(2023), 37, 1–8

CAMBRIDGE
UNIVERSITY PRESS

RESEARCH PAPER

Using Single Cell RNA Sequencing to Identify High-Risk Breast Cancer Oncogenes

Swati Negi[*†] and Alexander Rupp-Coppi[*†‡]

†Columbia University
‡Joint first authors
*Corresponding author. Email: swati.negi@columbia.edu; amr2311@columbia.edu

Abstract
Breast cancer is a malignant disease that affects 1 in 8 women. It is heterogeneous in nature and is attributed to genetic and environmental factors. With the advent of single-cell RNA (sCRNA) sequencing techniques, it is now possible to understand the biological behaviors of breast cancer such as cell heterogeneity, drug resistance, and breast cancer metastasis. We present a study that uses scRNA data to learn more about how the cancer metastasizes. We use machine learning models to build a classifier that predicts whether a patient is at risk of metastasis. Our best model was able to predict it accurately in 90% of cases and we conclude by identifying genes significant to prediction of breast cancer metastasis.

Keywords: rna sequencing, oncogenes, breast cancer metastasis

1. INTRODUCTION

Breast cancer is the most commonly diagnosed cancer among women worldwide and is a leading cause of cancer-related deaths. Metastasis, the spread of cancer cells from the primary tumor to other parts of the body, is responsible for the majority of breast cancer deaths. Thus, understanding the molecular mechanisms underlying breast cancer metastasis is critical for developing effective treatments.

Recent advances in single-cell RNA sequencing (scRNA-seq) technology have enabled the identification of molecular subtypes of breast cancer and the characterization of cellular heterogeneity within tumors. However, analyzing scRNA-seq data to identify biomarkers associated with metastasis is a challenging task due to the complexity of the data and the large number of variables.

Machine learning algorithms have emerged as powerful tools for analyzing large and complex biological datasets, including scRNA-seq data. In this study, we used a machine learning classifier to predict the risk of breast cancer metastasis using scRNA-seq data from the Genomic Data Commons (GDC). Our classifier uses a combination of supervised and unsupervised learning methods to identify molecular features that are predictive of metastasis.

2. METHODS

2.1 Sourcing Carcinoma Transcriptome Data

Breast carcinoma scRNA-seq transcriptome data sampled from 1,082 individuals were identified via a filtered search of publicly-accessible files from The Cancer Genome Atlas Program (TCGA), hosted on the National Cancer Institute (NCI)'s Genomic Data Commons (GDC) Data Portal. (For the exact query URL, a link to the GDC manifest files used for download, and a note about running GDC's data transfer utility on non-officially supported Apple ARM chipsets, please see the 'Online Resources' section of the appendix.)

Because of hardware compatibility difficulties encountered while running GDC's data transfer tool, download of the aggregate sample data was somewhat lossy. Only 1,110 distinct transcriptome GENCODE v36 augmented STAR gene count files were ultimately successfully downloaded; but because certain individuals within the cohort had multiple carcinomas sequenced, the final transcriptome file count still exceeded the total number of individuals within the sample population.

Transcriptome data was downloaded in two separate batches, separated by vital statuses of the sampled patients: the first batch consisted only of transcriptomes originating from individuals still alive at the time of sequencing (total count of 900 transcriptomes), while the second batch consisted of transcriptomes originating from individuals whose tumors were sequenced post-death (count of 110).

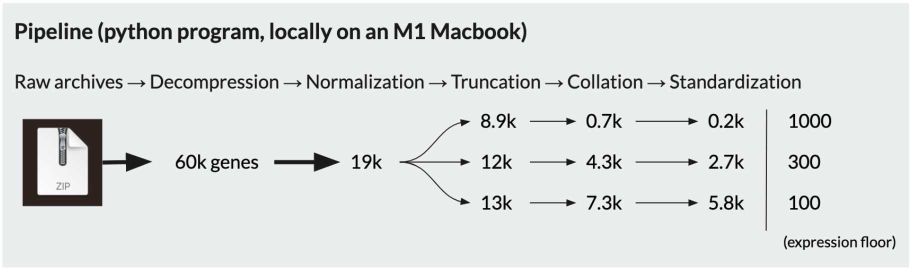
Pipeline (python program, locally on an M1 Macbook)
Raw archives → Decompression → Normalization → Truncation → Collation → Standardization
ZIP
60k genes
19k
8.9k
0.7k
0.2k
12k
4.3k
2.7k
13k
7.3k
5.8k
1000
300
100
(expression floor)

Figure 1. Preprocessing consisted of a sequence of independent, modular processing steps.

