Written Report – 6.419x Module 2

Name: (Felipe Mehsen Tufaile)

- **Problem 2**

- **Part 1: Visualization**

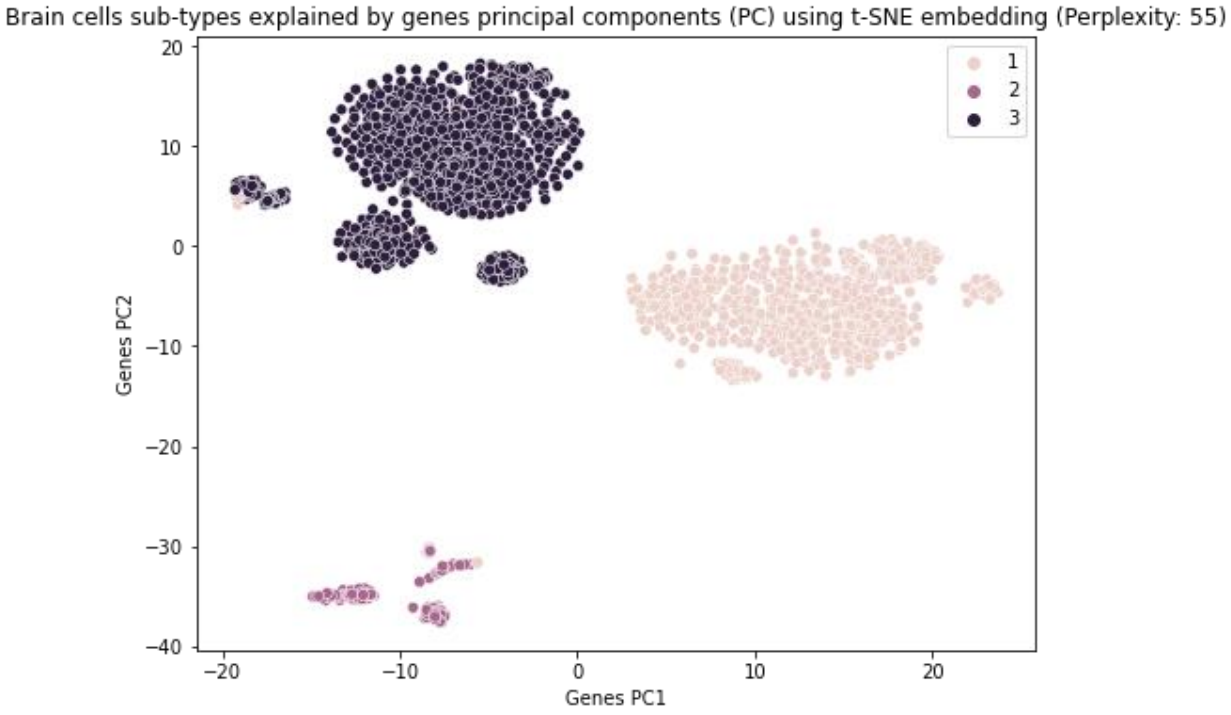
1. (3 points) Provide at least one visualization which clearly shows the existence of three main brain cell types as described by the scientist, and explain how it shows this. Your visualization should support the idea that cells from different groups can differ greatly.

Solution:

The following visualization was obtained by plotting the data with log2(X+1) transformation on the first two principal components of the t-SNE embedding for gene expression. Clusters were obtained applying K-means before implementing t-SNE. It is possible to note that t-SNE keeps similar type of brain cells together and different types of brain cells relatively far apart. We see that there is little overlap in the intervals for PC1 and PC2 that define each one of the clusters. Therefore, each cell type is defined by very different domains regarding PC1 and PC2, in other words, *different groups will differ greatly*:

cell_type	PC1_Domain	PC2_Domain	Center
1	[3.06, 23.68]	[-13.4, 1.31]	[12.99, -5.72]
2	[-14.91, -5.61]	[-37.6, -30.22]	[-10.05, -34.26]
3	[-19.29, 0.1]	[-3.54, 18.26]	[-7.73, 7.92]

