Sensational Single Cell Sequencing: Predicting Cell Types from scRNAseq Data

Ryan Nayebi
Mentor: Brianna Chrisman

Github Repository

Abstract:
The use of machine learning to analyze data obtained through single-cell RNA sequencing (scRNAseq) provides eye-opening insight into cellular biology. Applications of this type of data analysis range from understanding tumor heterogeneity (with potential in 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 different levels of gene expression within each cell. By training these models on our data I discover correlations between specific genes and their cellular phenotypic characteristics. Cell-type classification is highly relevant in areas within the medical industry. For example, cell-type classification 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 of each model have implications for establishing a framework for future research in cell-type classification.

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.

The Biology:
Every cell in our body generally (emphasis on generally) contains the same set of genetic information (genome) for encoding the functional behavior of that cell. Despite having the same information, our cells have very different functions. The reason for this is that each cell only looks at specific subsets of the entire genome (the subsets which are ignored/seen, depend on what type of cell it is). For example cardiac pacemaker cells (within the heart) would not pay attention to the subset of the genetic information that helps create the proteins necessary for foveolar cells (within the stomach). This is because when the RNA polymerase transcribes a section of the DNA (during transcription) it creates a messenger RNA (mRNA). This mRNA is then used to create proteins, with help of ribosomes, in a process known as translation. These proteins are what help the cell function. Therefore, pacemaker cells would not need to know how to create proteins that secrete mucus for stomach protection, and thus would have no need for looking at that subsection of the genome. These irrelevant sections of the genome are suppressed in many different ways, some include RNA interference (RNAi)[14] and shifts in chromatin structure caused by regulatory proteins (repressors).

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 possess specific properties (subpopulations within the unit) or if all the cells within the unit possess a property. Usually, the unit of cells is diverse and heterogeneous, however this is not obvious without single cell sequencing. Single cell sequencing also allows one to identify cells based on their RNA. Often it is difficult to detect differences in DNA, while RNA differs greatly from cell to cell (in comparison to DNA). 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 other 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 (and thus impractical). After choosing the healthy cells, they are then lysed. However, this process and its details vary depending on which technique is implemented. Some of the most popular single-cell isolation methods include: flow 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 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 a plethora of cells. In the data, we can see how much an individual gene is being expressed within that cell. To be able to see how much a gene is being expressed or not is by looking at the mRNA. The mRNA is an indirect indicator of how much a gene is being expressed, as mRNA is created during transcription. Note that the levels of gene expression in the data we used were logarithmically normalized.

The Machine Learning:

This process of single cell sequencing yields a large amount of data. But how do we make use of this data? We 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 (seen or unseen) is provided, the machine learning algorithm predicts an output from that given input. This prediction is based on previous labeled data entries, meaning that it has learned from this previous data. In this project we trained various supervised learning models to predict phenotypic features of a cell based on how much a gene is expressed (which is obtained via scRNAseq). The methods we chose to use are: logistic regression, K-Nearest Neighbors (KNN), Gaussian Naive Bayes, Random Forest, and Support Vector Machines.

The method in which logistic regression uses to create a model varies depending on its hyperparameters, however generally speaking logistic regression fits the data based on maximum likelihood (in comparison to least squares in linear regression). 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 amount 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 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.

* plot in quotes because data in higher dimensions can't be plotted

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. First 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
After 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 variety 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 decision 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 amount 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 uses these trees to create proximity matrices to help fill in missing data.

An SVC is a threshold used to classify the observations. It compensates by allowing for mis-classification in return for a higher accuracy (higher bias in return for low variance). Note that if the data exists within an $\mathbb{R}^n$ subspace, the support vector classifier will be an $\mathbb{R}^{n-1}$ hyperplane. However, when data overlaps such in the figure 1 , it is difficult to create a threshold that accurately accounts for these overlaps. To overcome this, support vector machines (SVMs)

---

* 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)

increase the dimensions of the data in order to create thresholds that more accurately grapple with overlap. And then an SVC is used to classify the data. To transform the data into higher dimensions different kernel functions are used. Some kernel functions work better than others depending on the dataset provided.

