Advanced Machine Learning Report

By Shruti Roy and Audrey Barszcz

1. Abstract

One of the largest obstacles for cancer treatment is the lack of a non-invasive and effective screening methods for early detection of cancer. The microRNAs have been widely recognised as promising biomarkers that can help detect the disease in early stages.

The data was modeled using variations of XgBoost, Random Forest, and OneVsRest classifiers to identify the stage of cancer of the patient. The best accuracy attained during the experiments was 74.05% using XgBoost with hyper-parameter tuning on a full dataset.

2. Data Description

In this case-controlled study, we use miRNA gene expression (978 features) from 8000 patients to diagnose 5 types of different stages of cancer, combined from 21 different datasets. A robust model will also prove further that miRNA profiles are good predictors for pre-emptive diagnosis of cancer.

3. Exploratory Data Analysis

1. Data imbalance

Data imbalance can be a problem when there is a disproportionate number of observations for each level of the target variable.

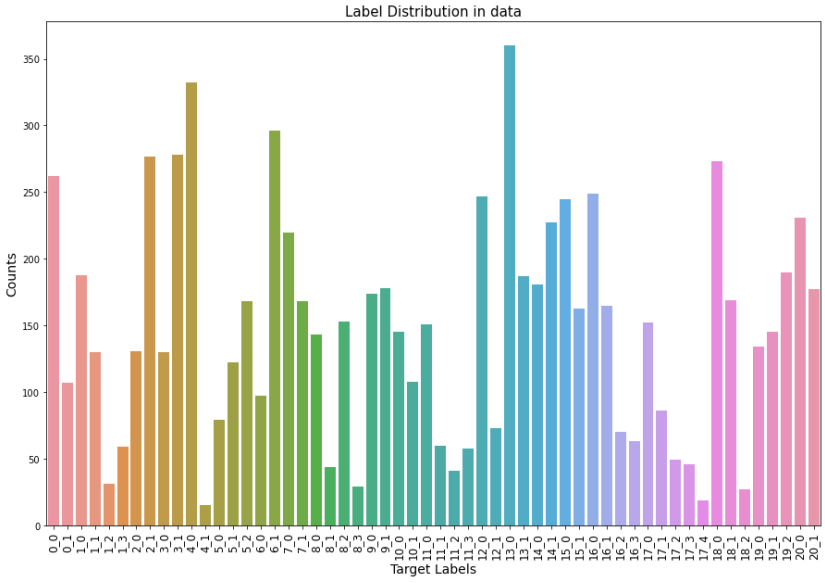

Fig 1.1: Data Imbalance: Target distribution in the training data

2. t-SNE representation of data

t-SNE is a dimension reduction technique that maps high dimensional data down to 2-3 dimensions, making it useful for plotting. Since the algorithm would take a considerable amount of time to run on the full data, we show two different transformations of the data that reduce some of the dimensionality of the data before performing t-SNE: PCA and truncated SVD. Based on the plots of the data, we can see clusters of different classes of observations grouped together in a nonlinear fashion leading us to believe that a linear model would not work well for this data.

