Detecting Breast Cancer with a Variety of Models

Carlos Vasquez
cvasq24@student.ubc.ca

Steven Mezei
st3v3nm@student.ubc.ca

Abstract

Breast cancer is the most common form of cancer for women in Canada and early detection is extremely important for patients. We looked at a microarray dataset in order to find which machine learning methods are best at detecting whether a cell came from cancerous breast tissue or not. The methods that we used are Naive Bayes, Logistic Regression, K-Means Clustering, Support Vector Machines (SVMs), Ensembles, and Autoencoders for dimensionality reduction. We found that the best performing models were Logistic Regression and Support Vector Machines with both methods having a 94.5% test accuracy.

1 Introduction

Breast cancer is a prevalent and complex disease that calls for better diagnostic and therapeutic approaches. With modern sequencing technologies, researchers can gather gene expression data from breast tissue samples quickly and efficiently. As such, a ubiquitous scenario in breast cancer samples is the identification of differential gene expression in cancerous tissues and non-cancerous ones: a binary classification problem. In turn, this may offer experts valuable insights into the molecular mechanisms underlying breast cancer, which could lead to more efficient precision medicine techniques, biomarker discovery, and novel therapeutic approaches.

In this research paper, we aimed to tackle this problem. We compared a variety of machine learning models that process this high-dimensional cell by gene matrix and determine whether the cell came from a cancerous breast tissue or not. While some of these approaches have already been attempted, we propose a comprehensive review of these algorithms. We will also apply deep latent variable models to examine whether they learn something meaningful, and test these latent representations against the entire data set. Consequently, we hope these representations learn meaningful patterns in the gene expression data for both classes. What distinguishes this study from previous ones is that we have adapted our performance metrics for this specific scenario. Not only do we care about predictive power, but we also care about the consequences of predicting a sample is not cancerous when the true class is cancer. As such, we will use Naive Bayes, Logistic Regression, K-Means Clustering, Support Vector Machines, Random Forests, XGBoost, AdaBoost, and Neural Networks to determine which methods perform the best in this binary classification problem. We also hope that they provide some insight into possible dysregulated biological mechanisms that may be involved in the formation of breast cancer.

2 Related Work

2.1 Neuroevolution as a tool for microarray gene expression pattern identification in cancer research

Neuroevolution is a technique that combines Artificial Neural Networks (ANN) and Evolutionary Computation (EC) "[Grisci, Feltes, and Dorn 2019]". This is derived from biology concepts such as inheritance, random variation and selection and applies them to machine learning. Firstly, genes that presented little difference between them were removed. In order to model evolution, 2 parents are

Course project for UBC CPSC 440/540, April 2023.

chosen and a new input is created based on the genes of the parents. This generates more samples that can be trained in order to fit the model. Both the starting samples, and the samples that were generated by Neuroevolution, were fitted in the model. We want to apply parts of this process into some of the methods that we use, and compare them to other possible ways to analyze a microarray dataset.

2.2 A multi-objective gene clustering algorithm guided by apriori biological knowledge with intensification and diversification strategies

Clustering is a useful technique for microarray datasets that organizes samples that have similar genetic expression "[Parraga-Alava, Dorn, and Inostroza-Ponta 2018]". Some points of concern from using clustering are finding connections that are not actually present but are only found due to random chance. In order to make sure that the connections found through clustering are substantial, external biological knowledge is used. This method is called "multi-objective gene clustering algorithm guided by apriori biological knowledge". We use a K-means clustering algorithm in order to group samples into either having cancer or not having cancer, however we will not be using external biological knowledge to create the clusters.

2.3 Methodology to identify a gene expression signature by merging microarray datasets

A common problem with microarray datasets is that they are high-dimensional and there are a low amount of samples in comparison "[Fajarda et al. 2023]". The methods that were used in this paper to address this concern were to merge multiple datasets before processing the data to increase the sample size. The datasets were pre-processed where a subset of genes are selected that are the most relevant. Once the pre-processing is complete, a supervised machine learning algorithm is used to classify individuals based on gene expressions. We will be using supervised methods such as Neural Networks and Support Vector Machines (SVM), but we are not merging datasets together.