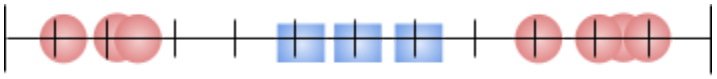

Figure 1

After dimensionality increase may look like:

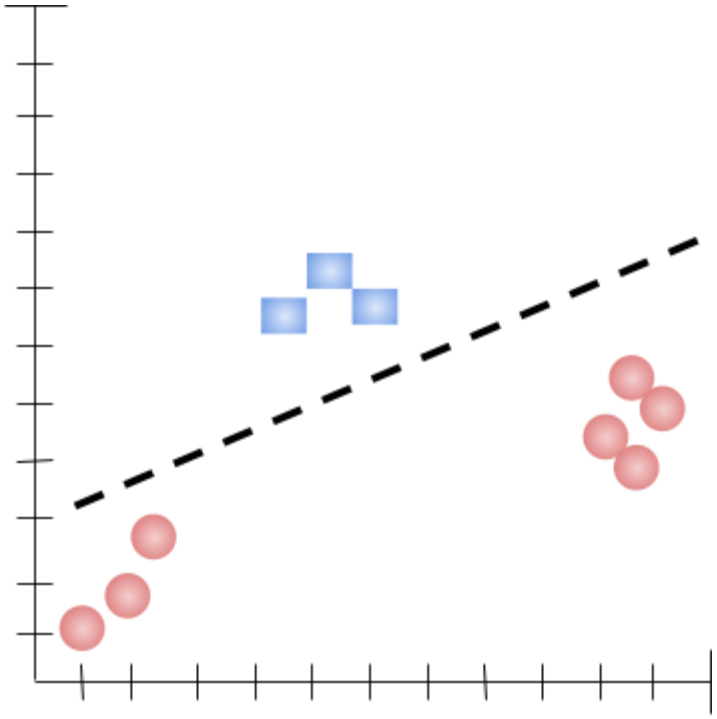

Figure 2

Although 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 hyper parameters 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 *best_params_* attribute is truly the best parameters for the model for the specific dataset.

Methods:

The first step in this process was to find and narrow down which datasets to use. Initially, we had many different options to choose from. The datasets came from a range of institutions such as 10x genomics and the Broad Institute of MIT and Harvard. We made sure to collect data from various types of cells and in large quantities in order to make the model as diverse and accurate as possible. In the end, we narrowed it down to choosing 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].

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

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

The figures in the results section that visually show what the data looks like was done by using 2 principal components and projecting them with color coding onto a 2d plane. This gives us a heads up for how challenging the data will be to predict.

As mentioned in the machine learning section, we used Cross Validation Grid Search to find the best hyper parameters per model. Although we could have compared all models via the use of a pipeline, doing each model individually helped us gain more knowledge on why some models didn't perform as well as others, and gave us a better sense of the data.

For each model a confusion matrix was also calculated to see how well the model performed on each cluster. Combining this with the 2d PCA plots gives us 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 know that our model is mostly performing well.

Results:

<u>T-Cell Dataset Results:</u>

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 hyper parameters that worked the best were very similar to the default parameters. The specifics can be seen on the github repo. 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 working super accurately, we ended up getting a nice elbow plot see figure 5.
(see specific notebooks in the github repo for more information / the range we searched over)