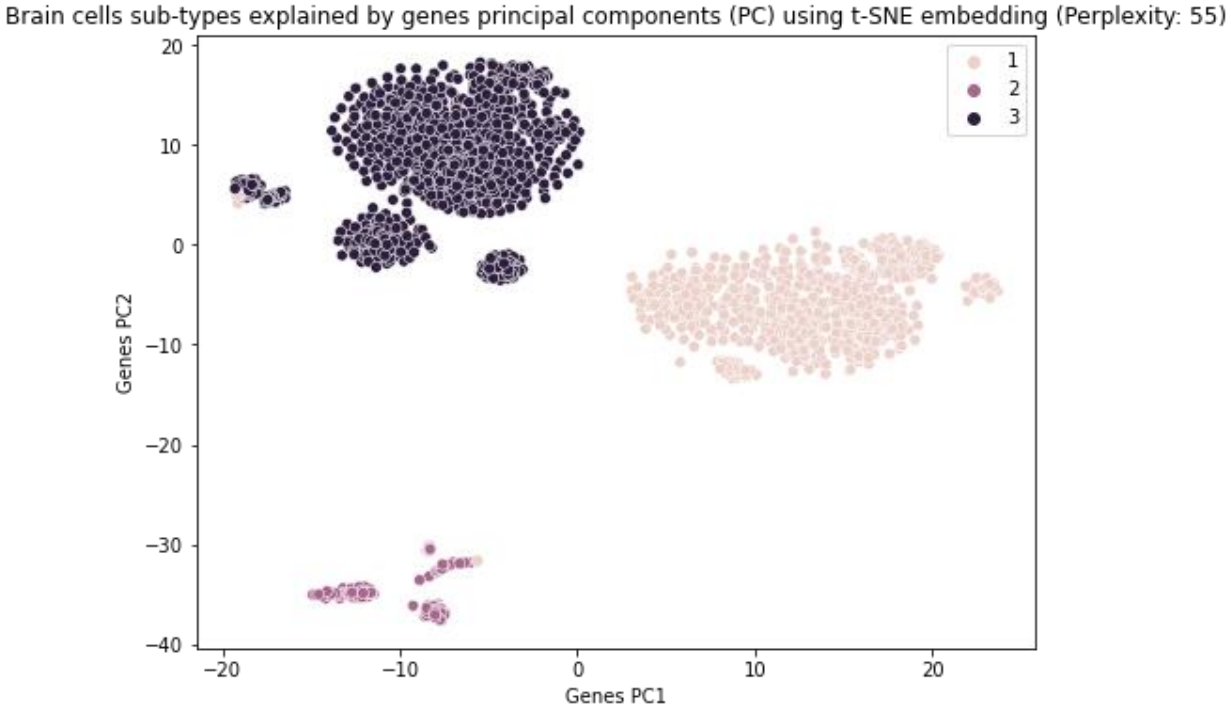
Nevertheless, it is still possible to visualize that some cells are positioned far away of their expected cluster. For example, some Brain cell type I (Beige Cluster) are positioned closest to the Brain cell type II (Pink Cluster) and to the Brain cell type III (Purple Cluster).



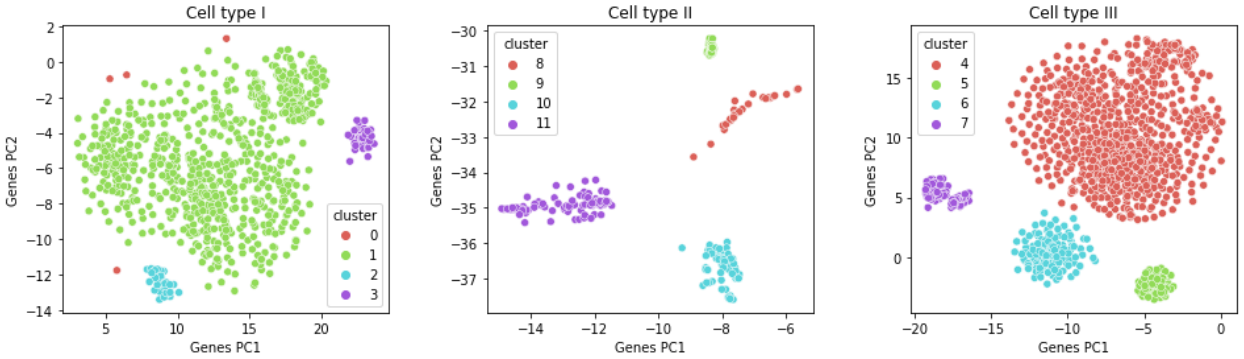
2. (4 points) Provide at least one visualization which supports the claim that within each of the three types, there are numerous possible sub-types for a cell. In your visualization, highlight which of the three main types these sub-types belong to. Again, explain how your visualization supports the claim.

Solution:

The same visualization as before can be used to show the sub-types of each cell type. If we look at Brain cell type III (Purple Cluster), for example, we can see a large aggregation of points surrounded by 3 other smaller groups of points. Each one of these 4 aggregation of points mentioned (1 large group and 3 other smaller groups) would represent sub-types of the Brain cell type III.



To better visualize these sub-types of cells, we can perform a DBSCAN clustering using the two first components of the t-SNE embedding. This will result in the following visualizations:



It becomes clear that Brain cell type I has 3 sub-types, whereas Brain cell type II and Brain cell type III has 4 sub-types each. The red cluster in the Brain cell type I (cluster 0) seems to be indicating the outliers of this cluster obtained through the DBSCAN clustering process. DBSCAN was performed using the following parameters: eps=1.51 and min_samples=10 (all other parameters were set to default).

▪ Part 2: Unsupervised Feature Selection

1. (4 points) Using your clustering method(s) of choice, find a suitable clustering for the cells. Briefly explain how you chose the number of clusters by appropriate visualizations and/or numerical findings. (to cluster cells into the subcategories instead of categories).

