. The threshold was found to be 6.5e-49, with an F1 score of 0.48. This threshold value was applied on a test set. The precision score was 0.67; the recall score was 0.94; and the F1 score was 0.78.

As can be seen in Figure 9, the algorithm is predicting all 1.0s, except for one input; the majority of the test data are 1.0s. This algorithm was successful because of a "shotgun" approach.

6. Conclusions

Mental health is becoming a serious concern in our society and enhancing diagnostic tools could potentially improve the issue [1]. In this project we examined the possibility of predicting one's mental state, specifically depression, by analyzing its RNA expression taken from white blood cells. From the five machine learning models we tested, anomaly detection performed the best with a precision score of 0.67; a recall score of 0.94; and a F1 score of 0.78. While this is a relatively good F1 score, the general results of our experiments were not great, since ideally, we wanted scores > 0.85.

One issue we acknowledge is the fact that our dataset size was quite small and skewed. We believe that obtaining a larger dataset could yield better accuracy results, as demonstrated by Zhao et al. [2]. For future work, we could also try different learning algorithms stitch as a multi-class classification model based on the BDI scores.

References

[1] National Alliance on Mental Illness. 'Mental Health By the Numbers'. Feb 2022. [ONLINE]: https://www.nami.org/mhstats

[2] Shu Zhao, Zhiwei Bao, Xinyi Zhao, Mengxiang Xu, Ming D. Li, and Zhongli Yang. 'Identification of diagnostic markers for major depressive disorder using machine learning methods'. Frontiers in Neuroscience, 15, 2021.

[3] Source code (requires Stanford login): https://code.stanford.edu/lautman/cs129