Written Report – 6.419x Module 2

Name: (vatsalya243001)

- **Problem 2**

Part 1: Visualization

A scientist tells you that cells in the brain are either excitatory neurons, inhibitory neurons, or non-neuronal cells. Cells from each of these three groups serve different functions within the brain. Within each of these three types, there are numerous distinct sub-types that a cell can be, and sub-types of the same larger class can serve similar functions. Your goal is to produce visualizations which show how the scientist's knowledge reflects in the data.

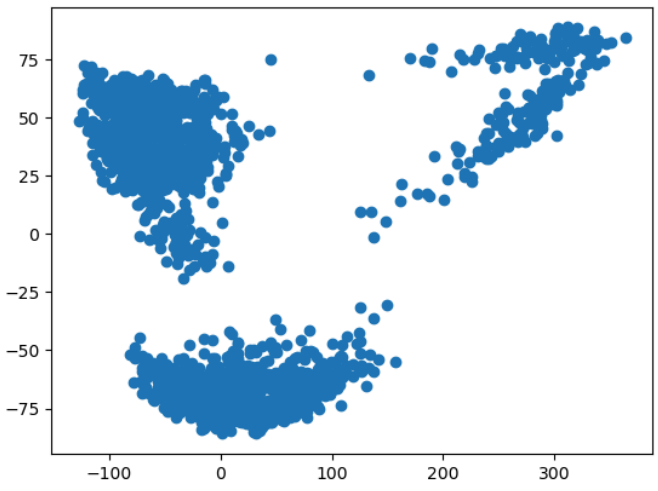
As in Problem 1, we recommend using PCA before running T-SNE or clustering algorithms, for quality and computational reasons.

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 visualization below depicts the projection of the dataset onto the first two orthonormal principal components, which capture the directions of maximum variance in the dataset. The figures clearly show three distinct groups of cells. These groups are significantly different from each other, forming three well-separated and well-defined clusters.

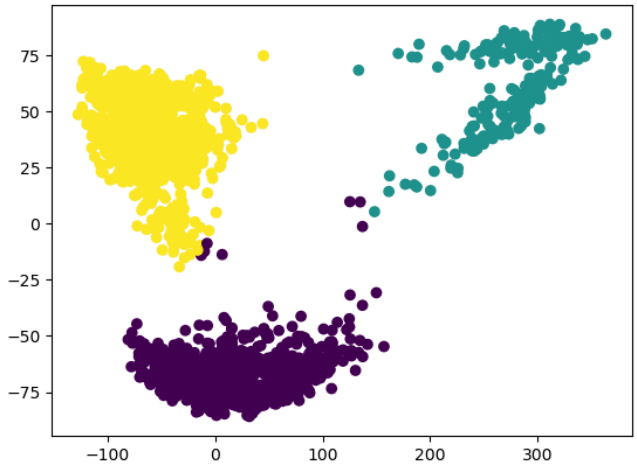
```
X_new = PCA(n_components=2).fit_transform(X)
plt.scatter(X_new[:,0], X_new[:,1])
```



We can identify the clusters using the k-means clustering algorithm on the projected space. One cluster is compact and centered around (-65, 40), another is elongated along the x-axis and

centered around (13, -65), and the third is somewhat scattered and centered around (272, 60) in terms of the (PC1, PC2) coordinates, as shown in the figure below.

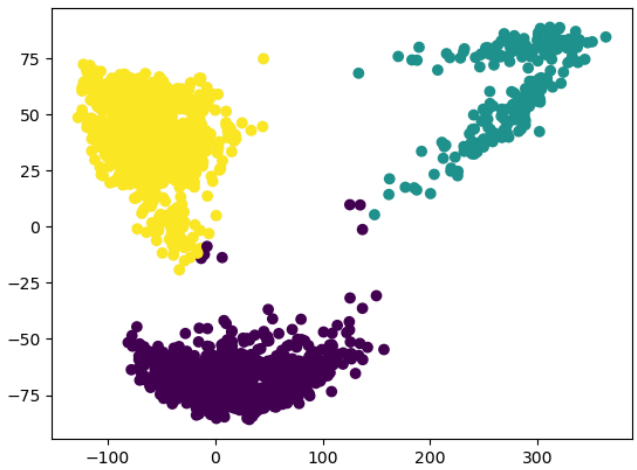
```
kmeans = KMeans(n_clusters=3, random_state=1).fit(X_new)
plt.scatter(X_new[:, 0], X_new[:, 1], c=kmeans.labels_)
```



