Predicting Cell Phenotypes from scRNAseq Data

Ryan Nayebi*
Mentor: Brianna Chrisman

*Woodside Priory School

Abstract

The use of machine learning to analyze data obtained through single-cell RNA sequencing (scRNAseq) provides eye-opening insights into cellular biology. Applications of this type of data analysis range from understanding tumor heterogeneity (with potential for creating life-saving, patient-specific treatment) to studying stochastic gene expression. In my research, I analyze the performance of a series of supervised learning models on three sets of scRNAseq data. Each model's goal is to predict cell type based on expression levels of different genes within each cell. By training these models on our data I discover correlations between expression levels of specific genes and their cellular phenotypes. Cell-type classification is highly relevant in areas within the medical industry. For example, classifying healthy versus cancerous cells can be used in detecting an early onset of cancer or detecting how far a tumor has spread. Additionally, my research is highly relevant for educational purposes. It provides students in both biology and computer science a real-world application of the central dogma of molecular biology and fundamentals of machine learning (ie. dimensionality reduction, cross-validation, grid searching, supervised learning models, confusion matrices, etc.). In the end, I discovered which supervised learning methods (including their specific hyperparameters), performed the best for differentiating cell types based on gene expression on each dataset. My models and analysis have implications for establishing a framework for future research in cell-type classification, and can be found in my project repository at https://github.com/rnayebi21/PolygenceMLResearch-scRNAseq

Background

The goal for my research project was to create a machine learning classifier that can differentiate cell types based on gene expression in order to be used for practical purposes.

Gene expression contains far more information about a cell compared to a phenotypic observation. The ability to observe and make meaning of this genetic information (at the single cell level) opens a new lens into cellular biology; one which may help detect an early onset of a specific disease or perhaps to confirm or deny a diagnosis. For example being able to detect a cell as malignant or benign can inform doctors if they need to immediately try localizing the cancer. Furthermore, the ability to detect the specific attributes in areas of a malignant tumor can allow doctors to provide treatment specific to a patient's needs.

The use of this classifier can also be extended for educational purposes for both biology and machine learning. For example, seeing how different levels of gene expression may correlate with specific types of cells, can help students gain a better and more real-world understanding of the central dogma of molecular biology. Additionally, this project showcases many different methods of supervised learning, and may be a good starting point for seeing the practical side of the theory taught in many machine learning classes.

Gene Expression

Every cell in our body generally contains the same set of genetic information (genome) for encoding the functional behavior of a cell. Despite having the same information, our cells perform very different functions. To accomplish this, each cell only reads, or expresses, specific subsets of the entire genome. For example, cardiac pacemaker cells (within the heart) would not use the subset of the genetic information that helps create the proteins exclusive to foveolar cells (within the stomach). During transcription, an enzyme called RNA polymerase *transcribes* a section of the DNA to create messenger RNA (mRNA). Ribosomes then use this mRNA to create proteins in a process known as translation. These proteins are what perform cellular processes and functions. Therefore, because pacemaker cells do not need to create proteins that secrete mucus for stomach protection, they do not transcribe, or *express* that subsection of the genome. These irrelevant sections of the genome are suppressed in many different ways in different organisms, including by RNA interference (RNAi)[14] and shifts in chromatin structure caused by regulatory proteins (repressors).

Single Cell Sequencing

Although the cells generally have the same set of genetic information, new errors frequently arise in the genome every time a cell undergoes mitosis. This is especially true for malignant tumors. Harsh tumors often have a high genetic variance, allowing the tumor to mutate and have a better chance of withstanding and surviving treatment. Assuming all cells have the **exact** same genome is erroneous, which is why looking at cells at their individual level is highly important.

Labelling a group of cells as a collective whole, where each cell has the same general function, rather than observing individual cells that make up a functioning unit, obscures important details only observable at the single cell level. For example, it is difficult to see if only specific cells within the unit show a specific phenotype or if all the cells within the unit show a phenotype. A collection of cells may be diverse and heterogeneous, however this is not obvious without single cell sequencing. Single cell sequencing also allows us to identify cells based on their RNA. While cells might have the exact same set of DNA, RNA levels differ greatly from cell to cell. Furthermore, DNA does not inform us about which genes are being expressed or not.