2.2 *Preprocessing Pipeline (Feature Gene Selection)*

2.2.1 *Overview*

The 1,110 raw STAR count transcriptome files were iteratively condensed and ultimately aggregated into just a handful of comma-separated value (CSV) tables by running a series of modular python scripts sequentially.

The general approach to each module is for the user to provide two input directories containing unprocessed data, one with that of the survivor cohort and the other with that of the deceased cohort; and to provide two output directories for processed data, also corresponding to survivors and deceased, respectively. Some, but not all modules have additional user-configurable parameters. The user then executes each module from the command line; modules are independent, and so their execution order is not fixed. The one constraint on ordering is data shape: files within an input directory must match the format expected by a specific module. (This flexibility with module ordering ended up being quite important while tuning feature gene reduction.)

The birds'-eye view of the final pipeline ordering is this: the raw STAR count files are condensed and normalized into one-to-one corresponding individual sample CSVs; those samples are then selectively truncated to reduce gene feature count; and then finally, the many individual sample CSVs are collated into just a handful of large tables, where each row is a sample, and the many columns are gene features.

As a consequence of variation in expression among genes generally between the survivor and deceased subpopulations, the collated tables resulting from the filtering process do not necessarily share all feature genes in common; in one final post-processing step, genes not shared between tables are therefore dropped via set intersection.

2.2.2 Example module execution

Commands chain to run the entire pipeline in one go, which takes a couple of minutes on an M1 Macbook:

```
   python preprocessing.py && python trimming.py && python collating.py
&& python cross_collating.py
```

Re-running just the truncation step on already pre-processed data:

```
python trimming.py
```

Re-collating and then set-intersecting to yield final feature tables:

```
python collating.py && python cross_collating.py
```

The truncation step can be skipped entirely if within the collating.py script the user alters the input directories to target raw preprocessed data instead of the post-truncated set:

```
   DATA_SURVIVORS_PATH = '../data/trimmed/survivors' => '../data/processed/survivors'
DATA_DECEASED_PATH = '../data/trimmed/deceased' => '../data/processed/deceased'
```

2.2.3 Initial Processing Pass (preprocessing.py)

This step discards all genes within a sample which have an unstranded expression count of zero, and all genes labeled as anything but protein-coding. Individual gene expression counts greater than zero are normalized by the total unstranded expression count across all genes within a sample. (Because the classifier trained ultimately depends only on expression level, it was decided that using unstranded counts, while less accurate than using stranded counts, was adequately sufficient for this use case.)

The 60,000 genes from the raw STAR count files are reduced to just 20,000, and the column count per sample file is reduced from eight to three: gene name, unstranded count, and normalized count.

Genes are listed within the output CSV in descending order of expression level.

An example processed file:

```
    data origin: '../data/decompressed/deceased/8256e097-4476...
19437 coding genes total
9942608277 total expressions counted
gene name,unstranded count,normalized_count
CPB1,1156203,0.000116287694
COL1A1,903009,9.082214e-05
COL1A2,675157,6.790542e-05
FN1,598832,6.0228864e-05
...
```

2.2.4 Truncation Step (trimming.py)

The samples output from the initial processing pass contain 20,000 distinct feature genes; many of those genes are not useful for training a tumor classifier, because they do not impact cancer-related pathways.

Manual inspection of just a couple samples shows that those genes expressed most highly are most often structural (e.g. collagen-related) or mitochondrial; at the other end of the expression distribution, a great many genes have expression counts of exactly one, suggesting that their presence is noise.

It was therefore decided to reduce feature genes to just those near the middle of the expression distribution for a given sample. Specifically, genes beneath a user-defined, non-normalized expression count threshold, and at a non-normalized expression count beyond a user-defined number of standard deviations from the distribution's median expression were rejected from the final feature set.

The researcher-chosen target feature count was in the low-mid thousands; via trial and error, it was identified that a window of no more than 50 deviations and an expression count floor of between 100 and 300 yielded post-rejection sets of 10,000 genes. Of those genes, median normalized expression count per gene was about 0.005%, within a range of 0.0006% to 0.15% of total expression.