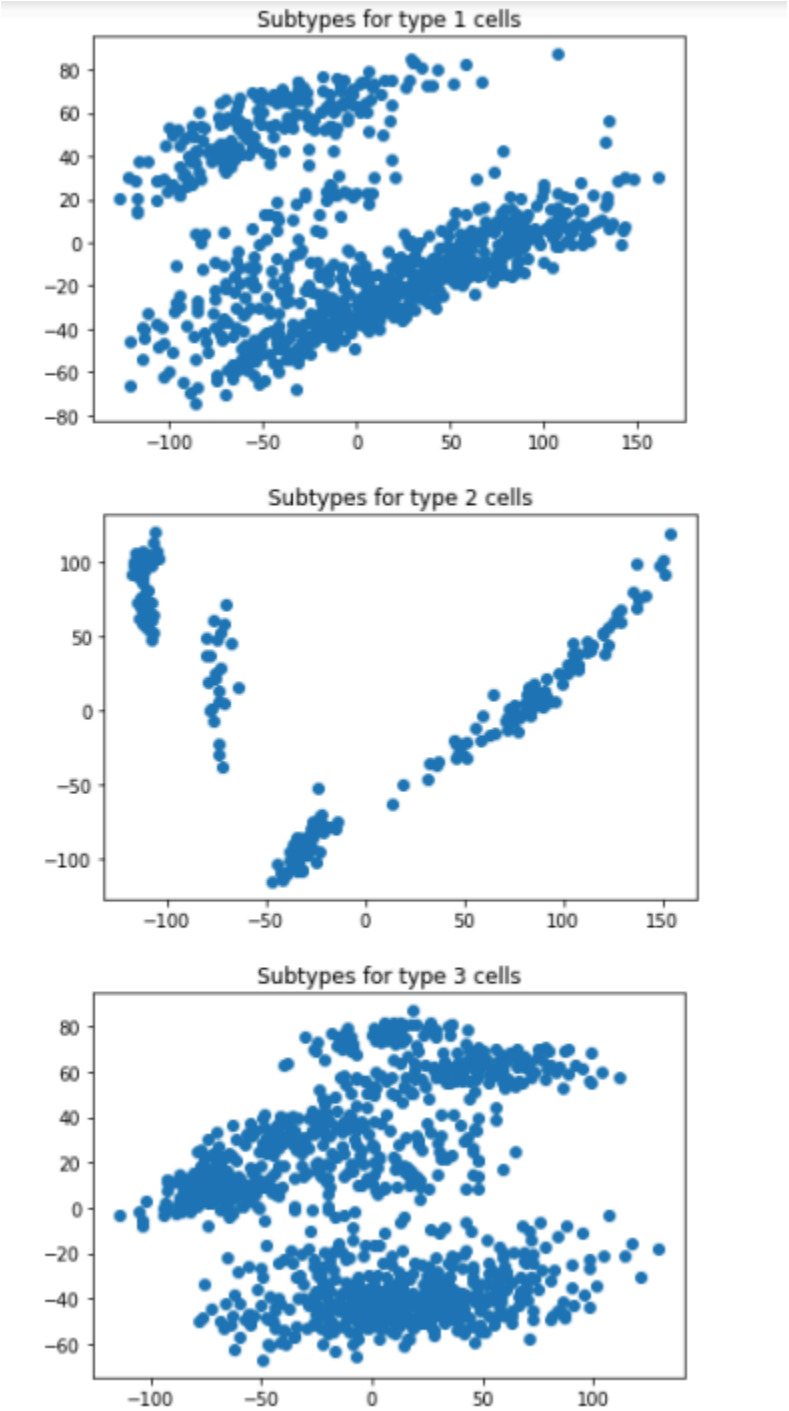
2. (4 points) Provide at least one visualization which supports the claim that within each of the three types, there are numerous possible sub-types for a cell. In your visualization, highlight which of the three main types these sub-types belong to. Again, explain how your visualization supports the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

Solution:

Let's divide the data into the three clusters identified above (as shown and numbered in the figure below), and then apply the k-means algorithm to each group individually to identify subclusters (subtypes).



```
for i in range(3):
    X_sub = X[kmeans.labels_==i]
    X_sub_new = pca.fit_transform(X_sub)
    plt.scatter(X_sub_new[:,0], X_sub_new[:,1])
    plt.show()
```



As can be seen from the above figures, the individual types can have many subtypes (some of them are very well-separated).

Part 2: Unsupervised Feature Selection

Now we attempt to find informative genes which can help us differentiate between cells, using only unlabeled data. A genomics researcher would use specialized, domain-specific tools to select these genes. We will instead take a general approach using logistic regression in conjunction with clustering. Briefly speaking, we will use the p2_unsupervised dataset to cluster the data. Treating those cluster labels as ground truth, we will fit a logistic regression model and use its coefficients to select features. Finally, to evaluate the quality of these features, we will fit another logistic regression model on the training set in p2_evaluation, and run it on the test set in the same folder.

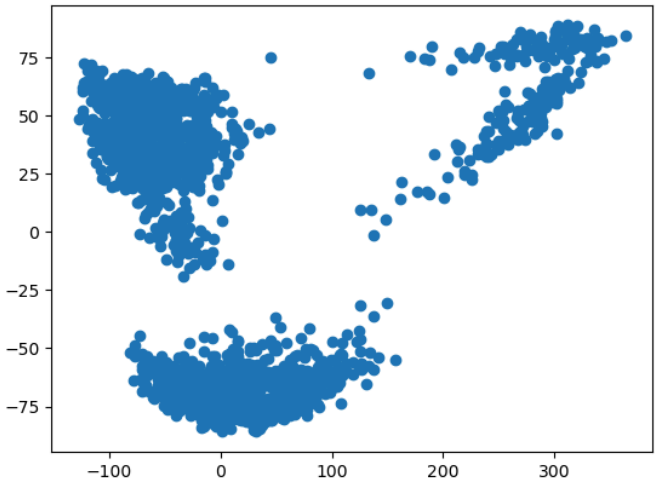
1. (4 Points) Using your clustering method(s) of choice, find a suitable clustering for the cells. Briefly explain how you chose the number of clusters by appropriate visualizations and/or numerical findings. (to cluster cells into the subcategories instead of categories)

Solution:

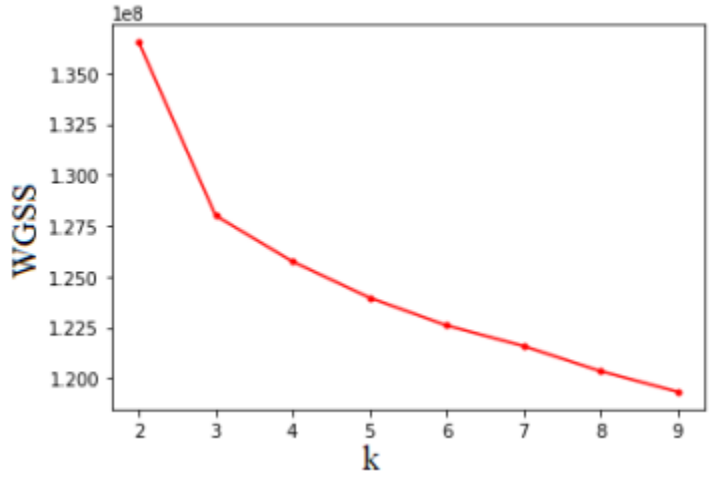
As before, let's use the k-means method to find suitable clusters for the cells. This time, we'll run k-means on the entire dataset, not just the projection on the top principal components. Before running k-means, we need to choose the number of clusters, k.

We can utilize the visualization of the data projected onto the first two principal components (as shown above) to determine an appropriate number of clusters. The visualization shows three well-separated clusters, so we can start with k=3.

```
X_new = PCA(n_components=2).fit_transform(X)
plt.scatter(X_new[:,0], X_new[:,1])
```



Alternatively, we can use the elbow method to determine the appropriate number of clusters.