In order to look at specific cells within a group of similar cells, researchers must break bonds between cells while keeping them alive. Researches use special indicators to mark the groups of cells as healthy (and thus viable) and unhealthy. After choosing the healthy cells, they are then

isolated and lysed. This process and its details vary depending on which technique is implemented. Some of the most popular single-cell isolation methods include: fluorescence activated cell sorting (FACS)[1,2], microfluidic technology[1,3], and microdroplet-based fluidynamics[1,4]. After single cell isolation, the genetic material is often isolated and manipulated into complementary DNA (cDNA) via reverse transcription[7]. cDNA is much more stable and easy to work with compared to RNA. Note that cDNA still retains the same information as RNA. During this reverse transcription process often DNA sequences, called genetic barcodes, are attached so tracing the genetic information back to its original cell is much easier. Commonly, second-strand synthesis follows this reverse transcription process, where second strands are created from the first strand cDNA (usually via poly(A) tailing or a template-switching mechanism)[5]. Lastly, the cDNA is then amplified before sequencing[1].

By the time the data is in our hands, we have expression levels from a plethora of cells. In the data, we can see how much an individual gene is being expressed within that cell. These expression levels are derived from observing mRNA. The amount of mRNA of a gene is an indirect indicator of how much a gene is being expressed, as mRNA is created during transcription.

Machine Learning:

This process of single-cell sequencing yields a large amount of data. But how do we make use of this data? I used supervised learning techniques in order to make sense of the data at hand. Supervised learning is a technique where the computer learns from labeled data, as opposed to unsupervised learning where the algorithm would find patterns within the data on its own. In other words, in supervised learning, when an input is provided, the machine learning algorithm predicts an output value. This prediction is based on previous labeled data entries, meaning that it has learned from this previous data. In this project, I trained various supervised learning models to predict phenotypes of a cell-based on how much a gene is expressed (which is obtained via scRNAseq). The model types I chose to use were: logistic regression, K-Nearest Neighbors (KNN), Gaussian Naive Bayes, random forest, and support vector machines.

Generally speaking, logistic regression fits the data based on maximum likelihood. Through maximum likelihood, the weights of the linear combination (which represent the correlations between inputs and outputs) change to best fit the patterns of the data.

In KNN, classification depends on the nearest data-points around the new input. The number of nearest data-points (nearest neighbors) is provided by the user, and represents the “K” in “KNN”. For example, let K = 5. First, we will “plot”* the new data point amongst the rest of the data, and then look at the 5 nearest neighbors. Based on those other 5 data points, if say 4 of them are in

[1] * plot in quotes because data in higher dimensions can’t be plotted

category α, and 1 is in category β, the new data point will be classified as α. This is because a majority of the neighbors are categorized as α.

The theory behind Gaussian Naive Bayes supervised learning technique is based on statistics. First, Gaussian curves are created for each feature based on the training data. Next, the prior probabilities are calculated, denoted as $P(y)$. Then we calculate the likelihood of each feature based on the Gaussian curves, denoted as $L(x)$. Often we then take the log of all of these likelihoods (to prevent underflow). This written as:

$$P(y) \cdot \ln\left(L(x_1) \cdot L(x_2) \cdot L(x_3) \ldots \cdot L(x_n)\right)^{*}$$

Where n is the number of features
Or in other notations[6]:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right) \quad \dagger$$

Where σ_y and μ_y are predicted based on maximum likelihood
Afterwards, we choose the output depending on which outcome is more likely to occur, based on these calculations.

In random forest, a bootstrapped dataset is created from the training data to create a series of decision trees. Each level in each decision tree uses a random subset of x amount of variables from the bootstrapped dataset. Thus by doing this multiple times we end up with a large set of decision trees. When we feed a new data-point into the random forest algorithm to classify it, we pass it through each decision tree and keep track of the output of each decision tree. The majority label based on all the decision trees is what determines the classification. We then calculate the out-of-bag error based on the data excluded in the bootstrapped dataset. Next, we can change x (the number of variables used per level in each tree) and find the most optimal number of variables, one which minimizes the out-of-bag error. Interestingly, because of the plethora of decision trees, random forest also can use these trees to create proximity matrices to help fill in missing data.