(One of the genes ultimately identified by our classifier as most significant for correlating expression with tumor lethality, KLF10, averaged an expression level of 0.03% in non-lethal tumors and 0.05% in lethal tumors.)

The final feature set count was particularly sensitive to the given expression floor: floors higher than 1000 shrink the gene set to just a couple hundred, indicating that a majority of genes within the defined acceptable deviation from median had considerably low expression level, possibly suggesting that they were noise, and that the final feature set used for training could in fact have been even smaller without losing significant accuracy.

2.2.5 Collation and Set Intersection (collating.py and cross_collating.py)

Following sample truncation, individual sample files needed to be collated into many-sample data frames to be useful for training.

Initial collation of the survivor and deceased tables separately resulted in data frames that in practice only shared about half of all feature genes, which is unusable for model training.

The researchers suspect but have not rigorously statistically validated that because transcriptome expression data generally follows a non-normal, negative binomial distribution; because expression variance, even within a constrained window around the sample median, varies on the order of hundreds of counts to tens of thousands; and because gene expression profiles between lethal and non-lethal tumors ultimately turned out to be quite distinct, the sets of genes which passed truncation for the survivor and deceased cohorts, respectively, are in fact quite different.

In practice, if the rejection window was set to only two or three deviations from median, the post-truncation collations in fact shared no common genes at all.

Therefore, to meet the goal feature count of 5,000, the truncation step was tuned to yield 10,000 distinct features per subpopulation, so that following a simple intersection of subpopulation sets, a satisfactory number of genes would be found in common, adequate for training.

Ultimately, by choosing two expression floors (100 and 300), and keeping deviation count high (count=50), the STAR count processing pipeline yielded two feature sets which were used for model training, one of 6000 genes (floor=100), and one of 3000 (floor=300).

2.3 Machine Learning Models

The dataset had class imbalance (900 survived, 210 deceased patients) and logistic regression with an F1 score of 76% was able to predict positive samples fairly accurately, but no negative samples were predicted at all (See Correlation matrix for Logistic Regression, fig 2). We therefore had to consider using sampling techniques to level the classes. After much experimenting, we found SMOTE performed best since it creates artificial samples of minority class instead of duplicating the samples.

With balanced data, we trained and evaluated three different machine learning models on our scRNA-seq data using the scikit-learn library in Python: logistic regression, decision trees, and random forest.

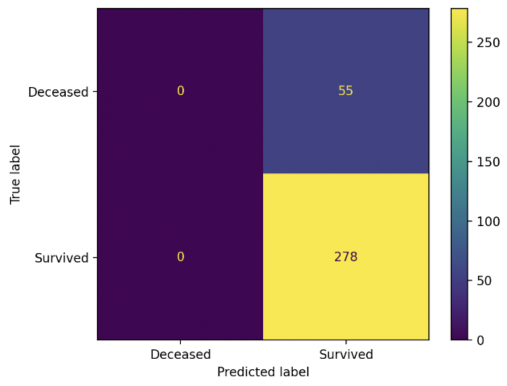

Figure 2. Fixing class imbalance with SMOTE

2.4 Model Evaluation

We used F1 score and AUC score to evaluate the models. Accuracy, although a good measure, is not preferred in clinical data where class imbalance is common and the model can predict positive samples correctly but very few negative samples are predicted accurately.

3. RESULTS

Our results show that random forest performed the best in predicting the risk of breast cancer metastasis using scRNA-seq data. When using 3,000 gene set, the random forest model achieved as f1 score of 0.9290, and with the 6,000 gene set, it achieved an f1 score of 0.9392. The logistic regression model had the lowest performance, with an f1 score of 0.7634 and 0.7848 for the 3,000 and 6,000 gene sets, respectively. The decision tree model had intermediate performance, with an f1 score of 0.8387 and 0.8076 for the 3,000 and 6,000 gene sets, respectively.