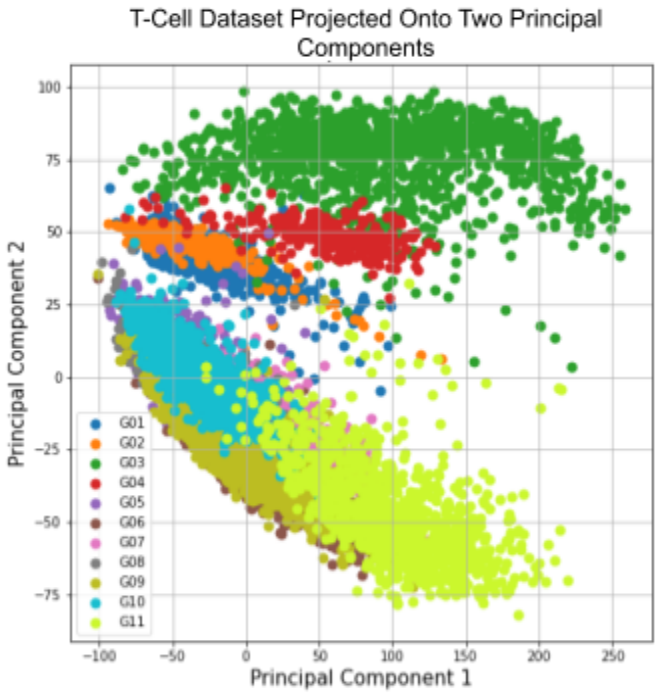

Figure 3

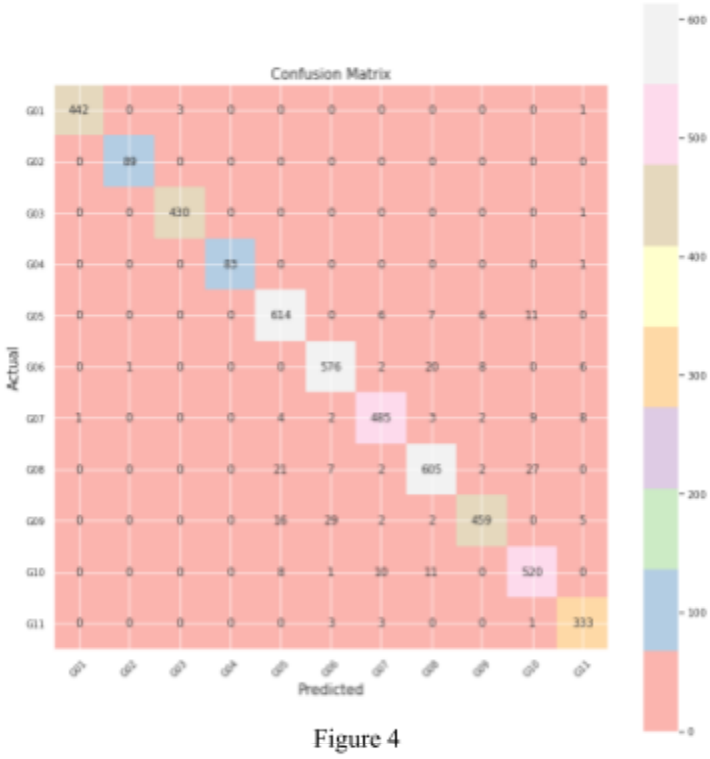

Figure 4

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	

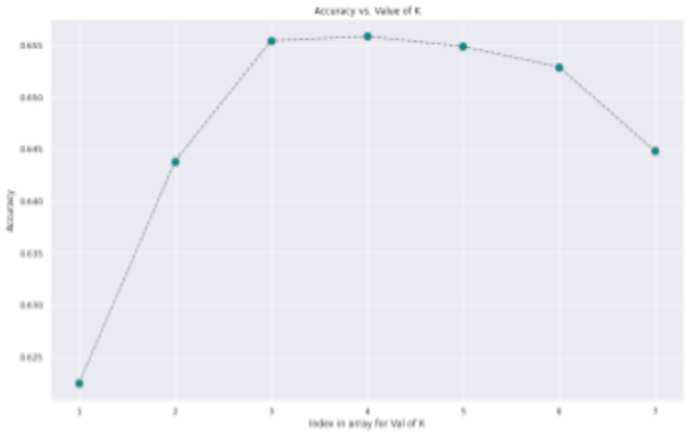

Figure 5

Figure 6

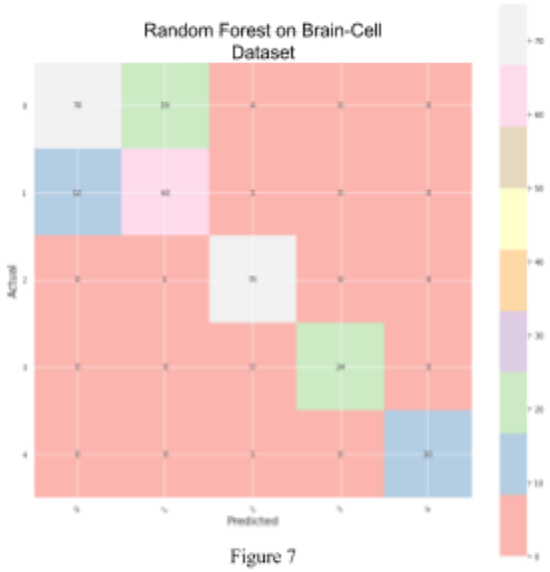

Figure 7

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

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

Brain-Cell Dataset Results:

As seen in figure 6, the clusters didn't seem completely random, they were decently defined. 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 yielded the best performance was using 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).

Stem-Cell Dataset Results:

As seen in figure 8, the clusters were a little messier than the other two. In this dataset a support vector machine performed the best. We ended up tuning a couple hyperparameters including: the value for *C*, the kernel model, 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).