Similar to logistic regression, support vector machines (SVM) create a threshold used to classify the observations. If the data has $\mathbb{R}^n$ features, the support vector classifier will be an $\mathbb{R}^{n-1}$

---

[2]* In LaTeX: P(y)\cdot\ln\left(L(x_1)\cdot L(x_2)\cdot L(x_3)\hdots\ \cdot\ L(x_n)\right)
✝ In LaTeX: P(x_i\mid y)=\frac{1}{\sqrt{2\pi\sigma_y^2}}\exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right)

hyperplane. However, when data overlaps (see figure 1 for visuals), it is difficult to create a logical threshold. To overcome this, support vector machines increase the dimensions of the data in order to create thresholds that more accurately grapple with overlap. To transform the data into higher dimensions different kernel functions are used. Some kernel functions work better than others depending on the dataset provided.

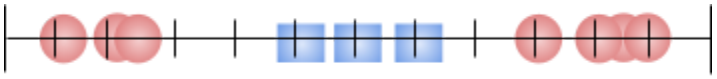

Figure 1

After dimensionality increase may look like:

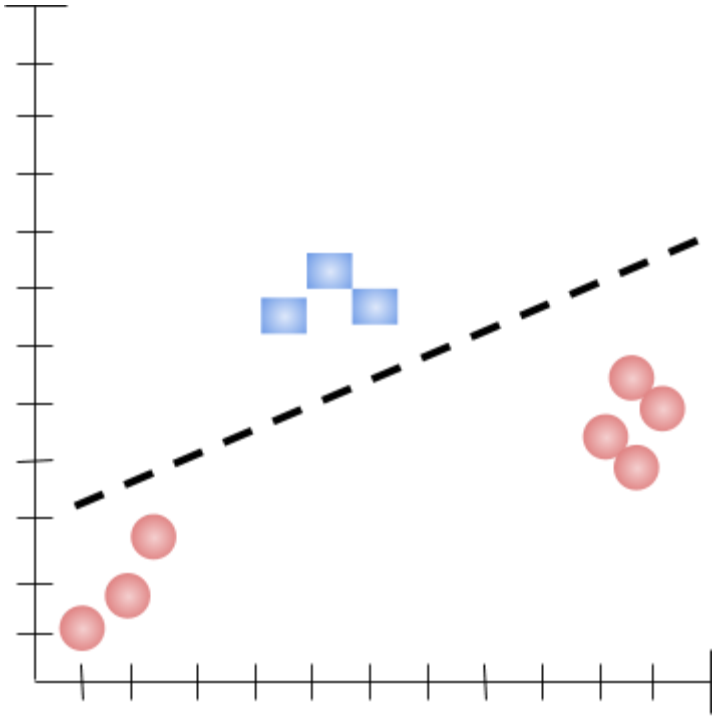

Figure 2

Given that there are a lot of models, one may ask: how can we determine which models are the best for our dataset? There are so many different hyperparameters for each model, and manually testing each version of each model can be painful. Thus, by using GridSearchCV we can tune these hyperparameters[6]. We create a search space based on the range of values per hyperparameter and use grid search to find the best model. Grid search then tests all the different variations of the models and uses cross-validation to ensure that the highest performing set of hyperparameters are truly the best parameters for the model for the specific dataset.

Methods

Data Collection

The first step in this project was to find and narrow down the datasets. I needed to choose a handful of datasets for building and training cell type classifiers. Initially, I had many different options to choose from. While searching for scRNAseq datasets, I found many datasets from a range of institutions such as 10x genomics and the Broad Institute of MIT and Harvard. I made sure to look for data collected from various types of cells and in large quantities in order to make the model as diverse and accurate as possible. In the end, I chose to focus on three single-cell datasets from published research within the Broad Institute. The first dataset is from *Defining T Cell States Associated with Response to Checkpoint Immunotherapy in Melanoma*[8, 9]. Here, researchers strived to understand and combat failures in checkpoint immunotherapy for melanoma. The researchers collected their data by taking tumor samples from melanoma patients, specifically those treated with checkpoint inhibitors. From these samples, they profiled the transcriptomes in order to analyze and improve checkpoint immunotherapy. The second dataset is from *T Helper Cell Cytokines Modulate Intestinal Stem Cell Renewal and Differentiation,* where researchers strived to gain a better understanding of how stem cells within the small intestine differentiate into specific cells[10, 11]. The third dataset is from *Mitogenic and Progenitor Programs in Single Pilocytic Astrocytoma Cells*, where researchers performed scRNA sequencing in order to better understand how mutations affected brain tumors[12, 13].

