Written Report – 6.419x Module 2

Problem 2: Larger unlabelled subset

Name: Sandipan Dey

Include your answers to all parts below of Problem 2 in your written report.

The written response section of problem is 26 points.

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.

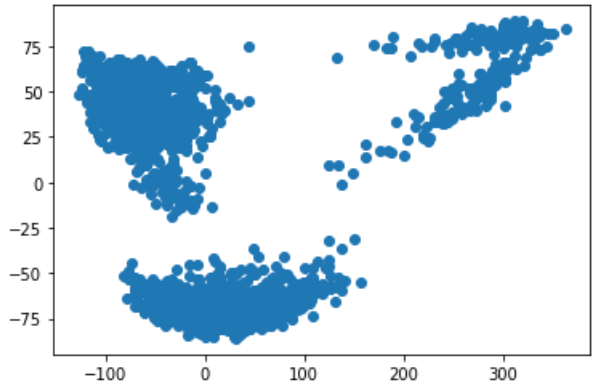
As in Problem 1, we recommend using PCA before running T-SNE or clustering algorithms, for quality and computational reasons.

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

Solution:

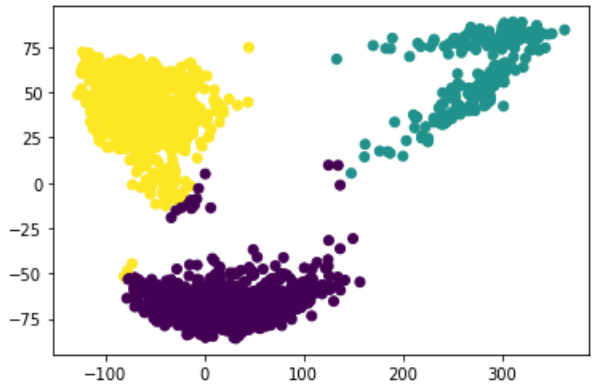
The following visualization shows the projection of the dataset on first two orthonormal principal components (capturing the top 2 maximum variance directions in the dataset). As can be seen from the below figures, there are clearly 3 distinct groups to which the cells belong to and the cells belonging to the groups differ greatly - they form 3 well-separated and well-formed clusters.

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



We can obtain the clusters with k-means clustering algorithm on the projected space: one cluster is compact and centered around (-65, 40), another one is elongated along the x-axis and centered around (13,-65), the third one being a little scattered and centered around (272, 60), in terms of the (PC1, PC2) coordinates, as can be seen from the below figure.

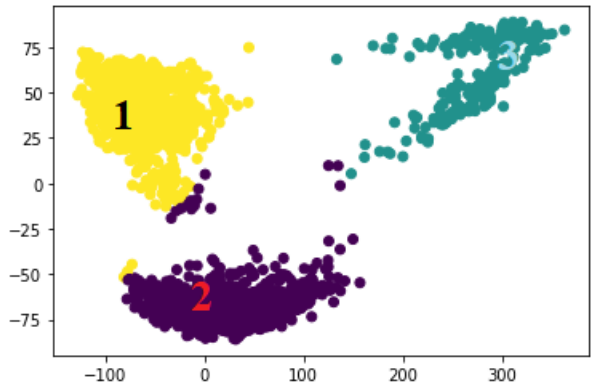
```
kmeans = KMeans(n_clusters=3, random_state=1).fit(X_new)
plt.scatter(X_new[:,0], X_new[:,1], c=kmeans.labels_)
```



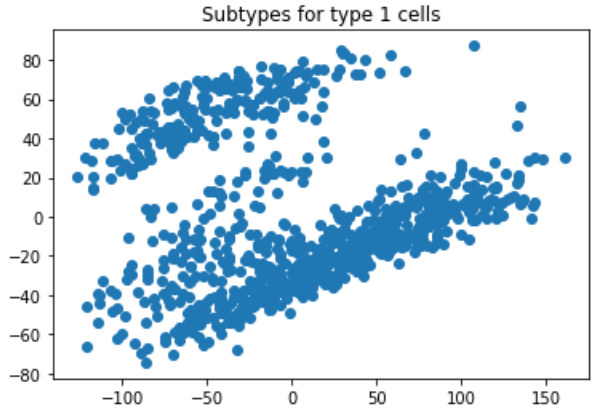
2. (4 points) Provide at least one visualization which supports the claim that within each of the three types, there are numerous possible sub-types for a cell. In your visualization, highlight which of the three main types these sub-types belong to. Again, explain how your visualization supports the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