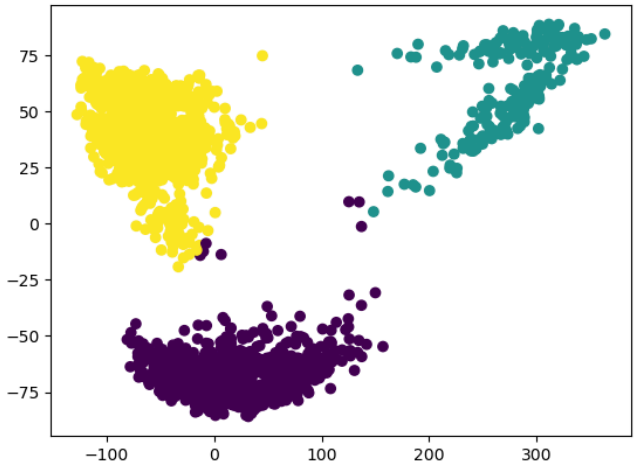
As we can see from the above plot, there is an elbow at k=3, so we can choose the number of clusters as 3.

Once we have chosen k, we can run k-means clustering on the entire dataset.

```
kmeans = KMeans(n_clusters=3, random_state=0).fit(X)
```

and then visualize the clustered dataset by projecting on the first 2 PCs, as before and color coding with cluster labels.

```
X_new = PCA(n_components=2).fit_transform(X)
plt.scatter(X_new[:,0], X_new[:,1], c=kmeans.labels
```



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 ℓ1, ℓ2, or elastic net, and separate the data into training and validation or use crossvalidation to select your model. Report your choice of regularization parameter and validation performance.

Multi-class logistic regression: *When the underlying data has more than two classes involved, we can adapt Logistic Regression which is usually used for binary classification by one-versus-rest approach. In particular, if we have classes, we train separate binary classification models using logistic regression. Each classifier for is trained to determine the probability of a data point belonging to class . To predict the class for a new point , we run all classifiers on and choose the class with the highest probability, i.e.,*

$$\hat{y} = \text{argmax}_{k \in \{1,\ldots,K\}} f_k(x).$$

Solution:

Scikit-learn's LogisticRegressionCV implements logistic regression as an optimization problem, binary class ℓ2 penalized logistic regression minimizes the following cost function:

$$\min_{w,c} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \log(\exp(-y_i(X_i^T w + c)) + 1).$$

Where C is the regularization parameter: no regularization amounts to setting C to a very high value. The following python code shows how it can be trained on the original data with clustering labels as the target labels. Each of the values in Cs (we used 1 and 10 here) describes the inverse of regularization strength and the grid-search over the Cs is performed to find the best C value for each class (we have 3 classes here) with CV (we used 5 folds).

```
from sklearn.linear_model import LogisticRegressionCV
kmeans = KMeans(n_clusters=3, random_state=0).fit(X)
y = kmeans.labels_
clf = LogisticRegressionCV(cv=5, Cs=[1, 10],
solver='liblinear',random_state=0).fit(X, y)
clf.scores_
# {0: array([[1. , 1. ],
            [1. , 1. ],
            [1. , 1. ],
            [1. , 1. ],
            [0.98845266, 0.98845266]])
1: array([[0.97465438, 0.97465438],
          [1. , 1. ],
          [1. , 1. ],
          [1. , 1. ],
```

```
          [0.99769053, 0.99769053]]),
2: array([[1. , 1. ],
          [1. , 1. ],
         [1. , 1. ],
         [1. , 1. ],
         [0.98614319, 0.98614319]])}
clf.C_ # best C values for 3 classes
# array([1, 1, 1])
```

As can be seen from above, the CV scores on each of the 5 folds for all of the 3 classes are above 97%.

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 log 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.

Solution:

Using the top 100 selected features (determined by the highest coefficient values, obtained through ranking coefficients based on their maximum absolute value across classes), we trained a logistic regression classifier on the training data. Upon evaluating this classifier on the test data, we achieved approximately 91.1% accuracy on the test evaluation dataset.

```
def evaluate(X_train, y_train, X_test, y_test, feature_indices):
  X_train = X_train[:,feature_indices]
  clf2 = LogisticRegressionCV(cv=5, solver='liblinear', random_state=0,
max_iter=1000).fit(X_train, y_train)
  X_test = X_test[:,feature_indices]
  print(clf2.score(X_test, y_test))
```

```
sel_feature_indices = np.argsort(np.max(np.abs(clf.coef_), axis=0))[-100:]
evaluate(X_train, y_train, X_test, y_test, sel_feature_indices)
# 0.9106498194945848
```

Baseline Model 1

If we use 100 random features and train a logistic regression model with those features and evaluate on test data, we get around 25.5% accuracy.

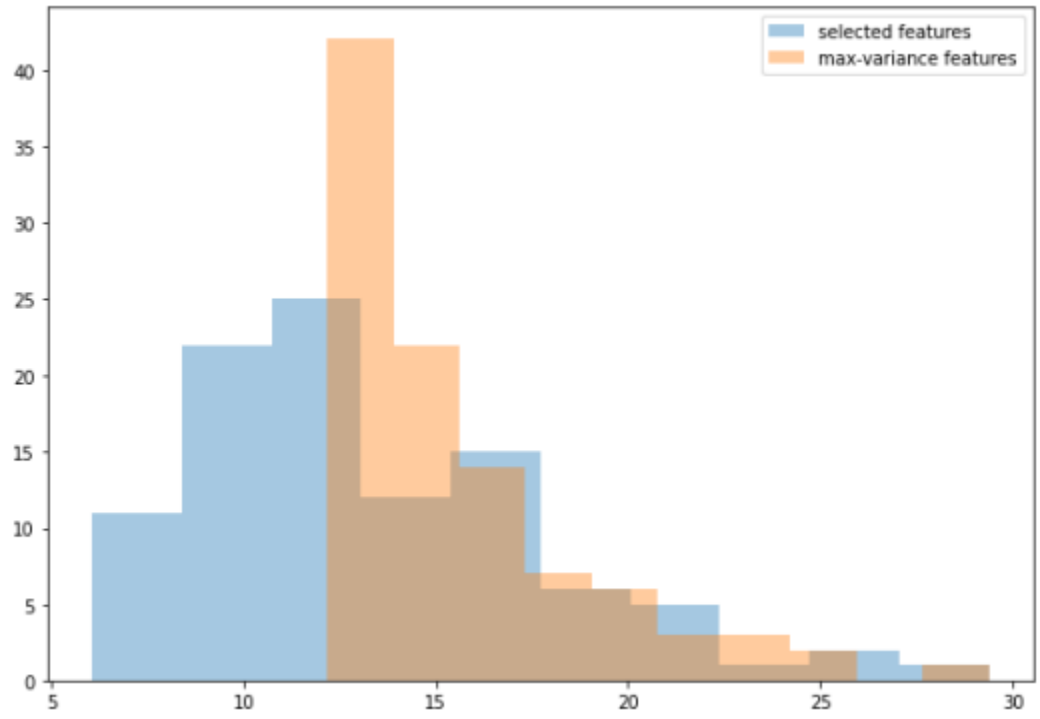
```
rnd_feature_indices = np.random.choice(range(clf.coef_.shape[1]), 100)
evaluate(X_train, y_train, X_test, y_test, rnd_feature_indices)
# 0.2545126353790614
```