Preprocessing

After this, I preprocessed the data in order to make it more streamlined and easy to work with. First, I got rid of unnecessary columns, rows, and labels that would not add any insight to the data. After this, I saw that the metadata and the gene expression data within the same dataset had different dimensions. To take care of this issue I used the cells and data points that the metadata and gene expression data had in common.

Feature Reduction

As there were thousands of cells, I knew that to fit the original data to the models would take a long time. Thus I decided to use PCA to reduce the features, while still maintaining the original patterns within the data. Essentially, through the use of eigendecomposition, we can find the most relevant aspects of the data (principal components), and input the lower dimension data (still contains most of the initial information) into the models to make training and fitting the models much easier.

Figures 3, 6, and 8 visually show what the data looks like after performing PCA and projecting the data points with color-coding onto a 2d plane. This informs me of how challenging the data will be to classify.

As mentioned in the machine learning section, I used cross-validation grid search to find the best hyperparameters per model. Although I could have compared all models via the use of a

pipeline, analyzing each model individually helped me gain more knowledge on why some models didn't perform as well as others and gave me a better sense of the data.

For each model, I also built a confusion matrix to see how well the model performed on each cell type. Combining this with the 2d PCA plots provides more insight into how well the model performed. If one group of cells, that seem to have little correlation with anything, drags down the accuracy score of the model, and if the rest of the cells have been accurately predicted, we can conclude our model is mostly performing well.

Results

For specific grid search parameters and optimized hyperparameters, please refer to the project repository at https://github.com/rnayebi21/PolygenceMLResearch-scRNAseq.

T-Cells:
As seen in figure 3, the clusters were well defined and weren't difficult to work with. The outliers were easily taken care of. Logistic regression ended up working the best (see figure 4 for confusion matrix and table 1 for accuracies). The hyperparameters that worked the best were very similar to the default parameters of the ScikitLearn model. The major difference was that the solver that was used was 'saga'. The rest of the default hyperparameters stayed mostly the same.

Despite KNN not performing as accurately as logistic regression, I ended up with a nice accuracy vs K-value plot, suggesting that the hyperparameter for number of neighbors was successfully optimized (see figure 5).