Solution:

For this part of the problem, clusters were defined using the following process:

1. Applying log2(x+1) transformation on the original data;
2. Reducing the number of features (genes) to 2 using t-SNE using perplexity=55 (all other parameters were set to default);
3. Implementing DBSCAN on the t-SNE embedding mentioned above using eps=1.51 and min_samples=10 (all other parameters were set to default);

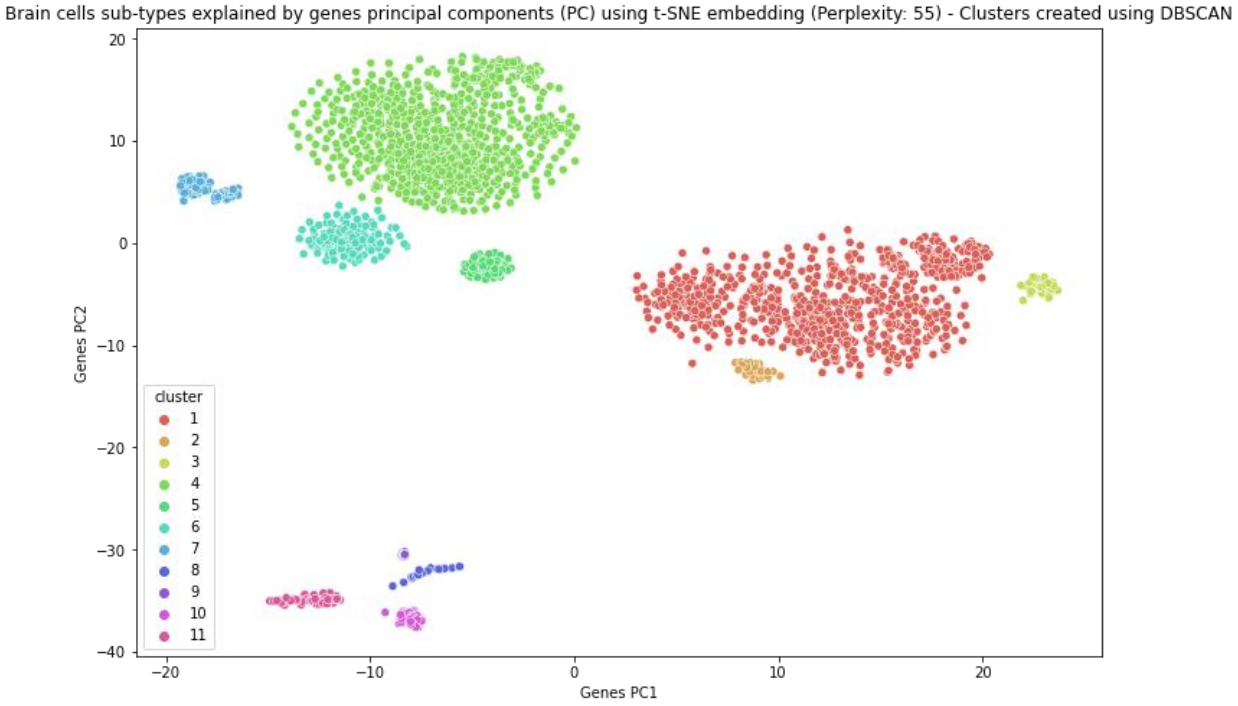
Following the above process, 12 clusters were obtained as described in the table below. We can see that cluster 0 and cluster 1 have very similar domain (PC1 and PC2) and Center. This suggests that this cluster could be grouped together reducing the total number of clusters to 11. All other clusters seem to significantly differ from each other, indicating that no further grouping is required.

cell_type	cell_sub_type	PC1_Domain	PC2_Domain	Center
1	0	[5.29, 13.4]	[-11.76, 1.31]	[7.73, -3.04]
1	1	[3.06, 20.33]	[-12.92, 0.69]	[12.65, -5.49]
1	2	[7.89, 10.1]	[-13.4, -11.67]	[8.87, -12.41]
1	3	[21.87, 23.68]	[-5.61, -3.3]	[22.82, -4.27]
2	8	[-8.89, -5.61]	[-33.57, -31.64]	[-7.2, -32.24]
2	9	[-8.47, -8.25]	[-30.69, -30.22]	[-8.34, -30.5]
2	10	[-9.26, -7.46]	[-37.6, -35.98]	[-8.01, -36.66]
2	11	[-14.91, -11.46]	[-35.43, -34.22]	[-12.96, -34.92]
3	4	[-13.83, 0.1]	[3.13, 18.26]	[-6.49, 10.53]
3	5	[-5.4, -3.09]	[-3.54, -0.91]	[-4.31, -2.29]
3	6	[-13.46, -8.2]	[-2.24, 3.69]	[-11.1, 0.4]
3	7	[-19.29, -16.46]	[4.14, 6.56]	[-18.03, 5.27]

The adjusting the number of clusters results in the updated description of each cluster:

cell_type	cell_sub_type	PC1_Domain	PC2_Domain	Center
1	1	[3.06, 20.33]	[-12.92, 1.31]	[12.62, -5.48]
1	2	[7.89, 10.1]	[-13.4, -11.67]	[8.87, -12.41]
1	3	[21.87, 23.68]	[-5.61, -3.3]	[22.82, -4.27]
2	8	[-8.89, -5.61]	[-33.57, -31.64]	[-7.2, -32.24]
2	9	[-8.47, -8.25]	[-30.69, -30.22]	[-8.34, -30.5]
2	10	[-9.26, -7.46]	[-37.6, -35.98]	[-8.01, -36.66]
2	11	[-14.91, -11.46]	[-35.43, -34.22]	[-12.96, -34.92]
3	4	[-13.83, 0.1]	[3.13, 18.26]	[-6.49, 10.53]
3	5	[-5.4, -3.09]	[-3.54, -0.91]	[-4.31, -2.29]
3	6	[-13.46, -8.2]	[-2.24, 3.69]	[-11.1, 0.4]
3	7	[-19.29, -16.46]	[4.14, 6.56]	[-18.03, 5.27]

Finally, the 11 cluster defined in the above table can be seen plotted together using the first two principal components of the t-SNE embedding in the visualization bellow. Looking at the image, it is possible to note that the 11 clusters obtained can be easily visually identified.