3 Methods

3.1 About the data set

Our dataset comprises RNA extracted from breast tissue samples using the TRIzol or TRI reagent method, quantified using a NanoDrop spectrophotometer, and labeled using microarray-based gene expression analysis, a common method for measuring gene expression levels in large gene sets simultaneously. The data processing steps used, such as background-correction and normalization, are standard techniques used to remove systematic biases and variability in the gene expression data. Removing non-uniform and below-background features helps to ensure the accuracy of the gene expression measurements. Overall, this data set provides valuable information that can be used to study the molecular mechanisms of breast cancer and identify potential therapeutic targets.

The data set is initially split into X_{train}, X_{test}, y_{train}, y_{test} (216 samples in training set, 73 in test set), stored and then used for every model. This ensured that all models use the same samples for each set, and that the same class proportions were present throughout the study. We let non-cancer samples be class 1, and class 0 for cancer samples. In the training set, the proportion of the cancer class was 0.505, while the non-cancer class accounted for 0.495, so they were balanced. In test set, the ratios were 0.466 and 0.534, respectively.

3.2 Training, Assessment, & Prediction for Supervised Models

The training procedure of most models can be summarized in a few steps; we chose this methodology because it is standard in assessing various models and optimizing their hyper-parameters. We will note if we approached the model in a different fashion, but generally:

1. If applicable, choose a range of candidates for the model's hyper-parameters.
2. In the case where it is easy to iterate through these potential hyper-parameters, we simply iterate through the list and perform 5-fold cross validation on X_{train} and y_{train} for each hyper-parameter.

3. We chose the hyper-parameter that yielded the highest mean score across the 5 folds. This is the model we used on the test data.
4. Assess/score the model on X_{test} and y_{test}.
5. Performance Metrics: Receiver Operating Characteristic (ROC) curved and Confusion Matrices.

Note that we used ROC curves because they visualize the trade off between the true positive rate and the false positive rate at different classification probability thresholds. As such, the Area Under the Curve (AUC) can be used to assess a classifier's performance. Then, as we briefly discussed in the introduction, we used confusion matrices because they show the number of true positives, false positives, true negatives, and false negatives. Since we aim to suggest models for breast cancer prediction, we truly care about the cost of predicting "not cancer" when the true class is "cancer" (false positive), as this might have devastating consequences in real-life scenarios. Confusion matrices help us in this objective as we will consider which models had the least number of this metric.

Now, we start our methods with one of the simplest models, Naive Bayes.

3.3 Naive Bayes

Based on the dimensionality of the dataset, we need to compress the dataset into less variables so that we can perform Naive Bayes at a computationally reasonable runtime. We did this by using Principal Componenent Analysis (PCA) and reducing the data set down to only 50 components. Note that we chose 50 components in an arbitrary manner, we simply wanted to h

We also turned the data set from having continuous features into binary features. We took the mean value for each dimension and stored it in an array. We then go through each sample and each dimension and check if the value is greater than the mean. We then assume that each variable is independent of each other.

We then calculate the parameters for each variable to see the probability of normal and probability of breast cancer. Once this is fitted, we predict by multiplying all of the probabilities for each variable and see if the probability of cancer is higher than the probability of not having cancer.

3.4 Logistic Regression

Logistic Regression works well with this problem as the gene data is continuous and whether a sample has breast cancer is binary. We used 5 fold cross validation in order to fit a logistic regression with all of the genes present. This utilises the softmax function in order to find the probability that a patient has breast cancer.

3.5 K-Means Clustering

This is an unsupervised learning model that fits the data into k clusters. We will use an even number of clusters because there are only 2 possible types in the data, breast cancer and no breast cancer. For each value of k, we perform 5 fold cross validation, these are the values of k that we used.

$$2, 4, 6, 8, 10, 12, 14$$

For each cluster, we will find the mode between cancer and no cancer. In order to predict breast cancer or not, the sample will be assigned to a cluster, and based on the mode of that cluster, that will determine whether there is cancer or not. We thought these values were reasonable, as adding more clusters might end up causing overfitting. It is found that k = 2 is the best performing in the cross-validation, so we used 2 clusters to fit our k-means model.

3.6 Support Vector Machines

The main hyper-parameter for SVMs is the kernel type, which we decided to be linear. Here we're relying on the assumption that the data is linearly separable. Again, this may be a point of discussion in later sections.