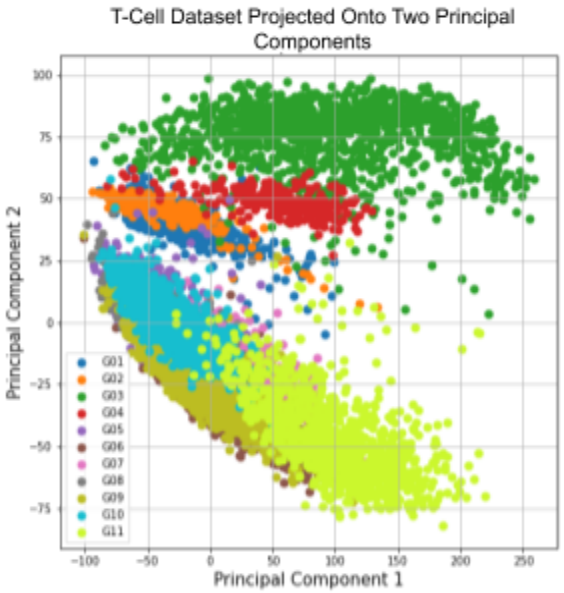

Figure 3

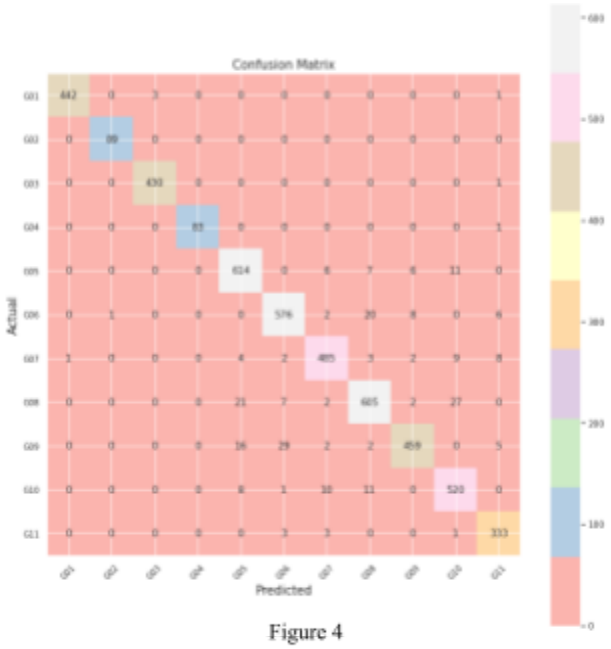

Figure 4

Cell Type Key For Figures 3-4 [8,9]:

G01	**G02**	**G03**	**G04**	**G05**	**G06**
B-cells	Plasma-cells	Monocytes/ Macrophages	Dendritic cells	Lymphocytes	Exhausted $CD8^+$ T-cells
G07	**G08**	**G09**	**G10**	**G11**	
Regulatory T-cells	Cytotoxic Lymphocytes	Exhausted/HS $CD8^+$ T-cells	Memory T-cells	Lymphocytes exhausted/cell-cycle	

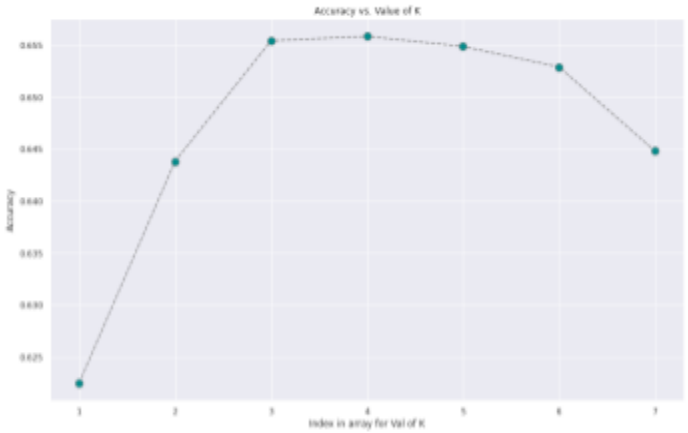

Figure 5

Brain Cells:

As seen in figure 6, the data points clustered by cell type fairly well. There are some outliers, but not too many where the data seems scattered. In this dataset random forest ended up performing the best. The two hyperparameters we tuned were *max_features* and *n_estimators*. The combination that yielded the best performance used the square root of the number of features for the max features and 954 estimators for the number of estimators (see figure 7 for confusion matrix and table 1 for accuracies).

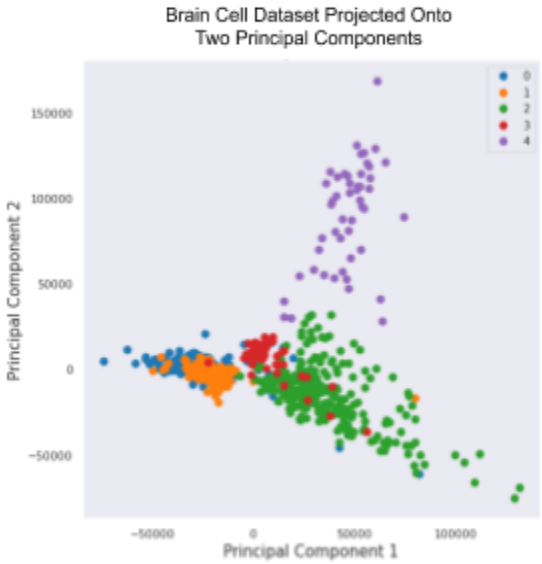

Figure 6

Figure 7

Cell Type Key For Figures 6-7 [12,13]:

0	**1**	**2**	**3**	**4**
Tumor	Tumor	Microglia	T cell	Macrophage

Stem Cells:

As seen in figure 8, the cells did not cluster by type as well as in the other two datasets. In this dataset, an SVM performed the best. I ended up tuning a couple of hyperparameters including the value for *C*, the kernel type, and the setting for gamma. (other hyperparameters that affected performance remained at their default values). The combination that ended up performing the best was using a polynomial kernel with degree 1, a *C* of ~3.6, and gamma was set to "scale" (see figure 9 for confusion matrix and table 1 for accuracies).