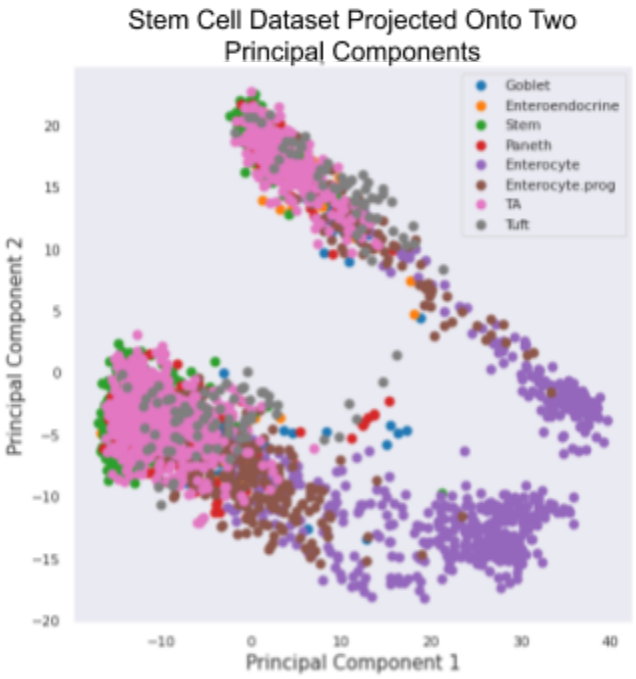

Figure 8

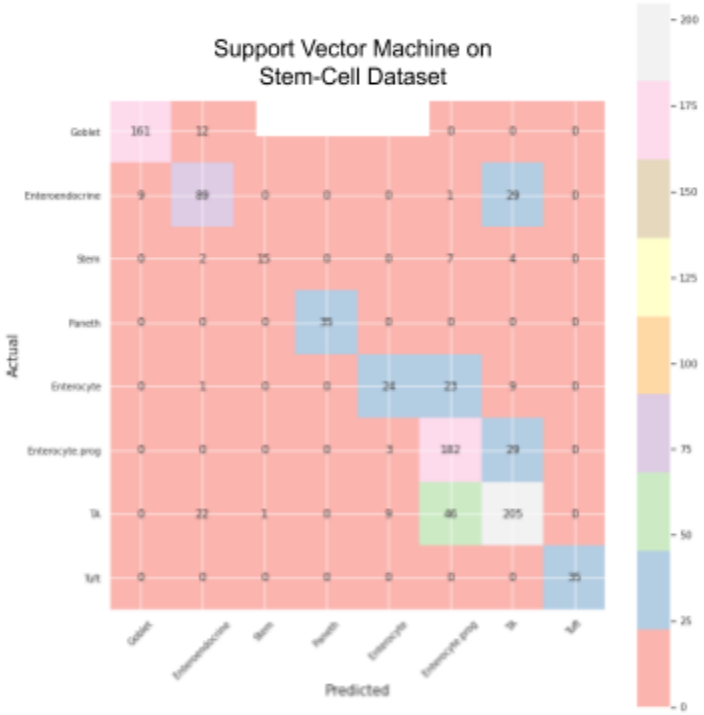

Figure 9

Table 1: Percent Accuracy Per Model Per Dataset

	Logistic Regression	K-Nearest Neighbors	Guassian 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 Section:

Unfortunately, we were unable to tune the hyper parameters for Random Forest in our T-cell dataset, and were unable to tune SVM in our brain-cell dataset over the ranges we would have liked to grid search over. A few days were not a sufficient amount of time to finish training the model. Even after these days passed, the notebook stopped fitting the data and closed the program as we exceeded the given runtime. We speculate that the reason it took so long to train was due to the sheer amount of cells in each dataset. Despite doing a dimensionality reduction via PCA the models still would not finish training. Although we could reduce the number of features to only a couple of principal components, we saw that by doing this a significant portion of the information is lost. In the future, we 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 we believe that logistic regression ended up performing better than KNN and Guassian Naive Bayes because the dataset had thousands of dimensions and because the data was log normalized when we initially downloaded it. We speculate that using KNN in large dimensions is a difficult task. We tested the model's performance using only a couple principal components, and it still did not perform well. We 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 Guassian 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 really nice Gaussian distribution. Our data, despite it being log normalized was likely not a perfect Gaussian distribution. This is just the nature of the data that we collected. However considering that there were 11 groups, 53% is much better than a random guess. The log-normalized data ended up working the best with the logistic regression, and gave us the best accuracy out of every model trained on the dataset. We 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. However for practicality reasons, even if SVM performed slightly better than logistic regression, we 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 we think SVM performed well, was because we grid searched over multiple different kernels, and the radial basis function kernel ended up working the best. This makes sense because our 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. This made us even more excited to wait for the other SVM models to train. Although the brain-cell dataset has less cells compared to the stem-cell dataset, the brain-cell dataset has more features than the stem-cell dataset. We believe that this is why we were unable to grid search over a range of hyperparameters for gridsearch. Although the stem-cell dataset has less features, it still has a significant amount of features overall. We were surprised that the SVM performed the best. We 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 (can't say much on Random Forest and SVM as not all of them even finished training).