3.7 Ensembles: Random Forests, AdaBoost, XGBoost

First, the hyper-parameter to determine for Random Forests is the depth of the forest. The deeper the model, the more complex it is because it considers more features of the data set, leading to overfitting. As such, we wanted a good trade-off between accuracy and depth. The list of hyper-parameters consisted of depths of:

$$1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 50.$$

While this may seem like a considerably large set of depths to consider, note that the data set contains $35,981$ genes, so these depths account for at most 14% of the original dimension. We thought this was reasonable, though it was entirely subjective and is a potential source of bias/overfitting.

Second, the hyper-parameter for AdaBoost is the number of weak learners. The more weak learners, the more likely we are to overfit to the training data. As such, we considered:

$$1, 5, 15, 25, 50, 75.$$

Since we aren't optimizing a considerably high number of weak learners, we figured that this set wouldn't overfit the data too much.

Since XGBoost is a more sophisticated model, there are more hyper-parameters to tune. We considered:

Table 1: Hyper-parameters considered for training XGBoost

Hyper-parameter	Values that were tested
Learning rate	$0.1, 0.01, 0.001$
Depth	$5, 10, 15, 20, 25, 30$
# of Estimators	$100, 200, 300, 400, 500$
Subsample Ratio of Examples	$0.5, 0.7, 0.9$
Subsampling Ratio of Features per Tree	$0.5, 0.7, 0.9$
α (L1 regularizer on weights)	$0, 0.1, 0.5, 1$
λ (L2 regularizer on weights)	$0, 0.1, 0.5, 1$

Evidently, there's too many possible combinations of these hyper-parameters, so our previous steps are not feasible. As such, we resorted to Randomized Search, which randomly samples subsets of hyper-parameters and evaluates them on the model. It outputs the best set of hyper-parameters it fings. Finally, we tested the optimal set of hyper-parameters on the test data and carried out the same metrics evaluation from before.

3.8 Neural Networks: Autoencoders

We used an unsupervised learning approach for this last portion of the study to learn whether lower-dimensional representations of the data set efficiently capture gene behavior. We trained a deep autoencoder with 6 layers that reduced the 35,981 genes to 500, then 250, then 2, and ultimately reconstructed it back to the original dimension using the same sequence. By minimizing the reconstruction loss (mean squared error between original sample and the reconstruction), using a learning rate of 0.001, batch size of 1 and a weight decay of 0.001, we trained the autoencoder from scratch for 10,000, 20,000, 30,000, and 40,000 iterations. We passed the entire data set to the encoder, and re-ran all the previous models on these 2-dimensional latent representations, to see if we get better results.

4 Results

As we've learned in lecture, Naive Bayes makes a strong set of assumptions regarding the distribution of features; specifically, it assumes that each feature is independent of all the others, given the class. We know that this isn't particularly true in our genetic context: high levels of a gene may induce high levels of another gene through enhancers, or the opposite can happen through silencer sequences. Thus, we expected to obtain relatively negative results with the Naive Bayes. Performance metrics are shown below in Table 2.

Table 2: Performance metrics for Naive Bayes model

Model	Test Accuracy	AUC	False Positives
Naive Bayes	75.3%	0.747	14

Although the test set accuracy is not too unsatisfactory, it is necessary that we obtained better results in order for models to be feasible in a clinical setting. This suggested the use of models that are able to use features together, unlike Naive Bayes. Logistic Regression is one of the first options we think of when we think about binary classification. Table 3 demonstrates its performance.

Table 3: Performance metrics for Logistic Regression model

Model	Test Accuracy	AUC	False Positives
Logistic Regression	94.5%	0.978	3

We immediately obtain better performance. Next, we used another model that also uses linear boundaries; the objective here was to confirm that the data set was indeed linearly separable. SVMs exhibited even better performance:

Table 4: Performance metrics for SVM model

Model	Test Accuracy	AUC	False Positives
SVM	94.4%	0.975	2

This is probably hard to beat, yet we were curious about testing the performance of ensemble models: i.e., what happens when we combined various small, simple models together. Shown below are the results for the three ensemble methods discussed in the previous section:

Table 5: Performance metrics for Ensembles

Model	Test Accuracy	AUC	False Positives
Random Forests	83.6%	0.937	4
AdaBoost	89.0%	0.971	4
XGBoost	90.4%	0.977	3

Then, we used K-means for clustering, so that we're able to determine whether there are noticeable clusters, which may indicate potential differential genes:

Table 6: Performance metrics for K-means

Model	Test Accuracy	AUC	False Positives
K-Means Clustering	83.6%	0.836	14

Figure 1 below shows the ROC curves for all the above models.