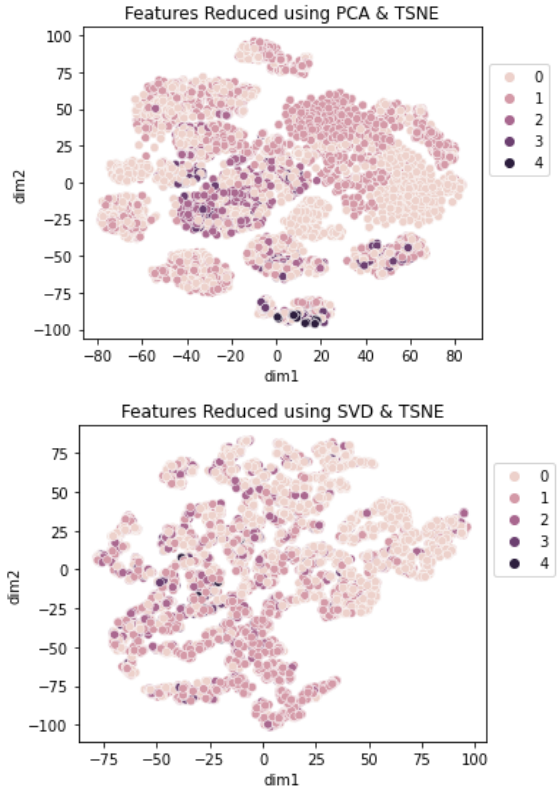

Fig 1.2: Observations plotted using SVD/PCA + t-SNE dimension reduction

4. Feature Engineering

1. Label Encoding the target variables based on Problem ID

Instead of choosing to create 21 models, we decided to take a different approach. We concatenated the problem ID and target to identify the target for each problem ID since the same target value for different problem IDs, implies different types of cancer. Then we used label encoder, and passed the full dataset through the models. After prediction we reverse label encoded the target variables to get the original values.

2. Dimensionality reduction using PCA

Since we have 979 features, we used Principal Component Analysis (PCA) to simplify the high-dimensional data. PCA transforms the data into fewer dimensions which represent the full dataset, hence improving processing speed, and sometimes even accuracy of the model. We noticed that with less than 30 features, 60% of the Cumulative Proportion variance was explained (See figure 1.4 in appendix).

3. Regularized Target Mean Encoding

We used the target variable as a basis to encode our “problem_id” variable. Since this process reduces cardinality, it helps the tree models to reach better loss faster (shorter tree). This also helps give us extra information about the target dataset.

4. Linear Discriminant Analysis to reduce features

LDA is a classifier, which fits conditional densities to the data based on the target variable. This was applied on all features except problem_id since it is a categorical variable to reduce the dimensionality.

5. SMOTE

SMOTE is a way of dealing with imbalanced data by over-sampling the underrepresented class. Since our data is highly skewed towards labels 0 and 1, our models have a tendency to only predict labels 0 and 1 and not predict other labels. SMOTE over-samples the under-represented classes in the dataset so that the model can learn features that correspond to those classes and can accurately predict them in validation and test sets. In our models, we used SMOTE with a sampling strategy of minority and created samples based on 9 nearest neighbors.

6. Machine learning methods

1. Extreme Gradient Boosting (XgBoost)

Extreme gradient boosting is an ensemble decision-tree based model that uses additive modeling framework (boosting). Tree models work well on structured tabular data, and hence it was appropriate for this problem. The models are built sequentially by reducing the residuals (or errors) of the previous models. In this project, we combined XgBoost with various feature engineering methods to derive predictions.

Experimental Results with XgBoost

Since we have 981 features, we created our baseline models by first doing dimensionality reduction using Principal Component Analysis and Linear Discriminant Analysis. Although they gave fairly good results (71.56% and 68.59%, respectively on validation), the accuracy on the test was affected by information loss.

We then ran vanilla XgBoost on the full dataset, and performed hyperparameter tuning on the learning rate and scale_pos_weight parameters. By decreasing the learning rate and increasing the scale_pos_weight parameters, this helped with the imbalanced dataset. We set the scale_pos_weight to be 300 since the ratio of the most occurring target variable (Class 0) to the least occurring target variable (Class 4) is approximately three hundred. Since we are using the full dataset, this model fitting took longer than the other models, and we finally got the best accuracy of 74% on the test dataset. Due to the latency of this model, it might not be scalable for big data. Hence, we can either restrict the dimensions of the data - with the tradeoff of accuracy - or explore alternative machine learning algorithms such as LightGBM.

2. RandomForest (RF)

RandomForest is an ensemble decision-tree based method that averages leaf predictions, in the case of regression, or takes the mode of leaf predictions, in the case of classification. We again chose a tree-based method since the data we are working with is tabular. The goal of RandomForest is that every tree is slightly different from the others in the ensemble, that way when averaged, the overall prediction is more accurate. Each tree in the ensemble is trained on different segments of the data and at each node, a different

subset of features is examined to determine which feature to make the split on. In this project, we combine RandomForest with OneVsRest and SMOTE.

3. OneVsRest (OVR)

OneVsRest classification breaks a multiclass classification problem down to multiple binary classification problems by training a model to perform binary classification on a particular level of the target variable vs. all other levels of the target variable. The downside to this model is that it requires a model for every level of the target, so instead of having one RandomForest model, we now have 5, one for each level of the target variable. So training the OneVsRest model with RandomForest may not be scalable to larger datasets.