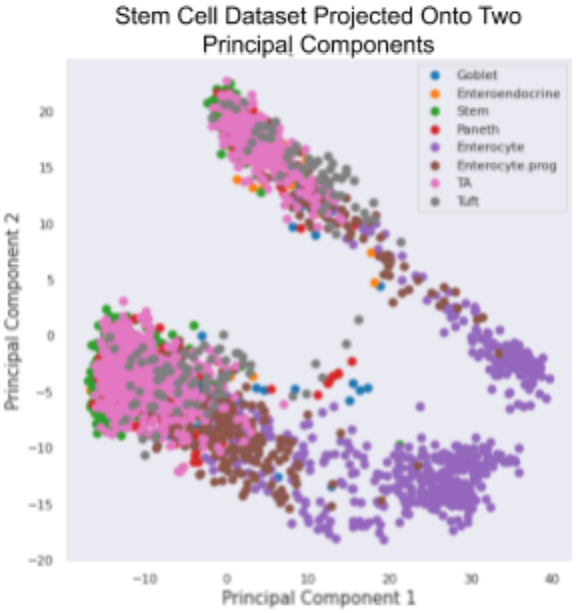

Figure 8

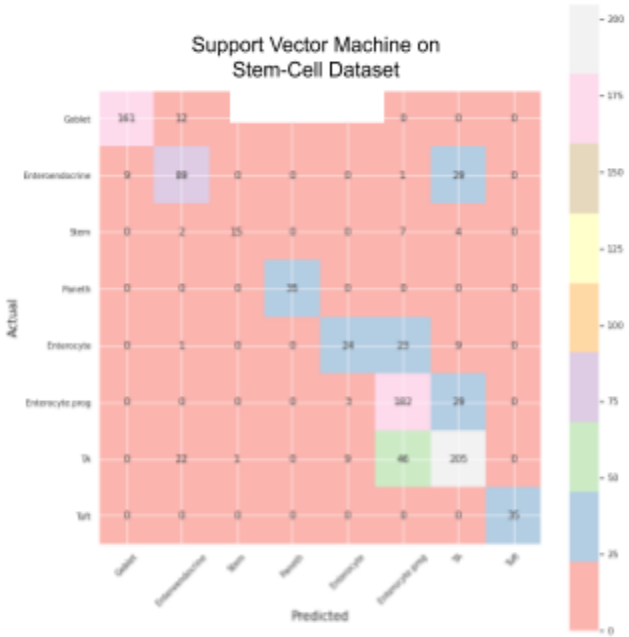

Figure 9

Table 1: Percent Accuracy Per Model Per Dataset

	Logistic Regression	K-Nearest Neighbors	Gaussian Naive Bayes	Random Forest	Support Vector Machine
T-cell dataset	**94.8%**	46.9%	52.6%	N/A	94.2%
Stem-cell dataset	76.2%	62.3%	35.9%	72.3%	**78.3%**
Brain-cell dataset	83.2 %	79.6%	78.9%	**86.4%**	N/A

Discussion:

Advantages and Limitations of Different Models

Unfortunately, because of the high computational costs, I was unable to tune the hyperparameters for random forest in our T-cell dataset, and was unable to tune SVM in the brain cell dataset over the ranges we would have liked to grid search over. I speculate that the reason it took so long to train was due to the sheer amount of cells in each dataset, and the large number of training iterations grid search requires. Despite doing a dimensionality reduction via PCA, the models still would not finish training. Although I could reduce the number of features to only a couple of principal components, I saw that by doing this a significant portion of the information was lost. In the future, I would aim to have access to more computing power and perhaps re-write some of the models to take advantage of GPUs and TPUs as well.