Baseline Model 2

If we use 100 high-variance features and train a logistic regression model with those features and evaluate on test data, we get around 92.5% accuracy.

```
maxvar_feature_indices = np.argsort(np.var(X, axis=0))[-100:]
evaluate(X_train, y_train, X_test, y_test, maxvar_feature_indices)
# 0.9250902527075813
```

The following histogram shows the comparison in the variances of the selected features and the highest-variance features.



- **Problem 2: Influence of Hyper-parameters**

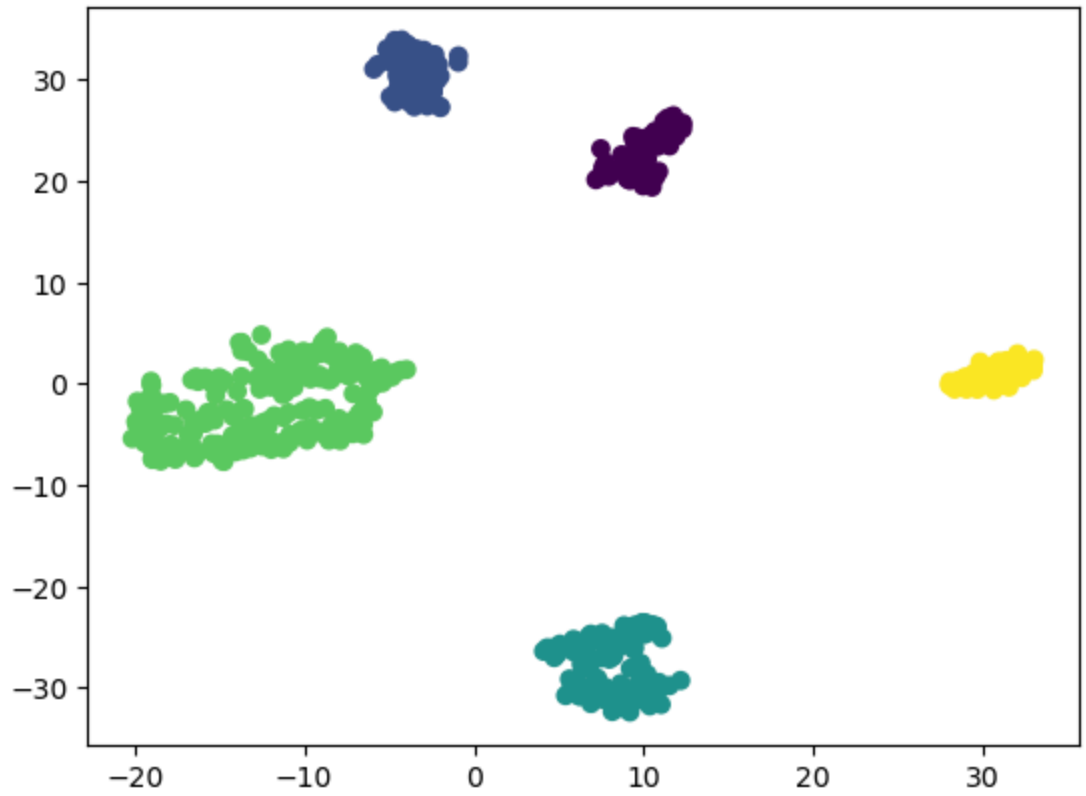
The hyper-parameter choices used in data analysis techniques can have a large impact on the inferences made. As you may have encountered, finding the best choice of parameter such as perplexity in T-SNE or the number of clusters can be an ambiguous problem. We will now investigate the sensitivity of your results to changes in these hyper-parameters, with the goal of understanding how your conclusions may vary depending on these choices.

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?

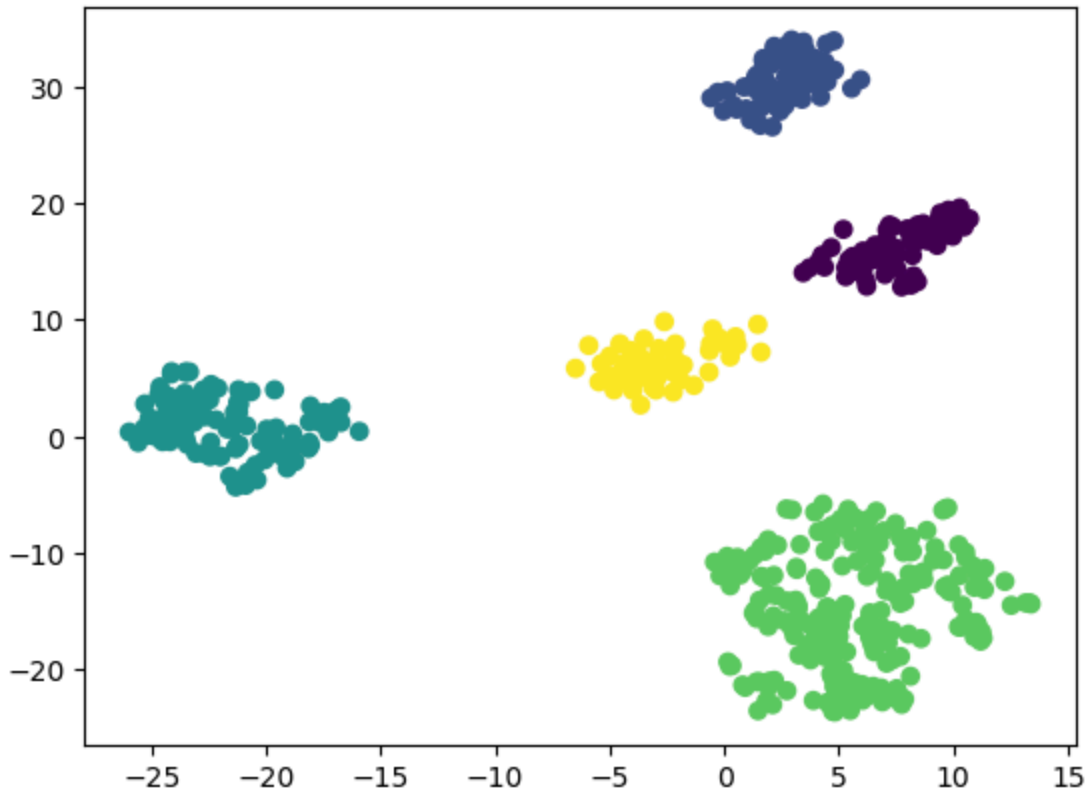
Solution:

