SUPERVISED LEARNING

Approach: We implemented different classification models using data preprocessing techniques like scaling(min-max/robust), feature selection(recursive feature selection), dimensionality reduction (PCA), etc. We then evaluate these models based on the model's accuracy and misclassification error.

Model 1: K Nearest Neighbors

We chose to implement KNN, because it's a non-parametric classification algorithm that does not make assumptions about data distribution. Our approach involves hyperparameter tuning, using a hyperparameter grid for K values (k = 5:15) and cross-validation for fold size (5 to 30). We store predictions for each configuration and track the one with the best mean accuracy. We select the best K value and make predictions using it.

Model 2: Naive Bayes

We use Naive Bayes with a mean correlation of 0.00218633. After loading the dataset, we split it into training/testing sets and train the model using the train() function from caret with cross-validation. We select the best fold combination based on accuracy, specify different numbers of folds (5 and 10) using a train control list, and print the result. Finally, we predict class labels for the testing set with the best model using predict().

Model 3: Random Forest with Recursive Feature Elimination

Random Forest can handle a large number of features without the need for feature selection or dimensionality reduction techniques making it an obvious choice. To implement this technique,

the data is prepared for cross-validation, and the rf() function is used to model and specify hyperparameters. The best fold combination is selected based on the highest accuracy, and the predict() function is used to predict class labels for the testing set using the best model.

Model 4: **Boosting & Ensemble**

We use boosting to create a stronger prediction model by combining weak or base models. After loading the data into a data frame and preparing it for cross-validation, we use the xgboost() function in train() to model the data and select the best hyperparameters with cross-validation, using a train control list. We select the best fold combination based on accuracy, print the result, and predict the class labels for the testing set using the best model with the predict() function. Boosting trains base models sequentially to fix previous models' flaws and improve overall prediction accuracy. We also tried an ensemble of boosting, rf & knn models.

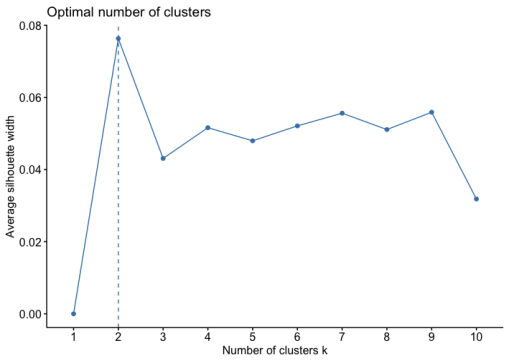
Model	**Accuracy**	**Misclassification**	**Parameters**
KNN	75.26%	24.74%	K=9, fold =13
Naive Bayes	62.3%	37%	fold = 6
Random Forest	68.5%	31.5%	Mtry = 23
Boosting	72%	28%	Max_depth = 9
LDA	57.2%	42.8%	fold = 26
SVM	53.5%	46.5%	sigma = 10 and C = 0.01
Ensemble	72.5%	27.5%	combined

UNSUPERVISED LEARNING

We were given 784-dimensional data and it had 1000 observations. For the given clustering problem, we understand that both scaling and PCA are helpful preprocessing techniques. Upon scaling the data across all its dimensions, we implemented PCA for dimensionality reduction of the data to identify directions of maximum variance. This helps to reduce noise and redundancy in the data, hence making identifying clusters easier. We keep only those dimensions that contribute to 95% variance of our data, this reduces the dimension of our data from 784 to 250.

Following data preprocessing, we need to search for the optimal number of clusters that will help us achieve good generalized performance and interpretability. We use the Elbow method (evaluates the within cluster sum of squared distances for a chosen range of k, and selects k with significant change in slope), Silhouette method (measures how close each point in one cluster is to points in the neighboring clusters, and selects k that maximizes the mean silhouette score). To further evaluate, we also used, Calinsky Criterion (based on within- and between-cluster distances) and Bayesian Information Criterion for Expectation-Maximization. Also, we used the R package 'NbClust', which uses various techniques to determine the optimal number of clusters.

When working with data having 250 dimensions (95% of total data variance), the above listed techniques suggest 2 as the optimal number of clusters.



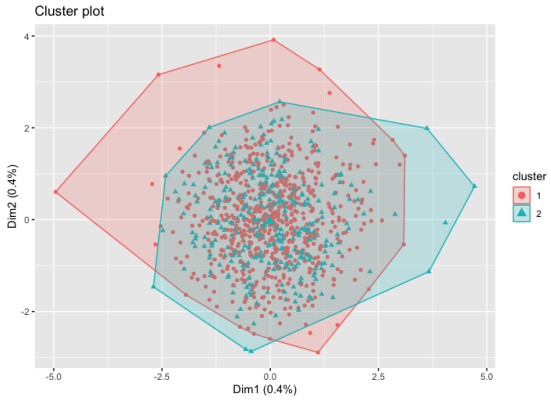

```
**********************************************************************
* Among all indices:
* 9 proposed 2 as the best number of clusters
* 5 proposed 3 as the best number of clusters
* 1 proposed 6 as the best number of clusters
* 2 proposed 7 as the best number of clusters
* 5 proposed 8 as the best number of clusters
* 1 proposed 15 as the best number of clusters

                 ***** Conclusion *****

* According to the majority rule, the best number of clusters is  2

**********************************************************************
```

Next, for the evaluated optimal value of k(=2), we implement clustering algorithms on data to group similar data points together. K-means clustering: The 'kmeans' function clusters the data into 2 clusters based on their similarity in the feature space (in the plot, axis are the two most significant features). Also, used the PAM (Partitioning Around Medoids) algorithm [a more robust form of k-means] and obtained similar clusters. Using the Hierarchical clustering and GMM clustering didn't help as it generated 2 clusters with very unequal proportions of data points in each cluster.



In order to test the general capabilities of the models, we evaluated the codes on data with 85 dimensions (comprising 75% variance of the data), and on the data with just 2 dimensions (the most significant ones)