In the T-cell dataset, I believe that logistic regression ended up performing better than KNN and Gaussian Naive Bayes because the dataset had thousands of dimensions and because the data was log normalized when I initially downloaded it. I speculate that using KNN in large dimensions is a difficult task. I tested the model's performance using only a couple principal components, and it still did not perform well. I believe this is the case because by only using a few principal components we do not retain a significant portion of the information in the dataset. The reason Gaussian Naive Bayes did not perform well was likely due to the fact that its calculations often depend on the fact that the data is in a nice Gaussian distribution. Despite being log normalized, the scRNAseq data was likely not a perfect Gaussian distribution. However, we emphasize that given that our datasets had up to 11 classes to predict, even seemingly low accuracies (around 50%) are much better than random guesses. The dataset and model combination with the best accuracy was the T-cell dataset with logistic regression. I believe that this is because the dataset had a large amount of data, and thus the outliers have less of an impact on the overall model. This would mean that the model did not overfit, and did an overall good job of classifying the cells. We can also see from figure 3 that the groups in this dataset were relatively well defined. This would help the classifier as well. We saw that SVM performed almost as well as the logistic regression on the T-cell dataset. However for practical

purposes, even if SVM performed slightly better than logistic regression, I would still choose logistic regression. This is because grid searching over SVM took a lot longer to complete compared to grid searching over logistic regression. The same goes for random forest. The reason I think SVM performed well was because I grid searched over multiple different kernels, and the radial basis function kernel ended up working the best. This makes sense because the data, even if transformed into higher dimensions, would be difficult to separate via a linear or polynomial kernel function.

In the stem-cell dataset, the SVM performed the best. Although the brain-cell dataset has fewer cells compared to the stem-cell dataset, the brain-cell dataset has more features than the stem-cell dataset. I believe that this is why I was unable to grid search over a range of hyperparameters for GridSearchCV. Although the stem-cell dataset has fewer features, it still has a significant amount of features overall. I was surprised that the SVM performed the best. I have typically seen in examples that SVMs have difficulty separating a large amount of features. However, for similar reasoning to the T-cell dataset, logistic regression also performed quite well, while GNB and KNN did not. Typically we see that random forest performs better than SVM in data with large amounts of features. However, this isn't the case with the stem-cell dataset. Note that many of the examples we saw did not use PCA. Perhaps PCA disproportionately affects random forest.

Lastly, in the Brain-cell dataset, we can see from table 1 that the random forest classifier works the best. I personally think that when confronted with large amounts of data collected in labs, with large amounts of features, that one should stray away from using GNB and KNN. They constantly performed subpar, while the other models usually performed well, especially logistic regression (I cannot say much on random forest and SVM as those were not implemented for all of the datasets due to computational limits).

Personal Reflections

Personally what went well for me was my own personal growth as a student interested in computer science. For me, machine learning was a relatively new area of computer science that I had only scratched the surface in. I had been comfortable with AI concepts, however searches such as A* and alpha-beta are vastly differ from supervised learning such as random forest and Gaussian Naive Bayes. I found myself becoming much more articulate in numpy, pandas, and scikit-learn as well. Initially, it was a little challenging getting used to the different attributes in each package. After using them in this project, I feel more comfortable and fluent with these packages. I also loved the fact that I could combine my interest in genetics with computer science in one project. As far as learning goes on the biological side, I learned a lot more about scRNA-seq and the different techniques in how scRNA-seq occurs, concepts which I never learned in high school biology.

With regards to the coding, what went well was creating the PCA plots, confusion matrices, and getting my first logistic regression working. These went relatively seamlessly. However, when it

came to cleaning and preprocessing the data (at least for the first dataset), I had a lot of trouble. This was mainly because I knew what I needed to do, but it was initially difficult trying to translate my ideas into the numpy and pandas code.

Trying to reduce dimensions of the data gave me some difficulties at first. Before using PCA I tried using standard deviations to remove the least important aspects of the data, however this did not end up going well as I deleted much more than I intended to. I saw that using PCA is a much better method of dimensionality reduction.

Another challenging aspect of the project was getting my SVM to train on certain datasets. 24 hours was not a sufficient amount of time to train the SVM. (Note this also happened when I ran GridSearchCV with random forest on the T-cells). Due to this problem with time, it was difficult to debug.

Future Directions

The three next goals I have for this project are incorporating neural networks and seeing how they compare with the models I have currently trained. Furthermore, I would love to create a more pleasant UI to be able to use this code in the real world for real applications. Currently, these models are just in a notebook and everyday users would not find this fun to interface with. And lastly, I would like to try writing the code to be compatible with GPUs and TPUs, to be able to train the models faster.