2. (6 points) We will now treat your cluster assignments as labels for supervised learning. Fit a logistic regression model to the original data (not principal components), with your clustering as the target labels. Since the data is high-dimensional, make sure to regularize your model using your choice of l_1, l_2, or elastic net, and separate the data into training and validation or use cross-validation to select your model. Report your choice of regularization parameter and validation performance.

Solution:

For this question, the "y" vector mentioned below contains the cluster labels for each one of the 2169 cells (rows) in the X_log matrix (X matrix with log2(X+1) transformation). Cluster labels range from 1 to 11 as indicated in the previous question (see image in the question). The X_log matrix and y vector were separated into training (70% of the cells) and test (30% of the cells) sets considering stratification on cluster labels (see code below). This way, each set would have the same percentage of each cluster label. In order to ensure that the sampling process is reproduceable, the random state parameter was set to 42.

```
# Splitting dataset into train, test and validation sets
X_train, X_test, y_train, y_test = train_test_split(
      X_log, y, test_size=0.30, stratify=y, random_state=42
)
```

The logistic regression model was trained using l1 regularization (Lasso regularization). Lasso regularization was used since it forces the coefficient of less important features to zero (or very close to zero), which is useful in terms of feature selection. For this reason, the solver chosen for optimization was the "liblinear" since it allows l1 regularization. The regularization parameter "C" was selected by chosen 100 possible choices (Cs=100) in a logarithmic scale between 1e-4 and 1e4 through cross-validation. The number of folds was set to 5. Finally, an one-versus-rest approach was implemented by setting the "multi_class" parameter to "ovr". The code implemented is shown in the below.

```
# Implementing cross-validation on logistic regression
cv_clf = LogisticRegressionCV(
    cv=5,                   # Number of cross-validation folds
    Cs=100,                 # Number of regularization values to attempt
    penalty='l1',           # Regularization method: l1 - Lasso
    solver='liblinear',     # Solver that allows l1 regularization
    tol=1e-5,               # Tolerance
    max_iter=100,           # Max number of iterations
    multi_class='ovr',      # Performing one-versus-rest
    random_state=42         # To ensure reproducibility given solver chosen
).fit(X_train, y_train)
```

After the model was trained, its performance was evaluated by predicting labels using the test set. This step is shown in the following code. The overall accuracy (across all cluster labels) obtained was 99.5%.

```
# predicting using the test set
preds = cv_clf.predict(X_test)

# overall performance
print(f"Overall Accuracy: {round(np.mean(1*(preds == y_test)), 3)}")
>> Overall Accuracy: 0.997
```

The accuracy obtained for each one of the cluster labels is shown in the table below.

Cluster Label	True Positive (Predicted)	Total	Accuracy
1	234	234	1.000000
2	11	11	1.000000
3	13	14	0.928571
4	238	239	0.995816
5	25	25	1.000000
6	41	41	1.000000
7	23	23	1.000000
8	7	7	1.000000
9	12	12	1.000000
10	18	18	1.000000
11	27	27	1.000000

3. (9 points) Select the features with the top 100 corresponding coefficient values (since this is a multi-class model, you can rank the coefficients using the maximum absolute value over classes, or the sum of absolute values). Take the evaluation training data in p2_evaluation and use a subset of the genes consisting of the features you selected. Train a logistic regression classifier on this training data, and evaluate its performance on the evaluation test data. Report your score. (Don't forget to take the log2(x+1) transform before training and testing.)

Compare the obtained score with two baselines: random features (take a random selection of 100 genes), and high-variance features (take the 100 genes with highest variance). Finally, compare the variances of the features you selected with the highest variance features by plotting a histogram of the variances of features selected by both methods.

For each one of the mentioned models in this question, features selected according to the following approach:

- Best 100 features:
  - index_best_100 = (-np.sum(np.abs(cv_clf.coef_), axis=0)).argsort()[:100]
- Random 100 features:
  - index_random_100 = np.random.choice(a=range(0, X_train_p2_eval.shape[1]), size=100)
- Highest Variance 100 features:
  - index_high_variance_100 = (-np.std(X_train_p2_eval, axis=0)).argsort()[:100]

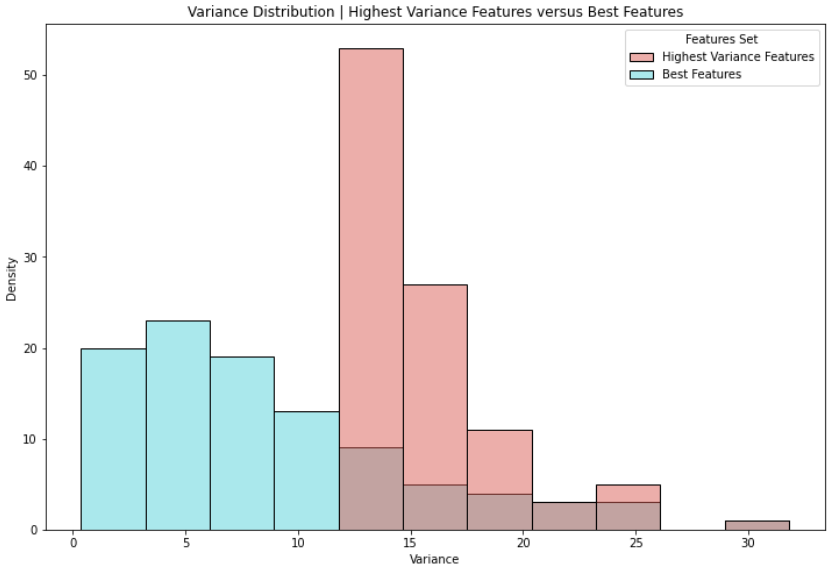
Where cv_clf is the Logistic Regression model obtained in the previous question.