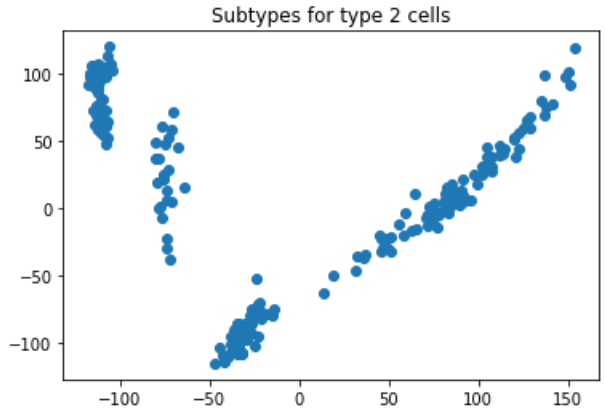
Solution:

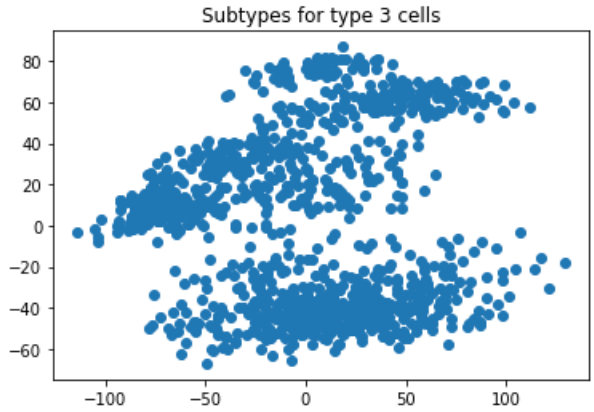
Let's subset the data by the three clusters (types) obtained above (as shown in the figure below and numbered) and run k-means algorithm again on each individual groups to find clusters (sub-types).




```
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

3. 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.

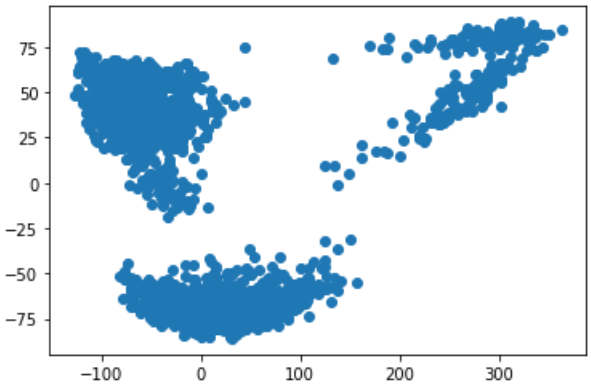
4. (4 points) Using your clustering method(s) of choice, find a suitable clustering for the cells. Briefly explain how you chose the number of clusters by appropriate visualizations and/or numerical findings.

Solution:

As earlier, let's use k-means method to find a suitable clustering for the cells. Note that this time we shall run k-means on the entire dataset, not on the one projected on top PCs. But, before running k-means we need to choose k: the number of clusters.

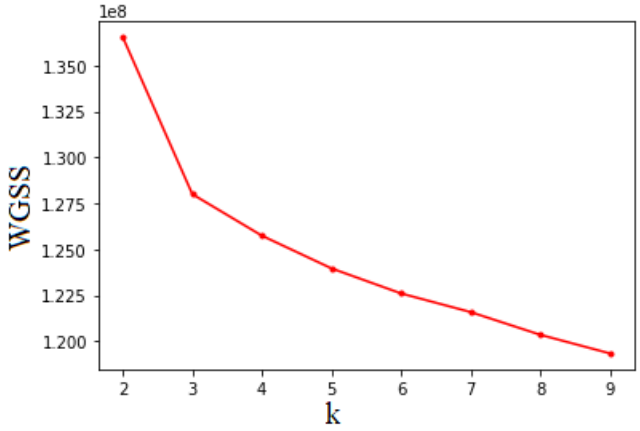
- We can use visualization (of the data projected on the first two PCs, as shown above) to choose the appropriate number of clusters, as can be seen, there are 3 well-separated clusters, so we can start with k=3