We can see from the next visualization that as we increase the number of top PCs to project the dataset on, adding more PCs explain the variance of the data better (capturing the signal more, although at the risk of adding noise too), as a result, we can see that the resulting T-SNE plot captures the 3 main types of the brain cells and creates more and more well-separated and compact clusters in two-dimension using that we can readily identify the 3 types of cell. With low number of top PCs selected, the percentage variance in the data explained is lesser, that's why it captures more groups (cell subgroups) instead of identifying the three primary cell types,

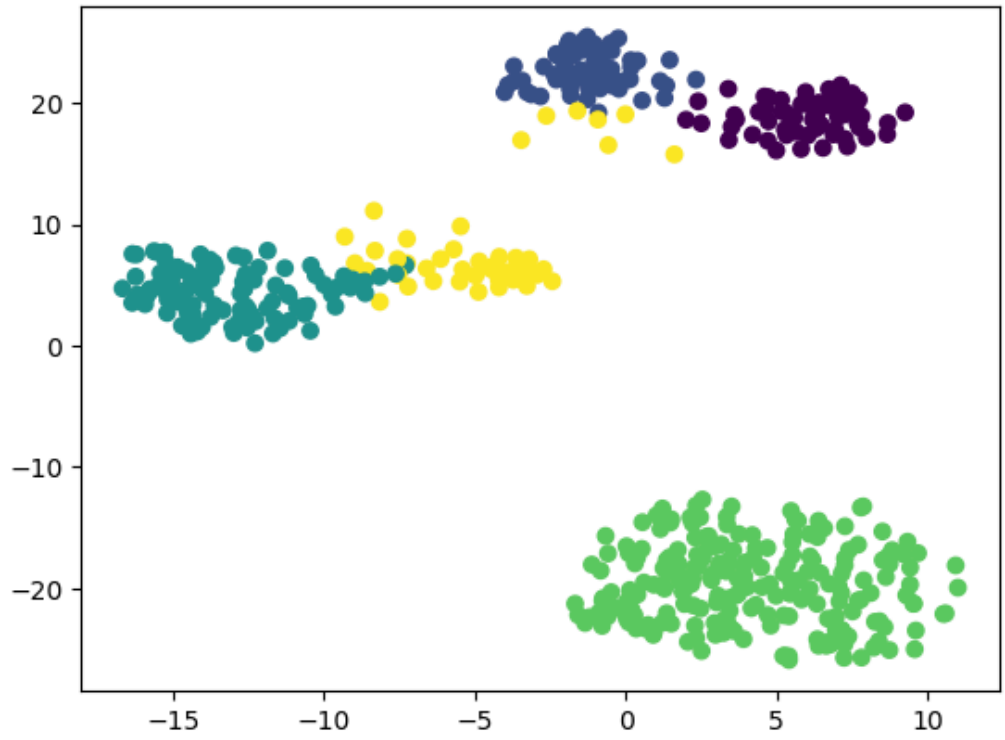
Number of PCs: 10



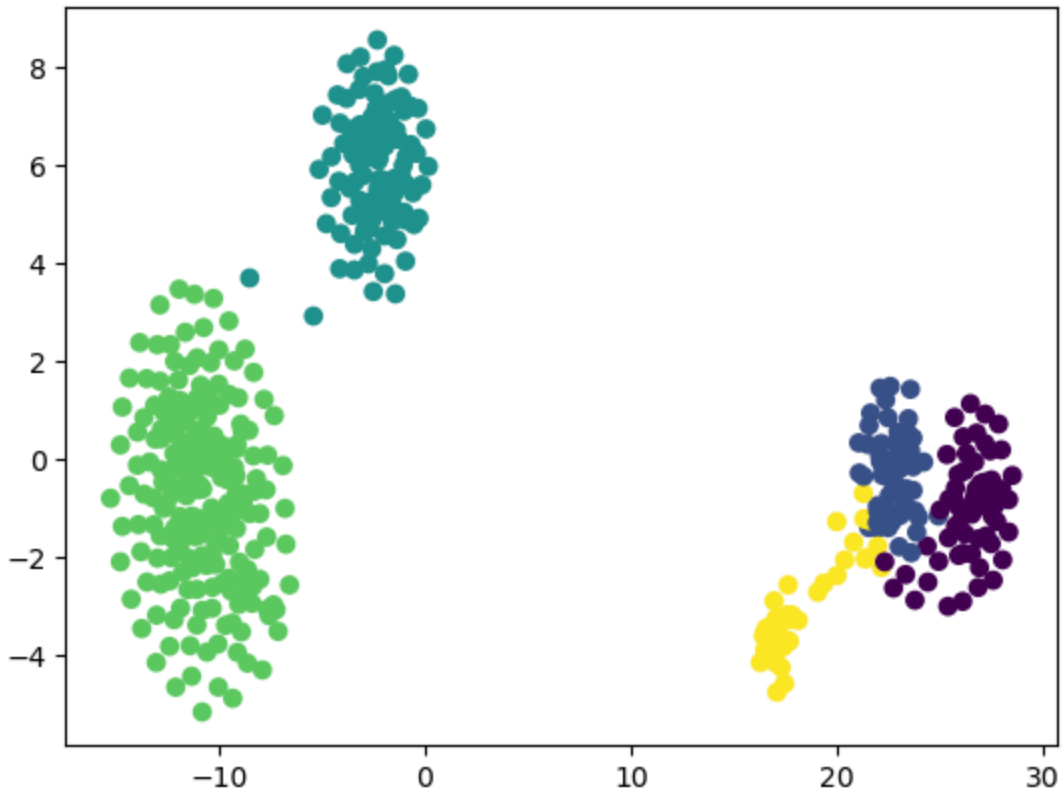
Number of PCs: 50



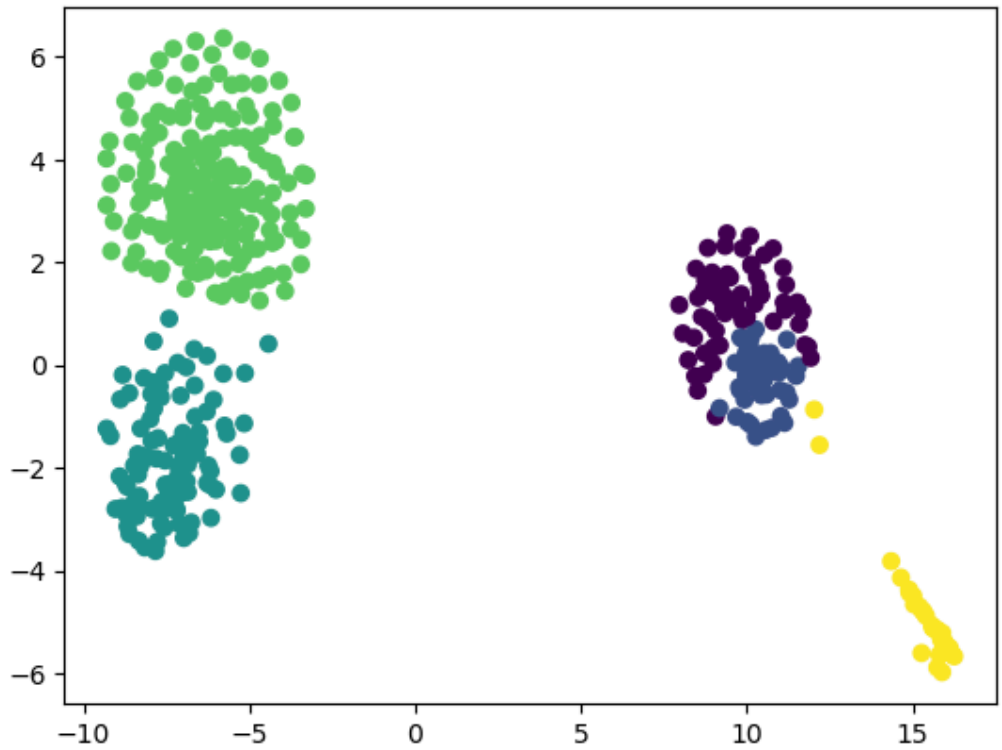
Number of PCs: 100



Number of PCs: 250



Number of PCs: 500



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.

Some possible choices of hyper-parameters are:

Category A (visualization):

- *T-SNE perplexity*
- *T-SNE learning rate*
- *T-SNE early exaggeration*
- *T-SNE initialization*
- *T-SNE number of iterations/convergence tolerance*