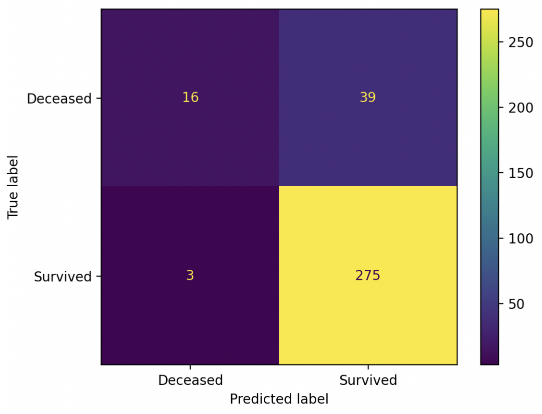

Figure 3. Confusion Matrix for Random Forest using 6k genes

When comparing the models using AUC score, we had random forest again giving the best AUC score of 0.7170 with 6,000 genes and 0.6951 score with 3,000 genes.

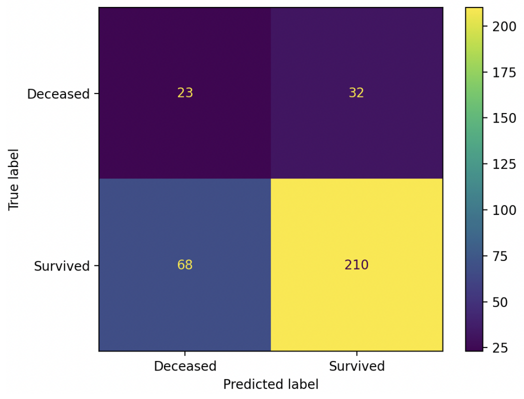

Figure 4. Confusion Matrix for Decision Tree Model using 6k genes

Decision Tree and Logistic Regression both had an insignificant ROC score ranging between 0.53 - 0.65.

4. DISCUSSION

The classifier we've trained has limited but significant use for correlating a sequenced breast carcinoma's gene expression profile with a likelihood of lethality, which in turn can be used as a proxy indicator for informing the stage of severity, and in particular the likelihood of metastasis, of a patient's cancer.

Per the confusion matrices in figures 3 and 4, our classifier cannot with confidence claim a patient's given carcinoma to be likely lethal, but with a high degree of certainty *can* claim the opposite, that a patient's cancer, based on its current transcriptome profile, is likely still survivable. In clinical settings, such an indication would likely have significant impact on determining therapeutic care regimens; of less medical relevance but of great importance to patient emotional well-being, having knowledge that such a dangerous disease is not yet past the point of no return could be a source of great hope.

More generally, our study demonstrates that machine learning approaches, particularly random forest classification, do have utility in predicting breast cancer outcomes. Additionally, the relative feature importances determined by our model could possibly be useful for identifying novel polygenic mechanisms of action in lethal breast cancer, a promising avenue for future research.

Acknowledgement

We thank Dr. Itsik Pe'er of Columbia University, and also Gauri Anand, MS CGC, for their feedback and suggestions throughout the duration of this project.

The following two papers were instrumentally informative for learning about current work on single cell RNA sequencing for breast cancer:

1. Single cell RNA sequencing for breast cancer: present and future [link]

2. Single-cell RNA sequencing reveals cell heterogeneity and transcriptome profile of breast cancer lymph node metastasis [link]