The Logistic Regression model was re-trained for each feature selection case using the same parameters mentioned in the previous question. The overall classification accuracy obtained for each case was:

- Best 100 features: 88.8%
- Random 100 features: 18.9%
- Highest Variance 100 features: 89.2%

The model with the highest variance features yielded the best accuracy among all models, although the accuracy obtained for the "Best" features model is not very far behind. The accuracy obtained for the model with 100 random features was considerably lower than the accuracy obtained for the other two model. The poor performance for this last model was also noticeable during the training process, since it was necessary to increase the number of maximum iterations to guarantee convergence. Finally, the histogram of the variances of the features selected by both methods ("Best" features and high variance features) shows that features in both sets encompass a lot of the variance in the dataset. That is, although the distribution of the features with higher variance is centered around a higher average variance, both distributions have a much higher average variance compared to the distribution of the variance of all 45768 features (see table below). This suggests that features with high prediction power are often features that capture significative amount of variance.

	Variance (All Features)	variance (Random Features)	variance (Best Features)	variance (High Variance Features)
count	45768.000000	100.000000	100.000000	100.000000
mean	1.542521	1.697416	8.572865	15.572741
std	2.556516	2.643577	6.480413	3.537081
min	0.000000	0.000000	0.357199	12.212345
25%	0.000487	0.001219	3.929286	13.174898
50%	0.083145	0.107587	6.770670	14.389025
75%	2.207559	2.647783	11.767781	17.083871
max	31.813997	11.210671	31.813997	31.813997

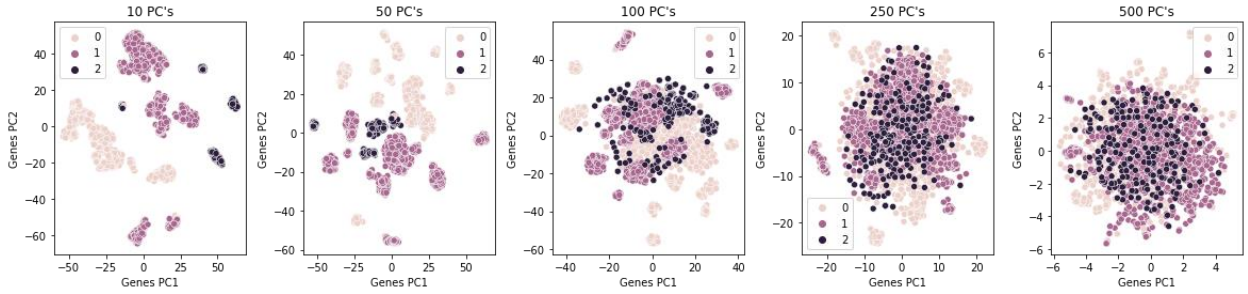


▪ Problem 3

1. (3 points) When we created the T-SNE plot in Problem 1, we ran T-SNE on the top 50 PC's of the data. But we could have easily chosen a different number of PC's to represent the data. Run T-SNE using 10, 50, 100, 250, and 500 PC's, and plot the resulting visualization for each. What do you observe as you increase the number of PC's used?

Solution:

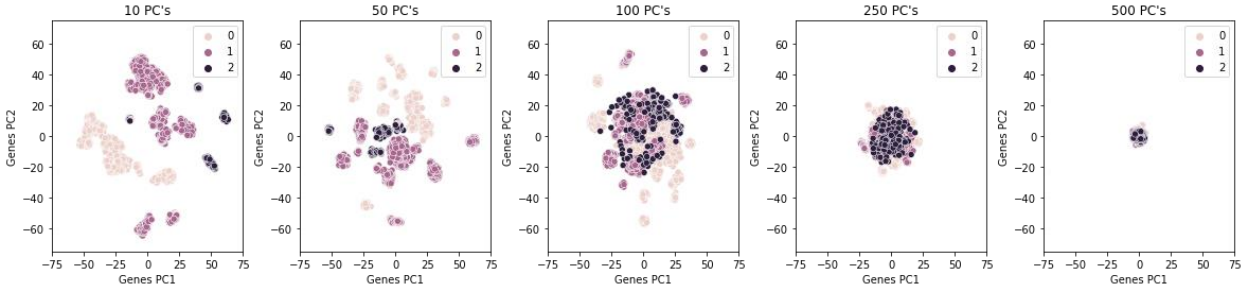
As we increase the number of principal components used in the t-SNE algorithm, the resulting plot seems to compress all data points into a cloud of points (see images below). The reason for that is probably because by adding more principal components we are adding more noise to the data which makes the t-SNE algorithm to overfit the data.