Category B (clustering/feature selection):

- *Effect of number of PC's chosen on clustering*
- *Type of clustering criterion used in hierarchical clustering (single linkage vs ward, for example)*
- *Number of clusters chosen for use in unsupervised feature selection and how it affects the quality of the chosen features*
- *Magnitude of regularization and its relation to your feature selection (for example, does under or over-regularizing the model lead to bad features being selected?)*
- *Type of regularization ($\ell 1$, $\ell 2$, elastic net) in the logistic regression step and how the resulting features selected differ*

For visualization hyper-parameters, provide substantial visualizations and explanation on how the parameter affects the image.

For clustering/feature selection, provide visualizations and/or numerical results which demonstrate how different choices affect the downstream visualizations and feature selection quality.

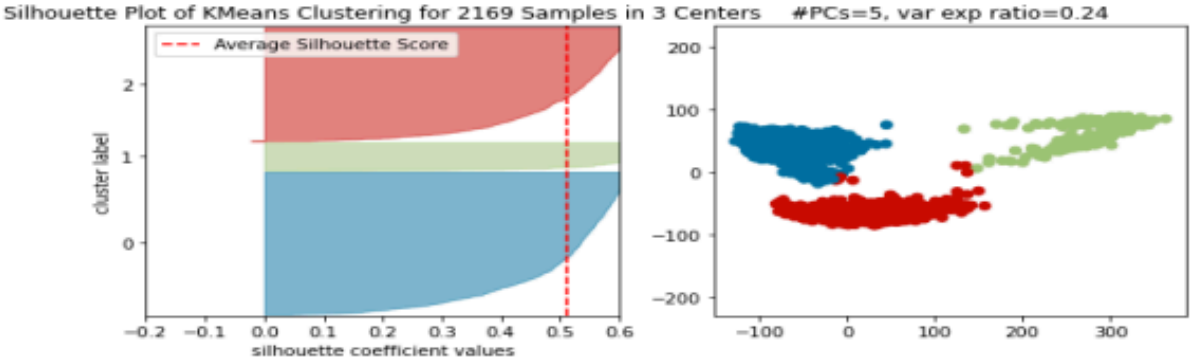
Provide adequate explanations in words for each of these visualizations and numerical results.

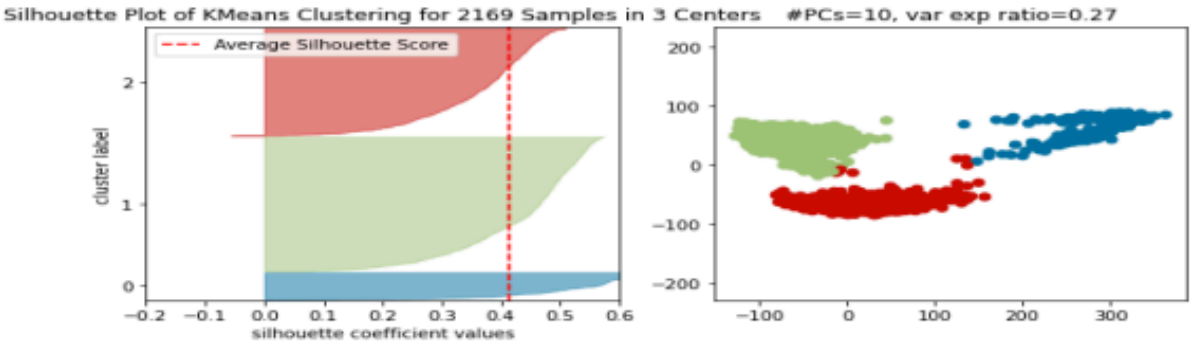
Solution:

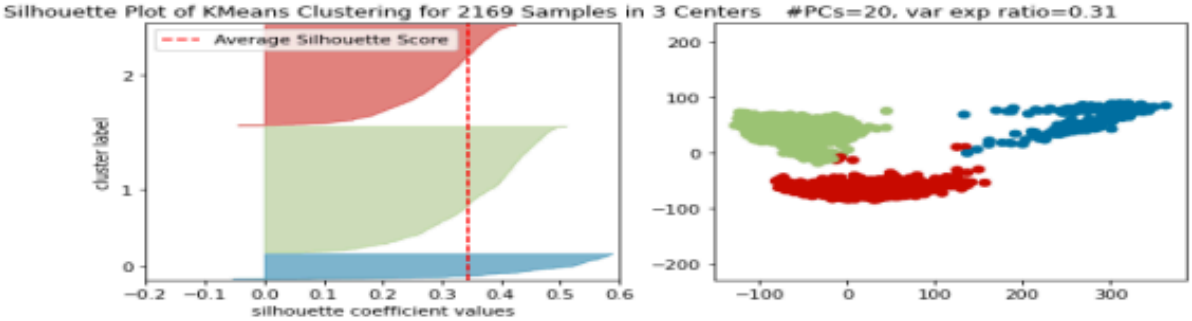
We will utilize the dataset p2_unsupervised for this analysis.

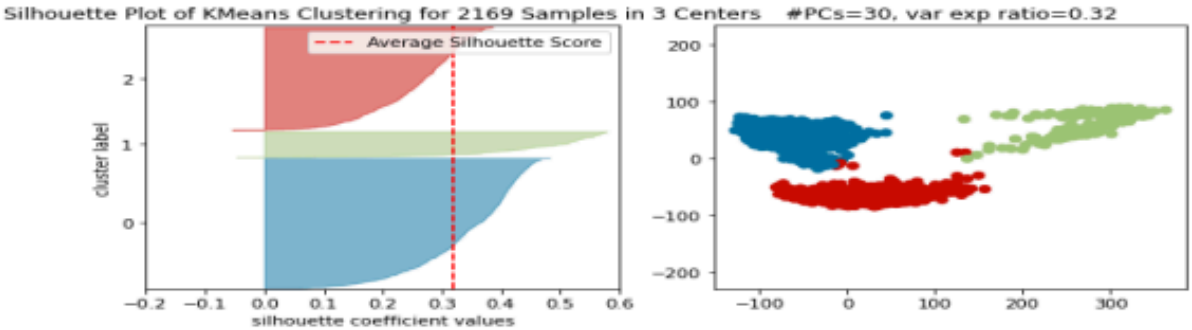
1. Impact of the number of principal components (PCs) selected on clustering (Category B):

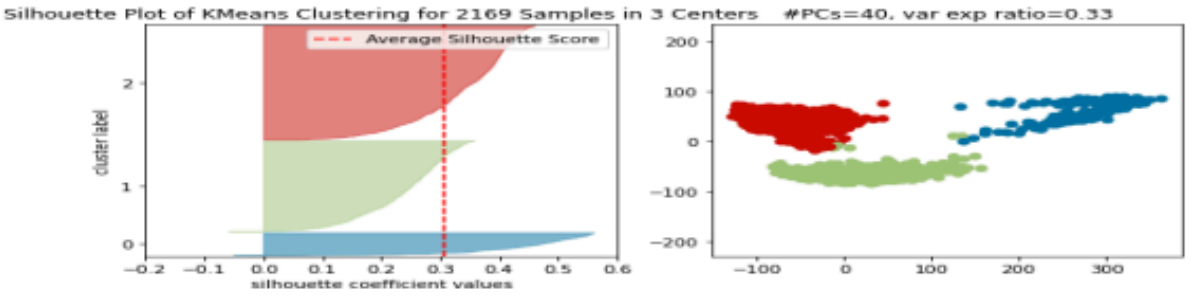
   We selected the top k PCs for varying values of k (ranging from 5 to 50), applied k-means clustering to cluster the projected dataset based on these top k PCs, and calculated the average silhouette score to assess cluster quality for each k value. This process is illustrated in the following visualization.

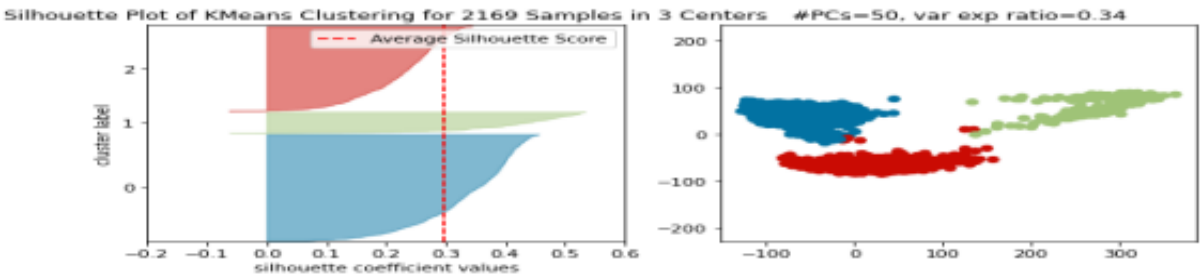

Silhouette Plot of KMeans Clustering for 2169 Samples in 3 Centers #PCs=5, var exp ratio=0.24
Average Silhouette Score
cluster label
silhouette coefficient values


Silhouette Plot of KMeans Clustering for 2169 Samples in 3 Centers #PCs=10, var exp ratio=0.27
Average Silhouette Score
cluster label
silhouette coefficient values