Experimental Results with RF, OVR, and SMOTE

We first tried fitting a model using RandomForest along with hyperparameter tuning. The results on validation for that model were 67.39% with a log loss of 0.7884. We thought this model could be improved upon using the OneVsRest classifier with RandomForest as the estimator. The combination of models yielded an accuracy of 69.94% on validation and 0.2816 log loss. This model had the lowest log loss we have seen out of all of our models. We then tried incorporating SMOTE into our pipeline in an effort to classify observations of the under-represented targets. This model yielded 69.46% accuracy on validation and 0.6900 log loss. Adding SMOTE to the pipeline improved results in comparison to the OVR + RF model alone on the public part of the test data (72.18% to 73.65%).

In addition to these RandomForest models, we also tried training individual RF models trained on data that had been resampled using SMOTE to each unique problem id. The accuracy on validation, however, was very low, 0.573 so it was not submitted.

4. Stack Ensemble Classifier

We tried using a Stack Ensemble using LightGBM and XgBoost chosen from previous experiments for the first layer and Random forest as the final layer. Although the prediction accuracy was decent (72.9% accuracy), the process was computationally intensive and took over 30 minutes to train. One of the reasons we think there wasn't a change in accuracy

was that both LightGBM and XgBoost might be predicting the same results, therefore, combining them did not result in an improvement in performance. The models used for stacking were:

1. LGBM + XgBoost + Random Forest
2. LGBM + Naive Bayes + Multi Layer Perceptron + Random Forest + Logistic Regression

Here is the brief summary of the model results talked about in this section:

Machine Learning Method	**Validation Accuracy**	**Log Loss**	**Leaderboard Accuracy**
LDA + XgBoost Model	61.95%	1.044	67.29%
PCA + XgBoost Model + Target Mean Encoding	70.44%	0.658	71.56%
XgBoost on full dataset	72.07%	0.744	74.02%
RF model	67.39%	0.788	Not submitted
RF + OVR model	69.94%	0.282	72.18%
SMOTE + RF + OVR model	69.46%	0.690	73.65%
21 RF models	57.30%	N/A	Not submitted
Stack Ensemble Classifier(LGBM + XgBoost + Random Forest)	71.04%	0.758	73.9%
Stack Ensemble Classifier(LGBM + Naive Bayes + MLP + RF + Logistic Regression)	73.3%	0.690	Not submitted

Table 1: Results of various machine learning models and feature engineering methods

About the Team

Team members: *Shruti Roy and Audrey Barszcz*

Repo: https://github.com/USF-ML2/project-statistically_significant

Kaggle ID: shrutidroy and audreybarszcz

Team Name: Statistically Significant

Appendix

Additional EDA done by the team:

Understanding the correlation (if present) between different gene variables

We plotted a correlation matrix of a subset of predictors. We noticed that all the predictors in this dataset are independent of each other since there is close to zero correlation between them.