PCA is a **linear** technique that projects the data in the directions of its greatest variance while trying to preserve its **global structure**. Since the technique attempts to gather most of the variance in the data in the first few components, the last components will explain very little of the variance and, therefore, can be interpreted as "noise".

T-SNE, on the other hand, is a **nonlinear** technique that preserves the **local structure** of the data by clustering nearby points together. Hence, when noise is introduced by using an excessive number of principal components, the t-SNE algorithm may "lose track" of the local structure of the data, since all data points will have the same low-variance high-dimension structure. This will make the algorithm compress

all data together keeping a relatively similar distance between all points, which gives this impression of a cloud of points.



2. (13 points) Pick three hyper-parameters below (the 3 is the total number that a report needs to analyze. It can take a) 2 from A, 1 from B, or b) 1 from A, 2 from B.) and analyze how changing the hyper-parameters affect the conclusions that can be drawn from the data. Please choose at least one hyper-parameter from each of the two categories (visualization and clustering/feature selection). At minimum, evaluate the hyper-parameters individually, but you may also evaluate how joint changes in the hyper-parameters affect the results. You may use any of the datasets we have given you in this project. For visualization hyper-parameters, you may find it productive to augment your analysis with experiments on synthetic data, though we request that you use real data in at least one demonstration?

Solution:

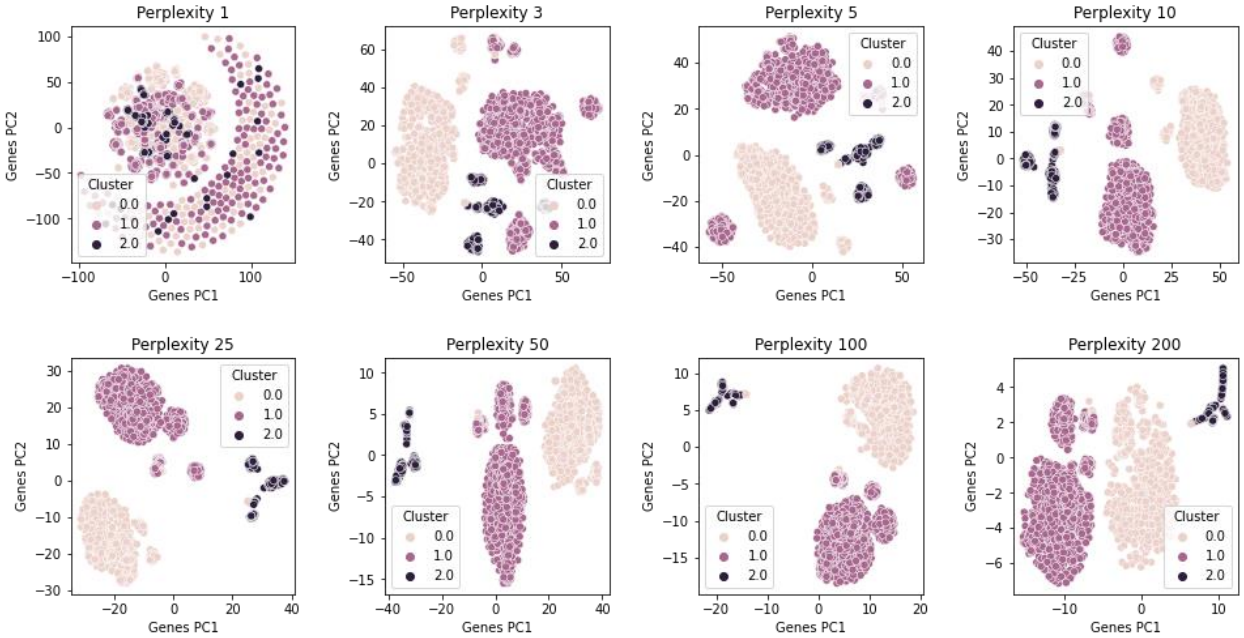
1. Category A(visualization) – t-SNE perplexity

The following sequence of images was generate plotting the log2(X+1) transformed data onto the two first components of t-SNE embeddings with varying perplexity values (learning rate was fixed to 200 and all other parameters were set to default). Looking at the images, we can observe that for small perplexity values t-SNE generates a large number of small clusters whereas for larger perplexity values t-SNE generates a small number of large clusters. This suggests that as we increase the perplexity value, smaller clusters are aggregated are aggregated together into larger clusters.

According to the documentation of the tool, the perplexity parameter is related to the number of nearest neighbors that is used to compute the probability distribution over pairs of high-dimensional data. In this sense, the parameter controls the balance between preserving **local structure** and preserving **global structure**: if we set a small perplexity value, we tend to preserve local structure whereas if we set a large perplexity value, we tend to preserve global structure.