Finally, we wanted to see whether autoencoders learned any meaningful representations of the data. In this case, we don't just care about the reconstruction loss, but we also care about how well the autoencoder's latent representations perform in the previous supervised learning algorithms. We started with the Ensemble methods. Table 7 shows the supervised models' performances using the latent representations for different autoencoders trained with various numbers of iterations. Please refer to Appendix A for a 2D-visualization of the latent variables in these models.

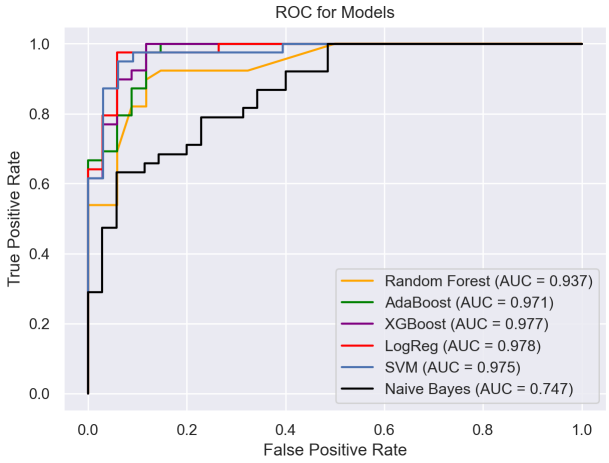

Figure 1: ROC curves for supervised learning models.

Table 7: Performance metrics for autoencoders across different training iterations

Autoencoder	Reconstruction Error	RF AUC	AdaBoost AUC	XGBoost AUC
10k iterations	40.620	0.531	0.537	0.500
20k iterations	23.480	0.793	0.782	0.671
30k iterations	13.7907	0.868	0.783	0.878
40k iterations	8.3201	0.643	0.697	0.720

5 Discussion and Future Work

In terms of all metrics, the SVMs performed the best. Naive Bayes is the worst performing model which is expected because turning the continuous gene data into a binary variable leads to a substantial amount of information being lost. K-Means clustering also had a problem with many false positives. This may be due to not reducing the dimensionality of the sample when we fit the model, or we have to do more pre-processing in order for this model to perform well. The best performing models were SVMs, Logistic Regression and XGBoost. These results are not surprising as these models are designed for classification problems with a continuous dataset.

The autoencoder trained on 30k iterations clearly outperforms all the other autoencoders, across all metrics. What's most interesting is that it even beats the 40k autoencoder. We expected autoencoders trained with more iterations to increasingly perform better in the Ensembles, yet performance peaked at 30k iterations. This indicates that after some threshold of iterations, we lose meaningfulness in the latent variables, so they no longer encapsulate gene expression behaviour, and are instead more interested in reconstructing a sample.

We chose a 2-dimensional bottleneck layer so that we can visualize the latents, but this is perhaps too restrictive and the reason why the AUCs are relatively low compared to the initial ones. As such, we could also expand the dimensions of the bottleneck layer to 3 (so that we can still visualize the latent space), or even more, but at the expense of losing interpretability. Nevertheless, the PCA projection (Figure 6 - Appendix A) shows less separation between both groups, suggesting that the autoencoders are actually learning informative features.

In future work, we plan to pre-process the data by selecting a subset of genes that are important before we fit our model. We also plan to increase the sample size by either merging other microarray datasets together or by using generative techniques such as Neuroevolution. We will also like to include methods such as Bayesian Logistic Regression and Convolutional Neural Networks (CNN).