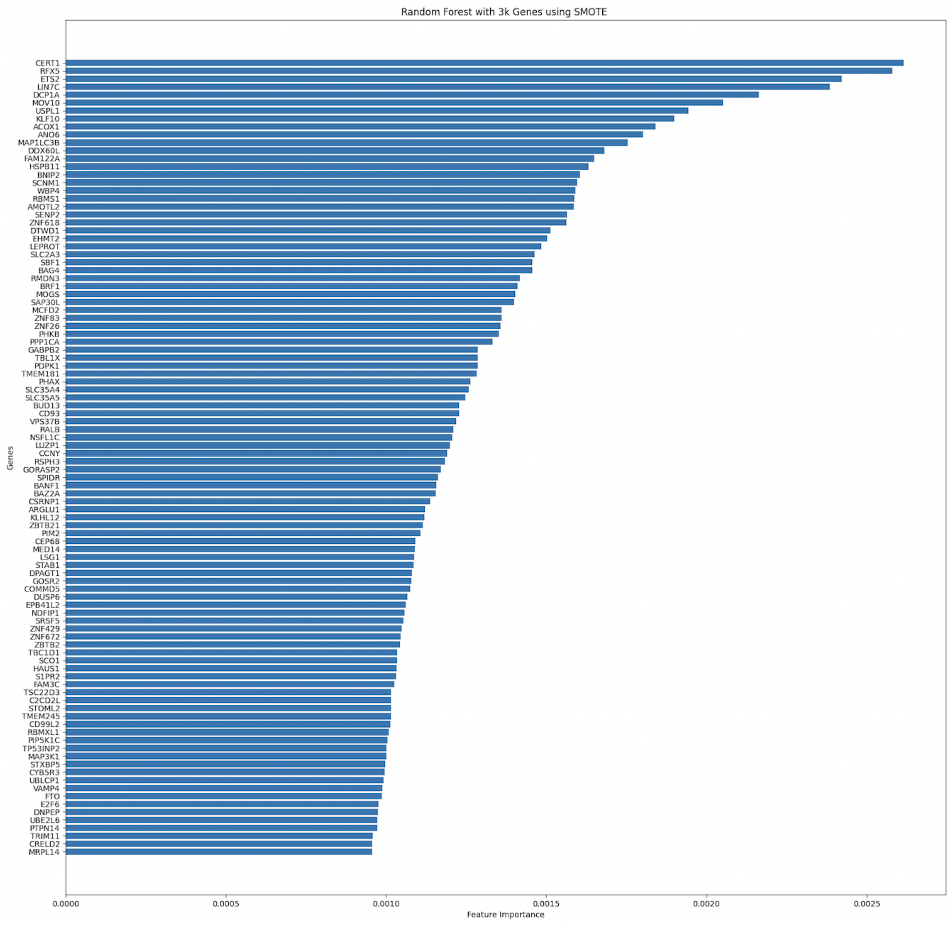

Figure 5. Feature Importance given by Random Forest using 3k genes

5. Appendix

5.1 Online Resources

Preprocessing pipeline and model-training source code:
https://github.com/rcoppy/oncogene-analysis

Compressed archives of training data:
https://github.com/rcoppy/oncogene-analysis/tree/main/data/archives

Link to the original query used to discover the sample cohort: GDC Website

Manifest files for downloading sample data from the GDC platform directly:
https://github.com/rcoppy/oncogene-analysis/tree/main/data/gdc-manifests

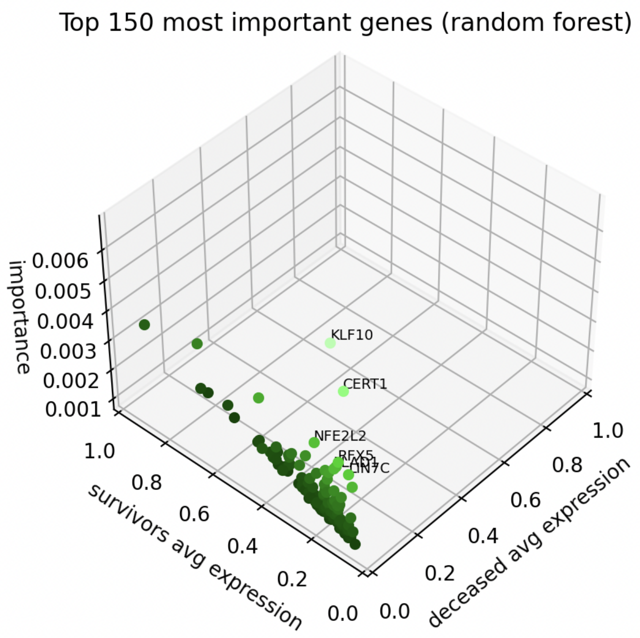

Figure 6. Plotting survivor vs deceased average gene expression in terms of classifier importance.

5.2 A note on GDC manifest files and data transfer tool incompatibility with Apple ARM computers

The linked-to manifest files can be used with the GDC Data Transfer Tool utility to download the 1110 transcriptome samples programmatically, instead of one-by-one; the samples altogether take up 4.6 GB of disk space. Using the transfer tool is recommended over manual download.

Unfortunately, the transfer tool doesn't officially support Apple's ARM-architecture chipsets; the utility can still be acceptably executed (albeit with stability issues) on M1 machines via the UTM virtual machine running an ARM distribution of Ubuntu with the Box64 compatibility program installed. Using this approach, we were able to download 90% of the 1,100 transcriptome samples, which was enough for our data-modeling purposes.