In addition to establishing a framework for future research in cell-type classification, my research is highly relevant for educational purposes. I hope that it can provide students in both biology and computer science a real-world application of the central dogma of molecular biology and the fundamentals of machine learning.

Work Cited:

1. Hwang, Byungjin, et al. “Single-Cell RNA Sequencing Technologies and Bioinformatics Pipelines.” *Nature News*, Nature Publishing Group, 7 Aug. 2018, www.nature.com/articles/s12276-018-0071-8.
2. Julius, M H, et al. “Demonstration That Antigen-Binding Cells Are Precursors of Antibody-Producing Cells after Purification with a Fluorescence-Activated Cell Sorter.” *Proceedings of the National Academy of Sciences of the United States of America*, U.S. National Library of Medicine, July 1972, www.ncbi.nlm.nih.gov/pmc/articles/PMC426835/.
3. Whitesides, George M. “The Origins and the Future of Microfluidics.” *Nature News*, Nature Publishing Group, 26 July 2006, www.nature.com/articles/nature05058.
4. SR;, Thorsen T;Roberts RW;Arnold FH;Quake. “Dynamic Pattern Formation in a Vesicle-Generating Microfluidic Device.” *Physical Review Letters*, U.S. National Library of Medicine, 30 Apr. 2001, pubmed.ncbi.nlm.nih.gov/11328121/.
5. Islam, Saiful, et al. “Quantitative Single-Cell RNA-Seq with Unique Molecular Identifiers.” *Nature Methods*, U.S. National Library of Medicine, 22 Dec. 2013, pubmed.ncbi.nlm.nih.gov/24363023/.
6. Pedregosa, Fabian, et al. “Scikit-Learn: Machine Learning in Python.” *Journal of Machine Learning Research*, 1 Jan. 1970, jmlr.csail.mit.edu/papers/v12/pedregosa11a.html.
7. Sambrook, Joseph, and David W. Russell. “Amplification of CDNA Generated by Reverse Transcription of MRNA.” *Cold Spring Harbor Protocols*, 1 Jan. 1970, cshprotocols.cshlp.org/content/2006/1/pdb.prot3837.citation.
8. Sade-Feldman, Moshe, et al. “Defining T Cell States Associated with Response to Checkpoint Immunotherapy in Melanoma.” *Cell*, U.S. National Library of Medicine, 10 Jan. 2019, pubmed.ncbi.nlm.nih.gov/30388456/.
9. Sade-Feldman, Moshe, et al. “Study: Defining T Cell States Associated with Response to Checkpoint Immunotherapy in Melanoma 16291 Cells.” *Single Cell Portal*, 10 Jan. 2019, singlecell.broadinstitute.org/single_cell/study/SCP398/defining-t-cell-states-associated-with-response-to-checkpoint-immunotherapy-in-melanoma#study-visualize.
10. Biton, Moshe, et al. “T Helper Cell Cytokines Modulate Intestinal Stem Cell Renewal and Differentiation.” *Cell*, U.S. National Library of Medicine, 1 Nov. 2018, pubmed.ncbi.nlm.nih.gov/30392957/.
11. Biton, Moshe, et al. “Study: Intestinal Stem Cell 28462 Cells.” *Single Cell Portal*, 1 Nov. 2019 singlecell.broadinstitute.org/single_cell/study/SCP241/intestinal-stem-cell
12. Reitman, Zachary J., et al. “Mitogenic and Progenitor Gene Programmes in Single Pilocytic Astrocytoma Cells.” *Nature Communications*, U.S. National Library of Medicine, 19 Aug. 2019, pubmed.ncbi.nlm.nih.gov/31427603/.

13. Reitman, Zachary J et al. “Study: Pilocytic Astrocytoma Single Cell RNA-Seq 931 Cells.” *Single Cell Portal*, 19 Aug. 2019, singlecell.broadinstitute.org/single_cell/study/SCP271/pilocytic-astrocytoma-single-cell-rna-seq.
14. “RNA Interference (RNAi).” *National Center for Biotechnology Information*, U.S. National Library of Medicine, www.ncbi.nlm.nih.gov/probe/docs/techrnai/.