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.

What went well with regards to the specific aspects of the coding, was creating the PCA plots, confusion matrices, and getting my first logistic regression working. These went relatively seamless. However, when it came to getting the proper dimensions (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 to the project was getting my Support Vector Machine to train on certain datasets. 12 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.

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.

Citations:

1. Hwang, B., Lee, J.H. & Bang, D. Single-cell RNA sequencing technologies and bioinformatics pipelines. Exp Mol Med 50, 96 (2018). https://doi.org/10.1038/s12276-018-0071-8
2. Reitman, Z., Paolella, B., Bergthold, G., Pelton, K., Becker, S., Jones, R., . . . Beroukhim, R. (2019, August 19). Mitogenic and progenitor gene programmes in single pilocytic astrocytoma cells. Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6700116/
3. Whitesides, G. The origins and the future of microfluidics. Nature 442, 368–373 (2006). https://doi.org/10.1038/nature05058
4. Thorsen T, Roberts RW, Arnold FH, Quake SR. Dynamic pattern formation in a vesicle-generating microfluidic device. Phys Rev Lett. 2001 Apr 30;86(18):4163-6. doi: 10.1103/PhysRevLett.86.4163. PMID: 11328121.
5. Islam, S., Zeisel, A., Joost, S. et al. Quantitative single-cell RNA-seq with unique molecular identifiers. Nat Methods 11, 163–166 (2014). https://doi.org/10.1038/nmeth.2772
6. Pedregosa F., Varoquaux G., Gramfort A., Michel V., Thirion B., Grisel O., Blondel M.,

Prettenhofer P., Weiss R., Dubourg V., Vanderplas J., Passos A., Cournapeau D., Brucher M., Perrot M., & Duchesnay E. Scikit-Learn: Machine Learning in Python, 2011, jmlr.csail.mit.edu/papers/v12/pedregosa11a.html.

7. Joseph Sambrook and David W. Russell
   Cold Spring Harb Protoc; 2006; doi:10.1101/pdb.prot3837
8. Sade-Feldman M, Yizhak K, Bjorgaard SL, Ray JP, de Boer CG, Jenkins RW, Lieb DJ, Chen JH, Frederick DT, Barzily-Rokni M, Freeman SS, Reuben A, Hoover PJ, Villani AC, Ivanova E, Portell A, Lizotte PH, Aref AR, Eliane JP, Hammond MR, Vitzthum H, Blackmon SM, Li B, Gopalakrishnan V, Reddy SM, Cooper ZA, Paweletz CP, Barbie DA, Stemmer-Rachamimov A, Flaherty KT, Wargo JA, Boland GM, Sullivan RJ, Getz G, Hacohen N. Defining T Cell States Associated with Response to Checkpoint Immunotherapy in Melanoma. Cell. 2018 Nov 1;175(4):998-1013.e20. doi: 10.1016/j.cell.2018.10.038. Erratum in: Cell. 2019 Jan 10;176(1-2):404. PMID: 30388456; PMCID: PMC6641984.
9. "Study: Defining T Cell States Associated with Response to Checkpoint Immunotherapy in Melanoma 16291 Cells." Single Cell Portal, singlecell.broadinstitute.org/single_cell/study/SCP398/defining-t-cell-states-associated-with-response-to-checkpoint-immunotherapy-in-melanoma.
10. Biton M, Haber AL, Rogel N, Burgin G et al. T Helper Cell Cytokines Modulate Intestinal Stem Cell Renewal and Differentiation. Cell 2018 Nov 15;175(5):1307-1320.e22. PMID: 30392957
11. Biton, Moshe, et al. "Study: Intestinal Stem Cell 28462 Cells." Single Cell Portal, 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 vol. 10,1 3731. 19 Aug. 2019, doi:10.1038/s41467-019-11493-2
13. Reitman, Zachary J et al. "Study: Pilocytic Astrocytoma Single Cell RNA-Seq 931 Cells." Single Cell Portal, 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/.