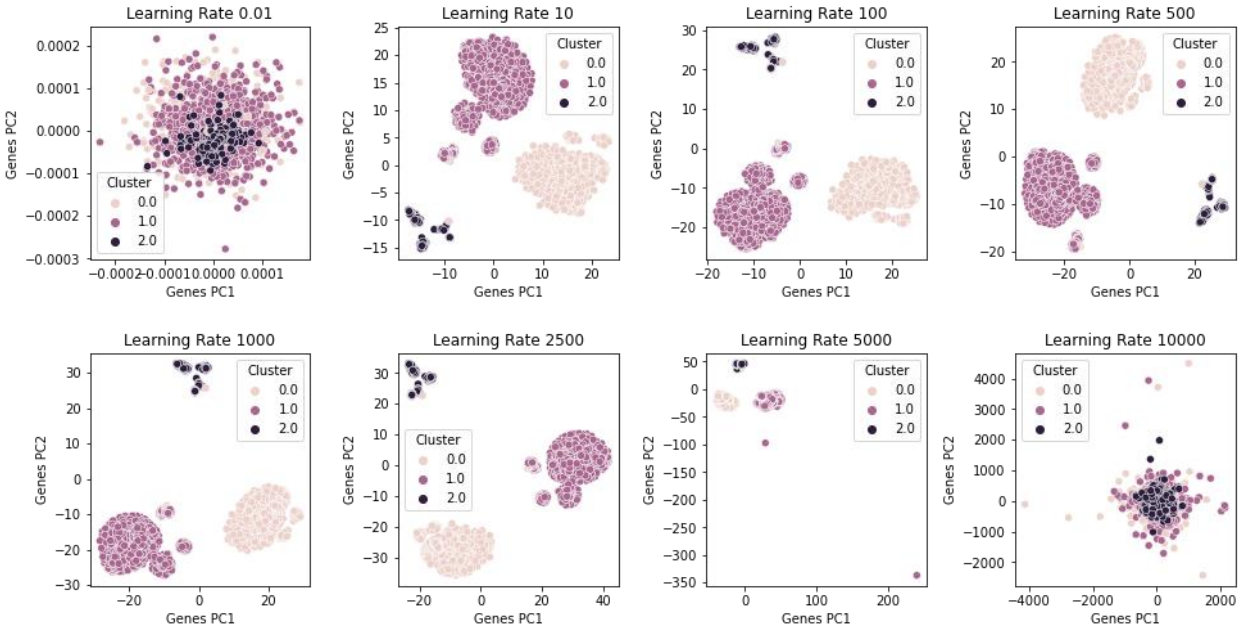
Looking back at the sequence of images we can see that for perplexity equal to 1, for example, t-SNE is grouping only points that are very close together in the original high-dimensional space, attempting to preserve the local structure. It is even noticeable that some datapoints are isolated as their one cluster, probably due to the fact that these points have no neighbors within its perplexity range.

On other hand, for perplexity equal to 200, t-SNE tends to "ignore" small differences between datapoints and focus on differences that are more significative for the set of datapoints as a whole. This results in a smaller number of more distinct cluster that are well-separated from one another, as an attempt to preserve the global structure.



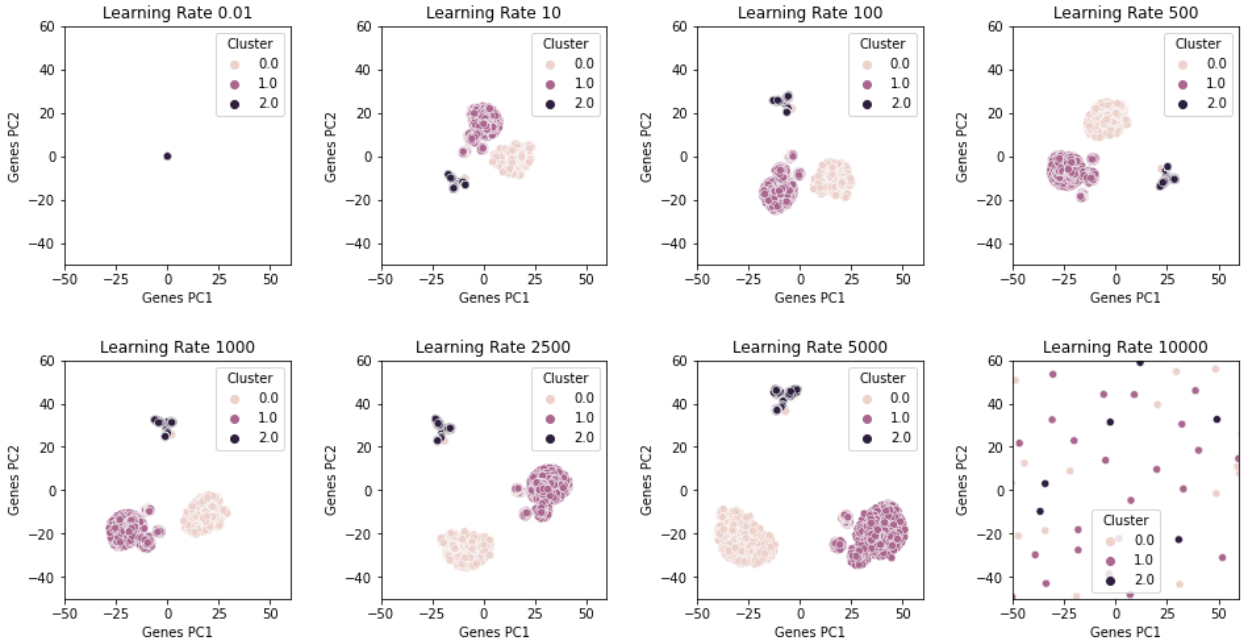
2. Category A (visualization) – t-SNE learning rate

The following sequence of images was generated plotting the log2(X+1) transformed data onto the two first components of t-SNE embeddings with varying learning rates (perplexity was fixed to 50 and all other parameters were set to default). We see that very small learning rate values, for instance 0.01, the algorithm group most datapoints close together into a single compact cluster, making it hard to tell the difference between different clusters. According to the documentation, this is described as most points looking like they are being compressed in a dense cloud with few outliers.