```
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 choose the appropriate number of clusters

```
WGSS = []
for i in range(2, 10):
   kmeans = KMeans(n_clusters=i, random_state=0).fit(X)   # alternatively use X projected on top-k PCs here to make it faster
   WGSS.append(kmeans.inertia_)
plt.plot(range(2,10), WGSS, 'r.-')
```



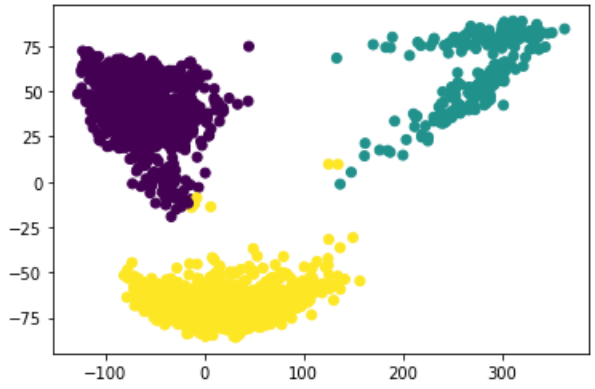
As we can see from the above plot, there is an elbow at k=3, so we can choose the number of clusters as 3.

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_)
```



5. (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.

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.,

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

Python tip: You may use liblinear solver in LogisticRegression or LogisticRegressionCV for one-versus-rest logistic regression.

Note: Recall that the p2_unsupervised_reduced and p2_evaluation_reduced folders contain datasets with a reduced number of genes, in case you are unable to run some of the procedures on the larger versions. In particular, a full logistic regression could take 1 or 2 GB of memory to run.

Solution:

Scikit-learn's LogisticRegressionCV implements logistic regression as an optimization problem, binary class ℓ_2 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 an 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.

   Note: The histogram should show the distribution of the variances of features selected by both methods. You could show the comparison by overlaying both histograms in the same plot.

   Hint: Refer to the recitation for some guidance if necessary.

   Solution:

   With top 100 selected features (corresponding to top 100 coefficient values, obtained by ranking the coefficients using the maximum absolute value over classes), training a logistic regression classifier with these features on training data and evaluating the performance over the test data, we get around 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.

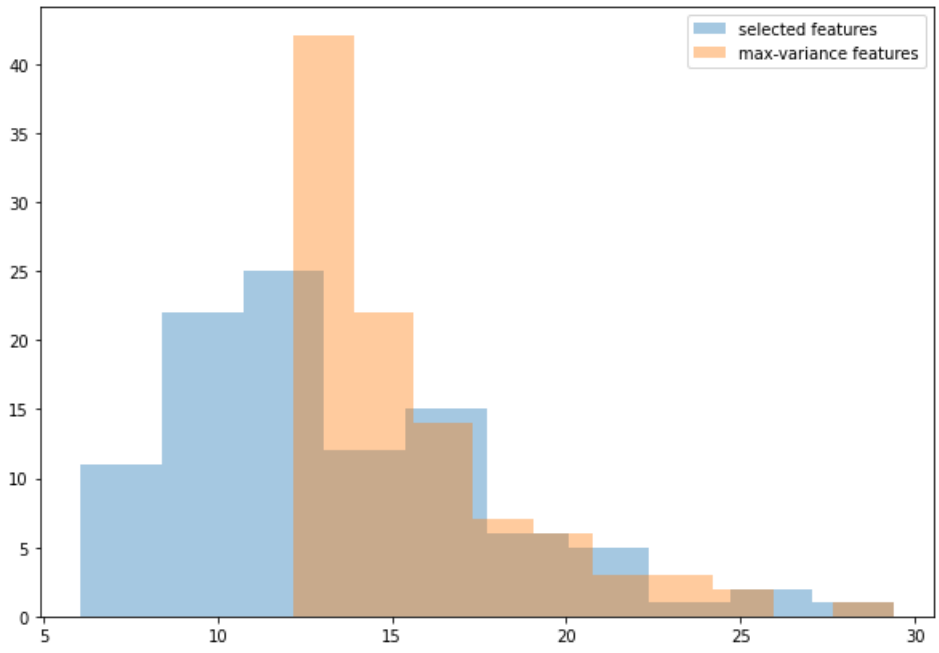
```
rnd_feature_indices = np.random.choice(range(clf.coef_.shape[1]), 100)
evaluate(X_train, y_train, X_test, y_test, rnd_feature_indices)
# 0.2545126353790614
```

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 3: Influence of Hyper-parameters (Written Report)

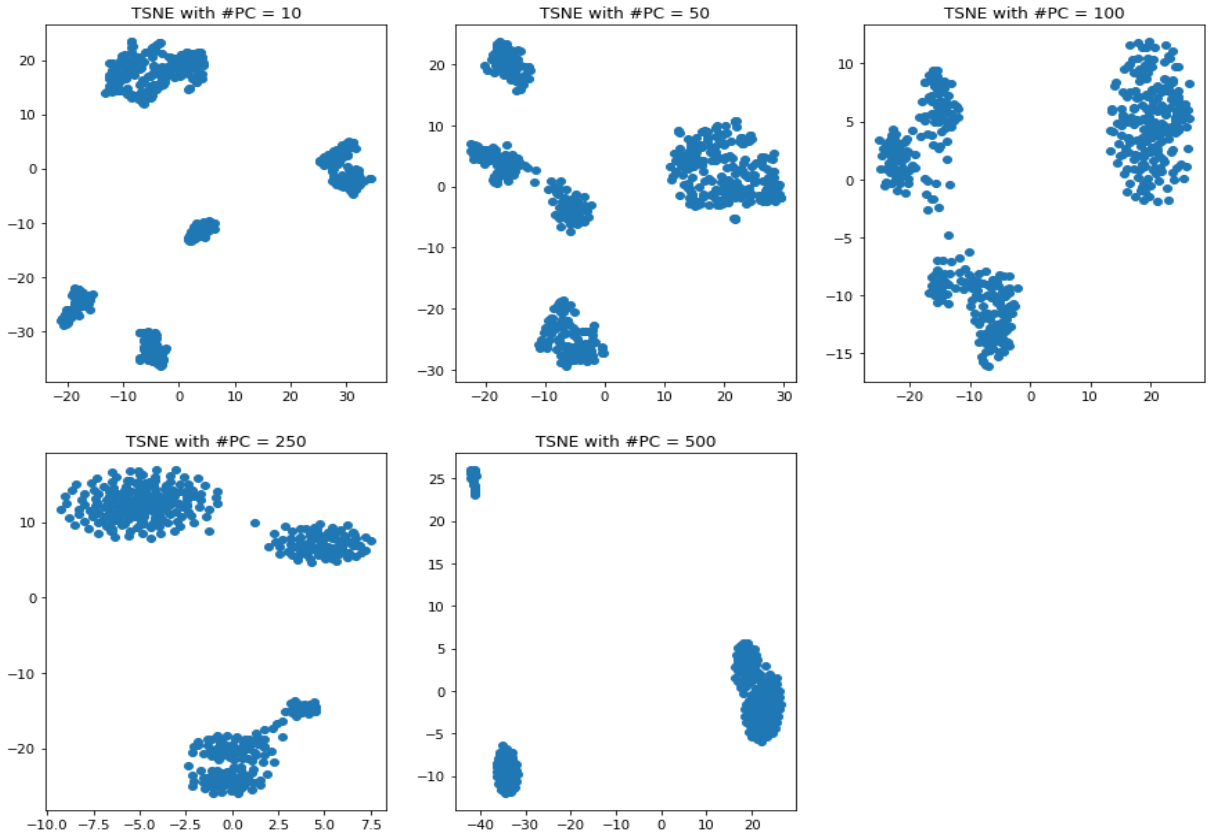
Include your answers to all parts of problem 3 in your written report. This problem is worth 16 points.

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.