References

Fajarda, Olga, João Rafael Almeida, Sara Duarte-Pereira, Raquel M. Silva, and José Luís Oliveira (2023). "Methodology to identify a gene expression signature by merging microarray datasets." *Computers in Biology and Medicine* 159, p. 106867. ISSN: 0010-4825. DOI: https://doi.org/10.1016/j.compbiomed.2023.106867. URL: https://www.sciencedirect.com/science/article/pii/S0010482523003323.

Grisci, Bruno Iochins, Bruno César Feltes, and Marcio Dorn (2019). "Neuroevolution as a tool for microarray gene expression pattern identification in cancer research." *Journal of Biomedical Informatics* 89, pp. 122–133. ISSN: 1532-0464. DOI: https://doi.org/10.1016/j.jbi.2018.11.013. URL: https://www.sciencedirect.com/science/article/pii/S1532046418302260.

Parraga-Alava, Jorge, Marcio Dorn, and Mario Inostroza-Ponta (2018). "A multi-objective gene clustering algorithm guided by apriori biological knowledge with intensification and diversification strategies." *BioData Mining* 11, p. 16. ISSN: 1756-0381. DOI: https://doi.org/10.1186/s13040-018-0178-4. URL: https://biodatamining.biomedcentral.com/articles/10.1186/s13040-018-0178-4.

A Latent Representations

Shown here are the latent representations of the various autoencoders:

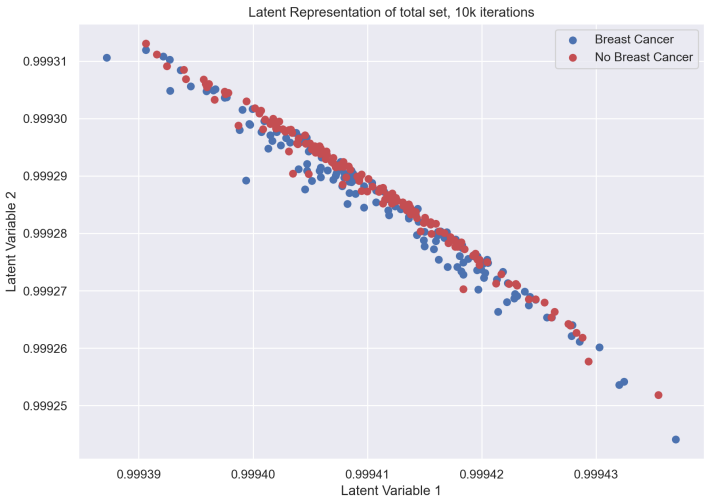

Figure 2: 10k autoencoder latent space

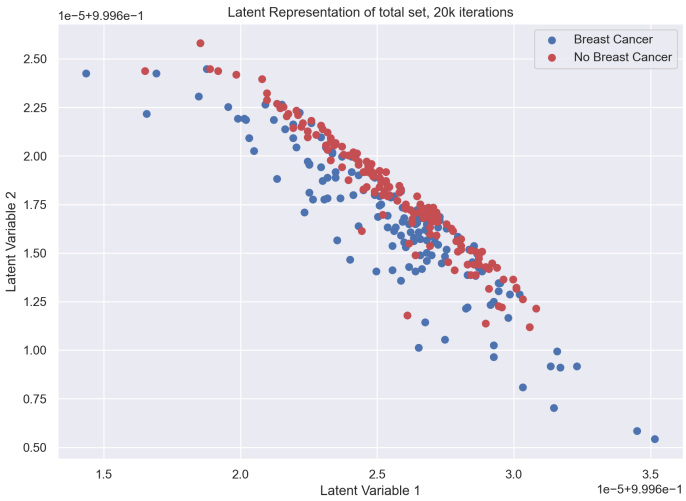

Figure 3: 20k autoencoder latent space

Figure 4: 30k autoencoder latent space

Figure 5: 40k autoencoder latent space

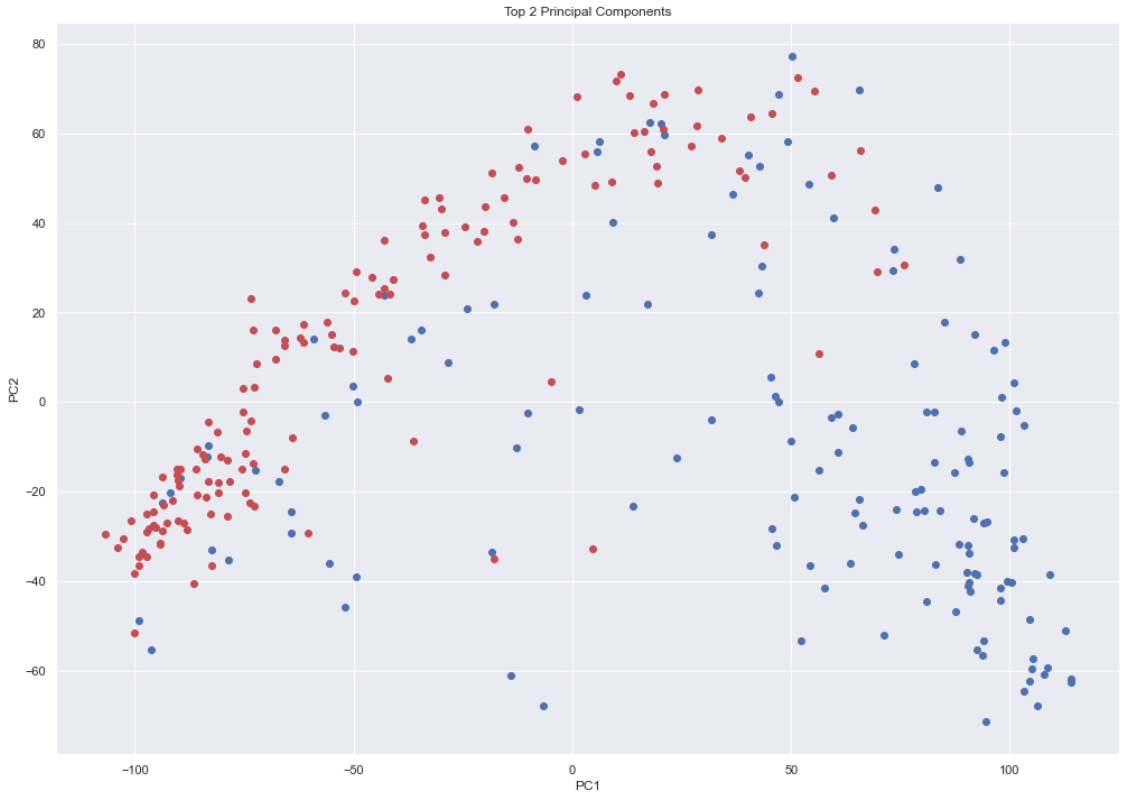

Figure 6: PCA space

B Confusion Matrices

We now show the confusion matrices for the models, from which we obtained the number of false positives from:

