SIG720

Machine Leaning

Distinction Task Report

Arunkumar Balaraman
S223919051

Table of Contents

TripAdvisor Dataset:

Background: In end user perspective, travel and tourism is mostly explorative in nature and repetitive travels to same locations are minimal. So, travellers have to take decisions regarding their destinations and associated facilities to be consumed without adequate prior or personal knowledge. The best option available is to leverage social media and internet. Tourism recommenders are the best solutions in this scenario.

Dataset file name: tripadvisor_review.csv

Dataset description: User's average feedback/rating information on 10 categories of attractions in East Asia captured from tripadvisor.com. Dataset contains 980 user records with 10 feedback attributes inferred from numerous destination reviews.

The dataset contains 980 rows and 11 columns All the features are numerical except of user variable is object.

No Duplicates entries were found in the dataset.

Numerical features range

Category 1: 0.34 to 3.22
Category 2: 0.00 to 3.64
Category 3: 0.13 to 3.62
Category 4: 0.15 to 3.44
Category 5: 0.06 to 3.30
Category 6: 0.14 to 3.76
Category 7: 3.16 to 3.21
Category 8: 2.42 to 3.39
Category 9: 0.74 to 3.17
Category 10: 2.14 to 3.66

Only for Category 7 where rating minumum is good. Also as the range of ratings is between 0 and 3.76. it indicates standard 0-5 rating scale we seen in the website.

Columns and Datatypes of each column

User ID object
Category 1 float64
Category 2 float64
Category 3 float64
Category 4 float64
Category 5 float64
Category 6 float64
Category 7 float64
Category 8 float64
Category 9 float64
Category 10 float64
User Id is Object and represent the user and it would not be useful for performing clustering hence dropped the column.

Univatite Analysis:
Describe:
Category 1: has a mean of 0.89 & standard deviation of 0.33. Values range from 0.34 to 3.22 & are right skewed.

Category 2: has a mean of 1.35 & standard deviation of 0.48. Values range from 0.0 to 3.64 & are right skewed.

Category 3: has a mean of 1.01 & standard deviation of 0.79. Values range from 0.13 to 3.62 & are right skewed.

Category 4: has a mean of 0.53 & standard deviation of 0.28. Values range from 0.15 to 3.44 & are right skewed.

Category 5: has a mean of 0.94 & standard deviation of 0.44. Values range from 0.06 to 3.3 & are right skewed.

Category 6: has a mean of 1.84 & standard deviation of 0.54. Values range from 0.14 to 3.76 & are right skewed.

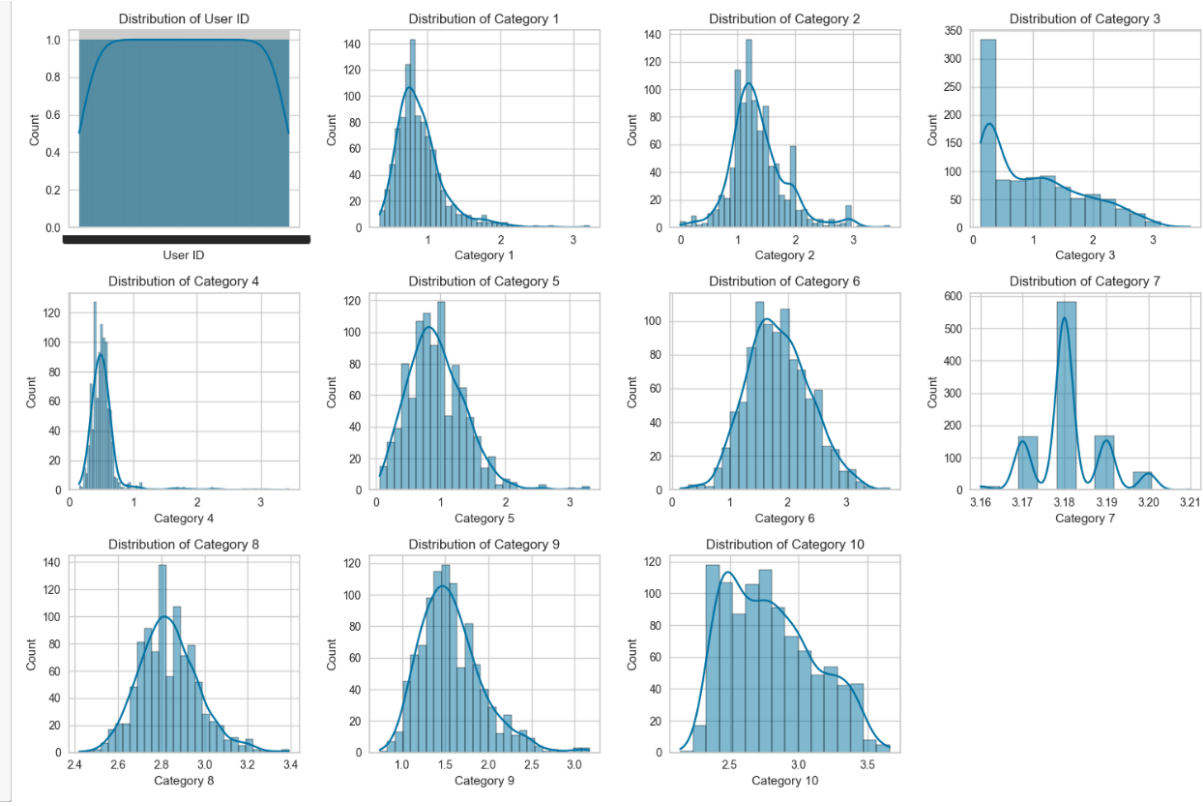
Category 7: has a mean of 3.18 & standard deviation of 0.01. Values range from 3.16 to 3.21 & are right skewed.