For very large learning rate, it seems that the algorithm is trying to isolate all datapoints into their own clusters. The apparent aggregation we see in the subplot with learning rate of 10000 might be due to the impact of outliers that are placed far apart from most datapoints. Therefore, if we zoom all subplots where most points are concentrated (i.e. PC1 and PC2 in the range of [-50, 60]) we can observe the impact of the learning rate more clearly (see next image).

Looking at the sequence of plots in the zoomed image below we note that the distance between clusters seems to increase as we increase the learning rate until each datapoint becomes its own cluster. In the documentation of the algorithm, this is described as a configuration where the data looks like a "ball" with any point approximately equidistant from its nearest neighbours.

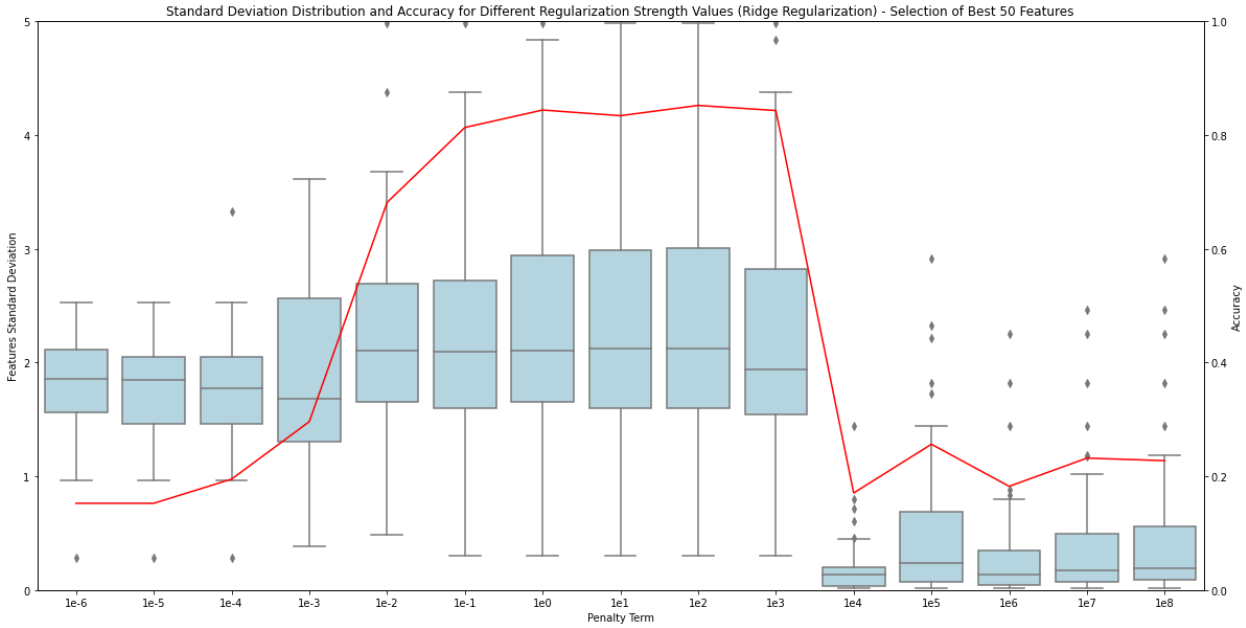


3. Category B (clustering / feature selection) – Magnitude of regularization and its relation to your feature selection

In order to analyze the impact of the magnitude of regularization in feature selection, it was trained 15 Logistic Regression models using **Ridge Regularization (l2),** with regularization strength varying from 1e-6 to 1e8. For each model, it was selected the **50 features** with the highest sum of the absolute values over all the classes. Results can be seen in the figure below.
Looking at the figure, it is possible to note the following aspects:

- For very small penalty terms (i.e. 1e-6 to 1e-4), the distribution of the standard deviation of the features selected (**box plot**) is very concentrated around the mean, indicating that features selected have very similar standard deviation;
- Increasing the penalty term from 1e-3 to 1e3 produces a significative increase in accuracy (**red line**) at the same time as features with higher standard deviation in average are selected. Additionally, is noticeable that the distribution of the standard deviation of the features become more spread out meaning that we might be selecting a better mix of features with high and low standard deviation. Therefore, by increasing the penalty term from 1e-3 to 1e3 we are capable of selecting a better mix of features;

- Finally, we also note that if we increase too much the penalty term (i.e. penalty term > 1e3) we end-up selecting mostly features with small standard deviation at the same time the accuracy of the model decreases significantly. Thus, suggesting that very large penalty terms lead to the choice of bad variables. The explanation for this fact would be that for large penalty terms all coefficients become very small, making all variables more equally likely to be chosen as best. Therefore, since most variables in the dataset studied have very low standard deviation, it is more likely that we end-up choosing bad variables with low standard deviation.



Reference

[1] T-SNE Algorithm Documentation, Scikit-Learn, 2023, https://scikit-learn.org/stable/index.html