2. (13 points) Pick three hyper-parameters below 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 (ℓ_1, ℓ_2, elastic net) in the logistic regression step and how the resulting features selected differ

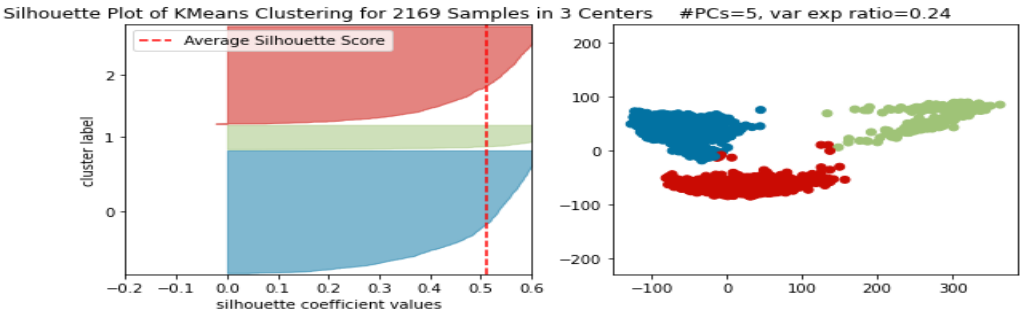
   For visualization hyper-parameters, provide substantial visualizations and explanation on how the parameter affects the image. For clustering/feature selection, provide visualizations and/or numerical results which demonstrate how different choices affect the downstream visualizations and feature selection quality. Provide adequate explanations in words for each of these visualizations and numerical results.

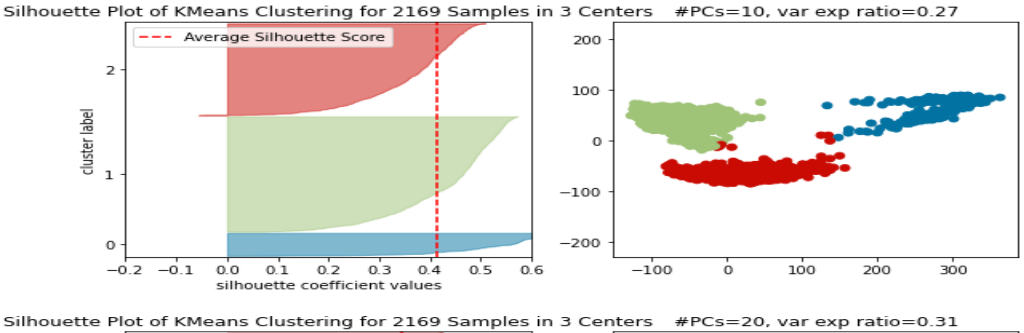
   Solution:

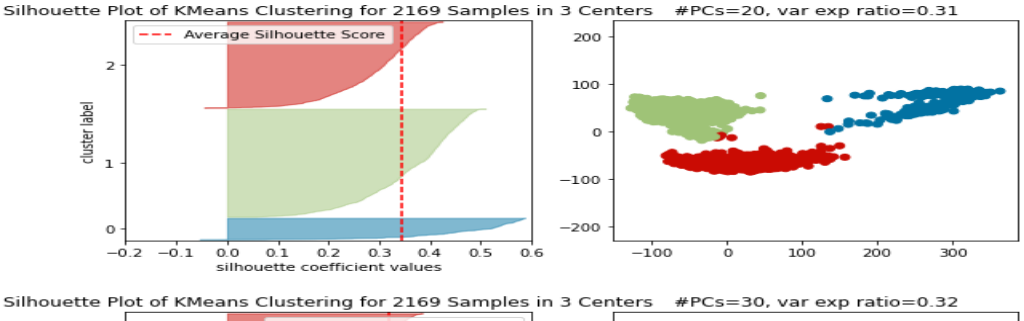
   We shall use the dataset *p2_unsupervised* for this problem.

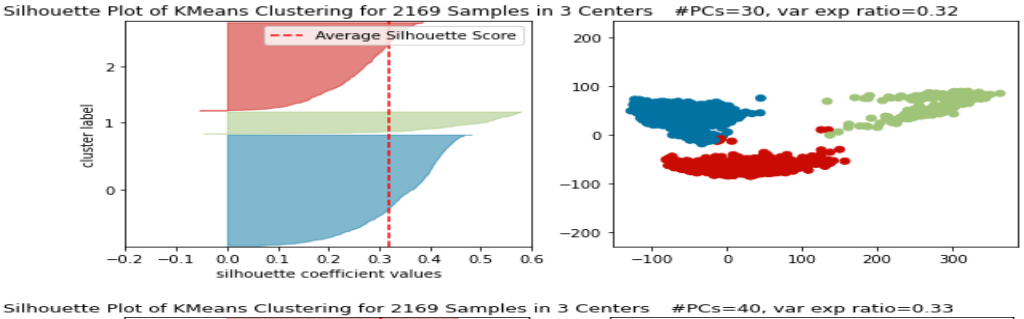
   1. Effect of number of PC's chosen on clustering (Category B)

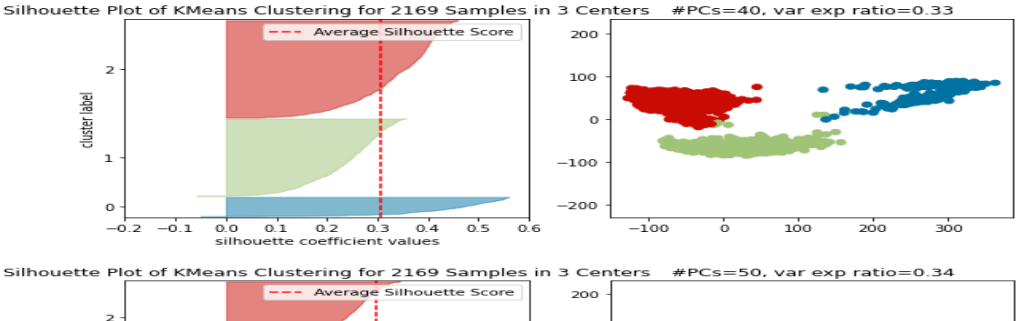
   Chosen top k PCs for different values of k (from 5 to 50), used k-means clustering to cluster the projected dataset on top k PCs and computed the average silhouette score to measure the cluster quality for each k, as shown in the following visualization.