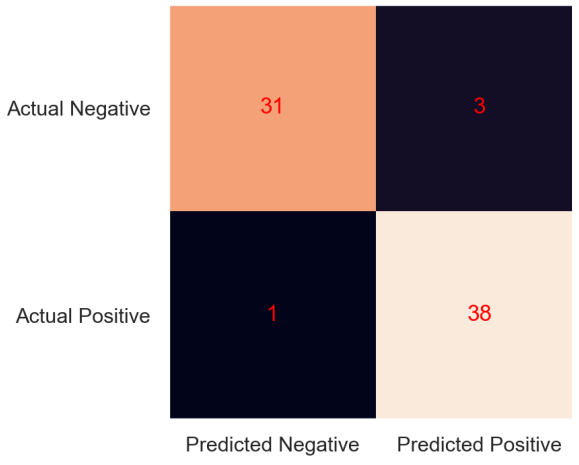

Figure 7: Log Reg Confusion Matrix

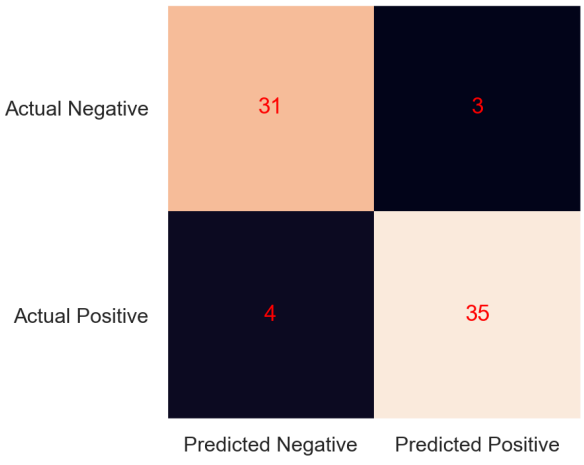

Figure 8: XGBoost Confusion Matrix

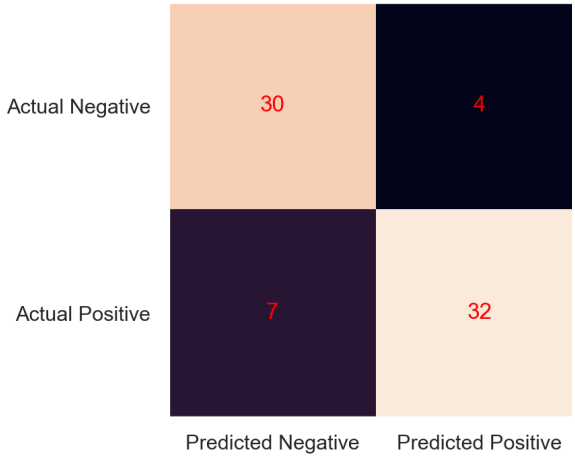

Figure 9: Random Forests Confusion Matrix

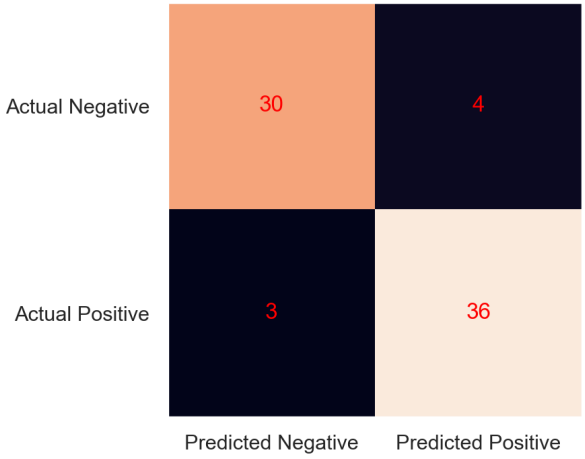

Figure 10: AdaBoost Confusion Matrix

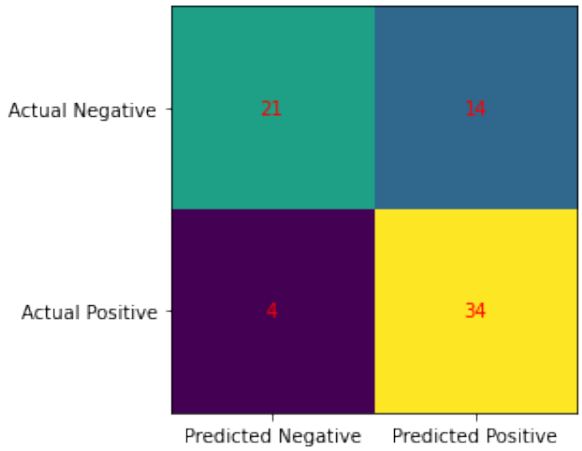

Figure 11: Naive Bayes Confusion Matrix

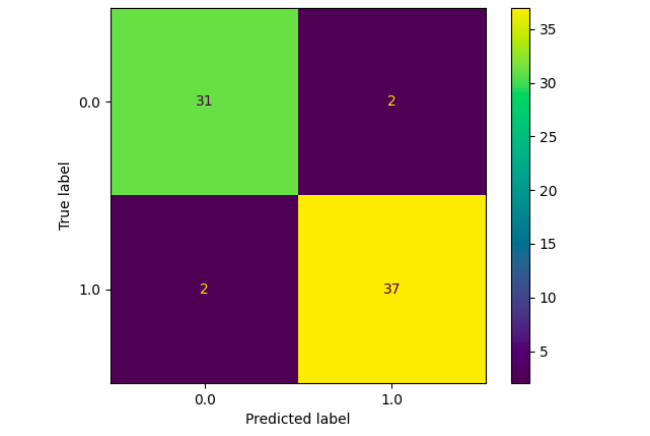

Figure 12: SVMs Confusion Matrix