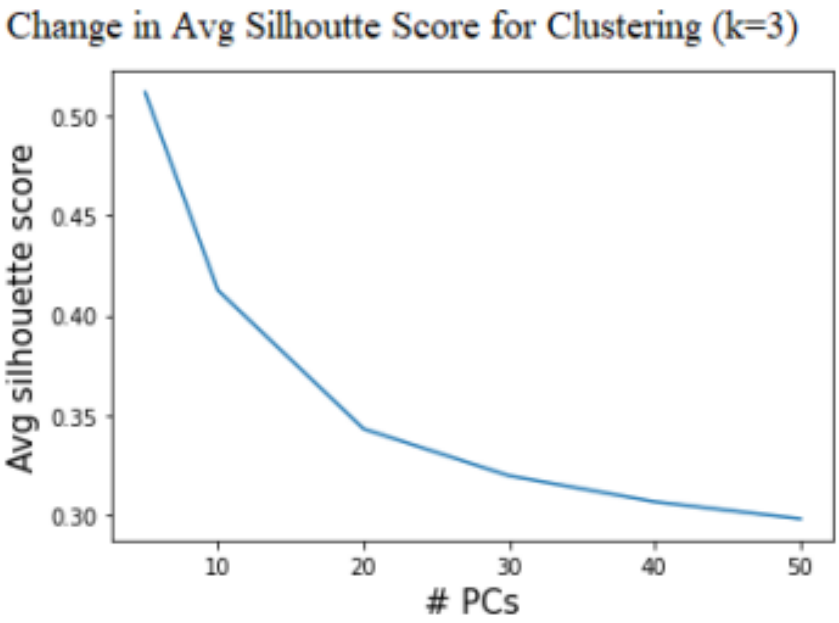

Silhouette Plot of KMeans Clustering for 2169 Samples in 3 Centers #PCs=20, var exp ratio=0.31
Average Silhouette Score
cluster label
silhouette coefficient values


Silhouette Plot of KMeans Clustering for 2169 Samples in 3 Centers #PCs=30, var exp ratio=0.32
Average Silhouette Score
cluster label
silhouette coefficient values


Silhouette Plot of KMeans Clustering for 2169 Samples in 3 Centers #PCs=40, var exp ratio=0.33
Average Silhouette Score
cluster label
silhouette coefficient values

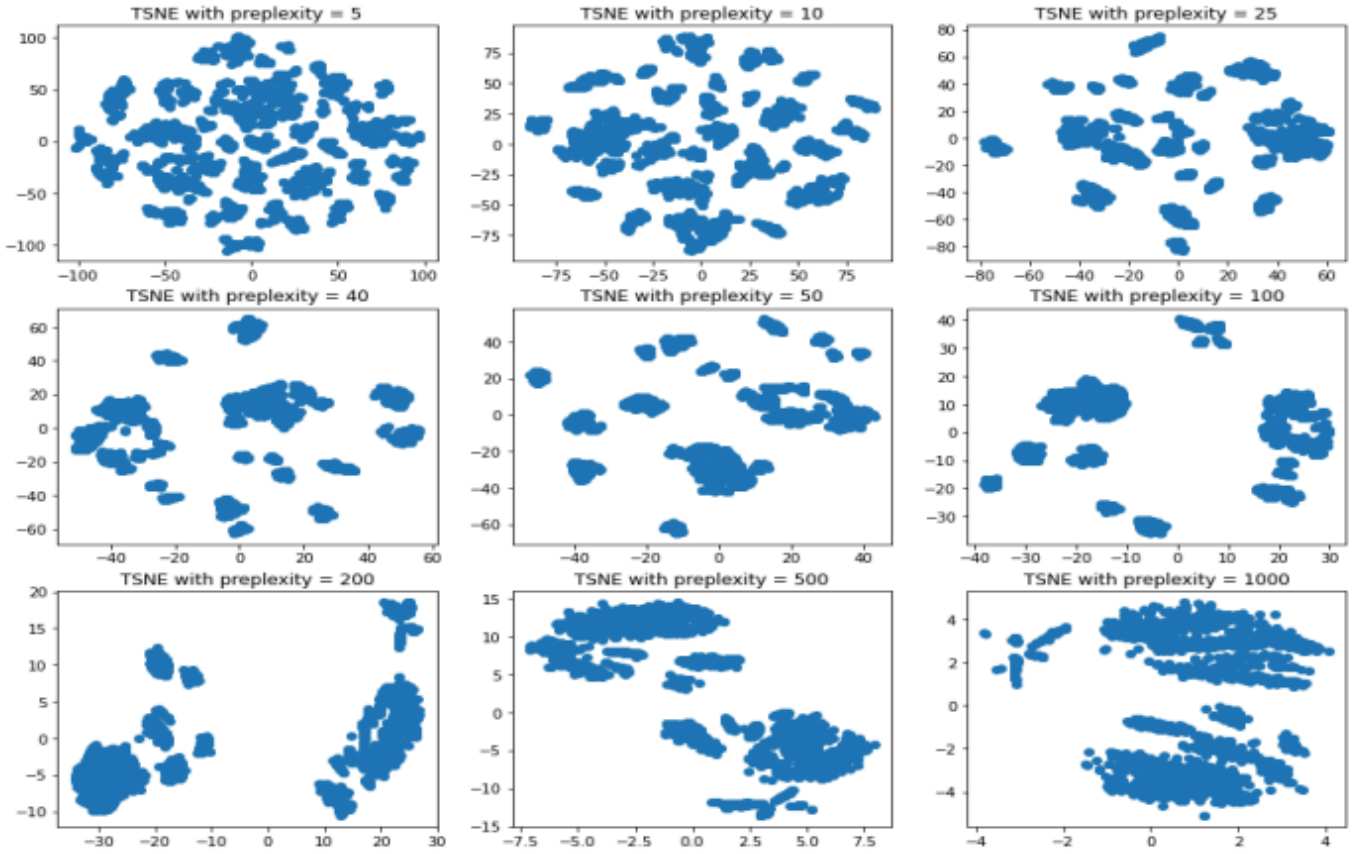


As can be seen from the above plots and also from the below figure, as we increase the number of PCs, the dimension of the data to be clustered increases, as well as more and more noise get included into the projected data, which increases WGSS and decreases the cluster quality, reducing the average silhouette score.



2. T-SNE perplexity (Category A):

Next, we fix the number of PCs onto which the data will be projected (using the top 50 PCs). To visualize the data, we vary the perplexity parameter for T-SNE from 5 to 1000. As illustrated in the following figure, lower perplexity values result in numerous small clusters in the T-SNE visualization, while higher perplexity values lead to fewer, larger clusters. Perplexity intuitively represents the number of nearest neighbors that T-SNE considers; a higher perplexity allows each point to perceive more neighbors, causing T-SNE to group them closely together in the 2D manifold.


TSNE with preplexity = 5
TSNE with preplexity = 10
TSNE with preplexity = 25
TSNE with preplexity = 40
TSNE with preplexity = 50
TSNE with preplexity = 100
TSNE with preplexity = 200
TSNE with preplexity = 500
TSNE with preplexity = 1000