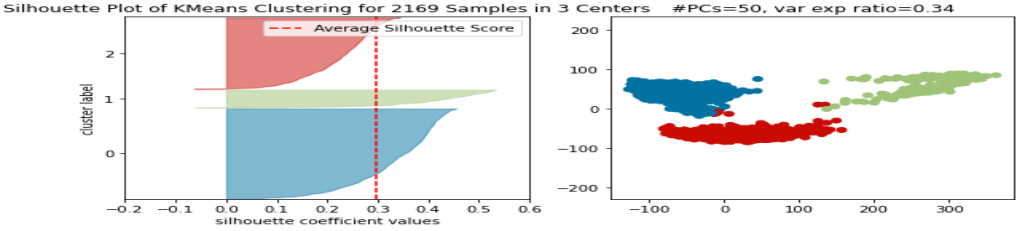

Silhouette Plot of KMeans Clustering for 2169 Samples in 3 Centers #PCs=5, var exp ratio=0.24
Average Silhouette Score
cluster label
silhouette coefficient values


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

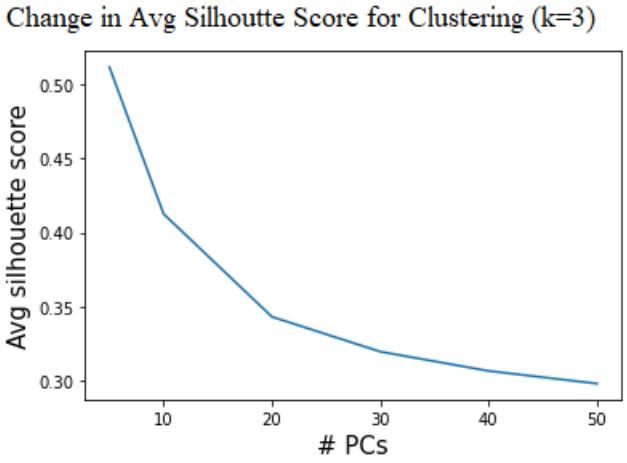

Silhouette Plot of KMeans Clustering for 2169 Samples in 3 Centers #PCs=20, var exp ratio=0.31
Average Silhouette Score
cluster label
silhouette coefficient values


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


Silhouette Plot of KMeans Clustering for 2169 Samples in 3 Centers #PCs=50, var exp ratio=0.34
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)

Now, let's fix the number of PCs the data to be projected on (use top 50 PCs). For visualization, let's vary the perplexity for T-SNE from 5 to 1000. As can be seen from next figure, for small perplexity values, numerous tiny groups appear in the T-SNE visualization and with large perplexity values, smaller number of groups of larger size appear in the visualization, since perplexity intuitively means number of nearest neighbors T-SNE can sense. With higher value of perplexity a point can sense many of the neighbors and T-SNE places all of them in proximity in the 2D manifold.