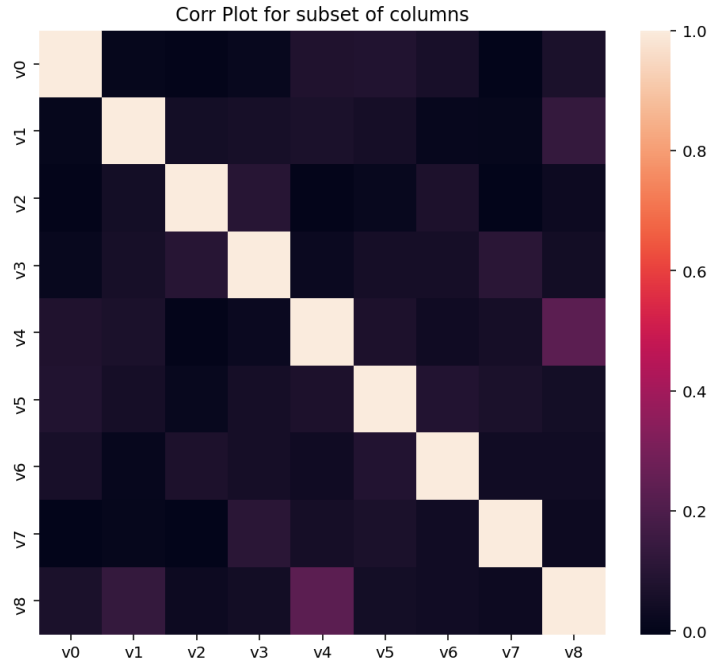

Fig 1.3: Correlation plot between 10 gene expression variables

Dimensionality reduction using PCA

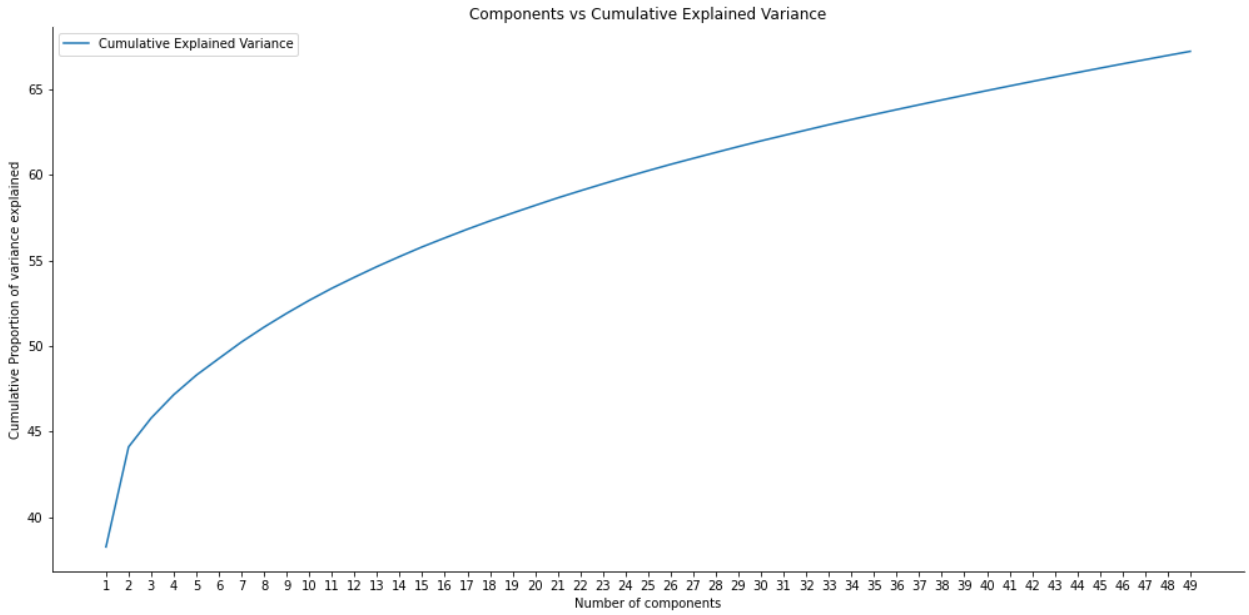

Fig 1.4: Components vs explained variance plot

3. Understanding the representation of target variables in different "Problem Ids"

To look at how different problem IDs (which could be different experiments) correlate with levels of the target, we looked at target level distributions within each problem ID. We found that each problem ID had a different distribution of target levels. Some did not even have all 5 types of cancer observations.

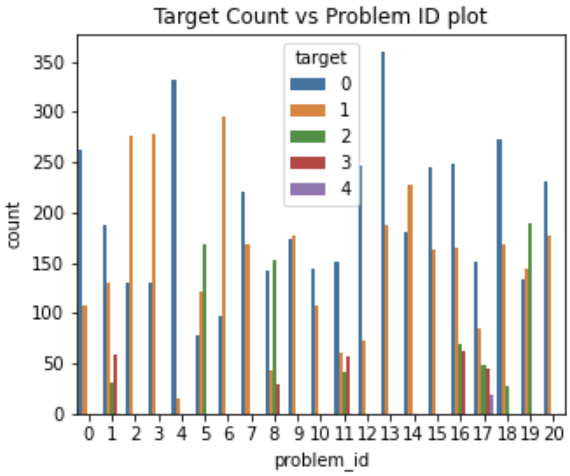

Fig 1.5: Target Distribution within each Problem-id