Category 8: has a mean of 2.84 & standard deviation of 0.14. Values range from 2.42 to 3.39 & are right skewed.

Category 9: has a mean of 1.57 & standard deviation of 0.36. Values range from 0.74 to 3.17 & are right skewed.

Category 10: has a mean of 2.80 & standard deviation of 0.32. Values range from 2.14 to 3.66 & are right skewed.

Histogram:



Category 1: is right skewed with most of the values around 0.5 to 1.0.

Category 2: is right skewed with majority of the values around 1.0 to 1.5.

Category 3: is right skewed. Most of the values are around 0.5.

Category 4: is right skewed with most of the values below 1.0.

Category 5: is right skewed with majority of the values below 1.0.

Category 6: is right skewed with majority of the values around 2.0.

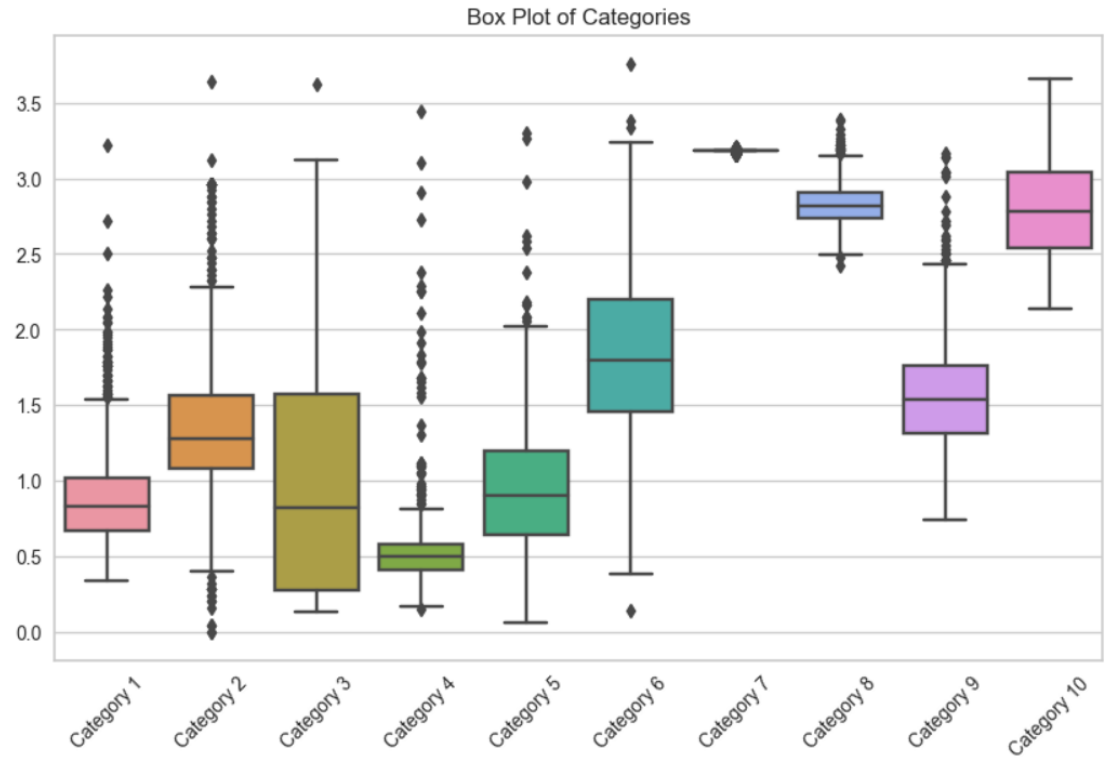
Category 7: shows almost all values around 3.18, indicating limited variance.

Category 8: is slightly left-skewed with majority of the values around 2.8.

Category 9: is right skewed with most values around 1.5.

Category 10: is slightly left-skewed with majority of the values around 3.0.

Boxplot



Category 1: median is slightly above 0.5 with a few outliers on the right.

Category 2: median is close to 1.2. It has few outliers, particularly on the left.

Category 3: median value is close to 0.5. It has several outliers on the right.

Category 4: median is near 0.5. There are few outliers on both ends.

Category 5: median is slightly above 0.5, and there are outliers on the right.

Category 6: median is close to 2.5. It has some outliers on the left.

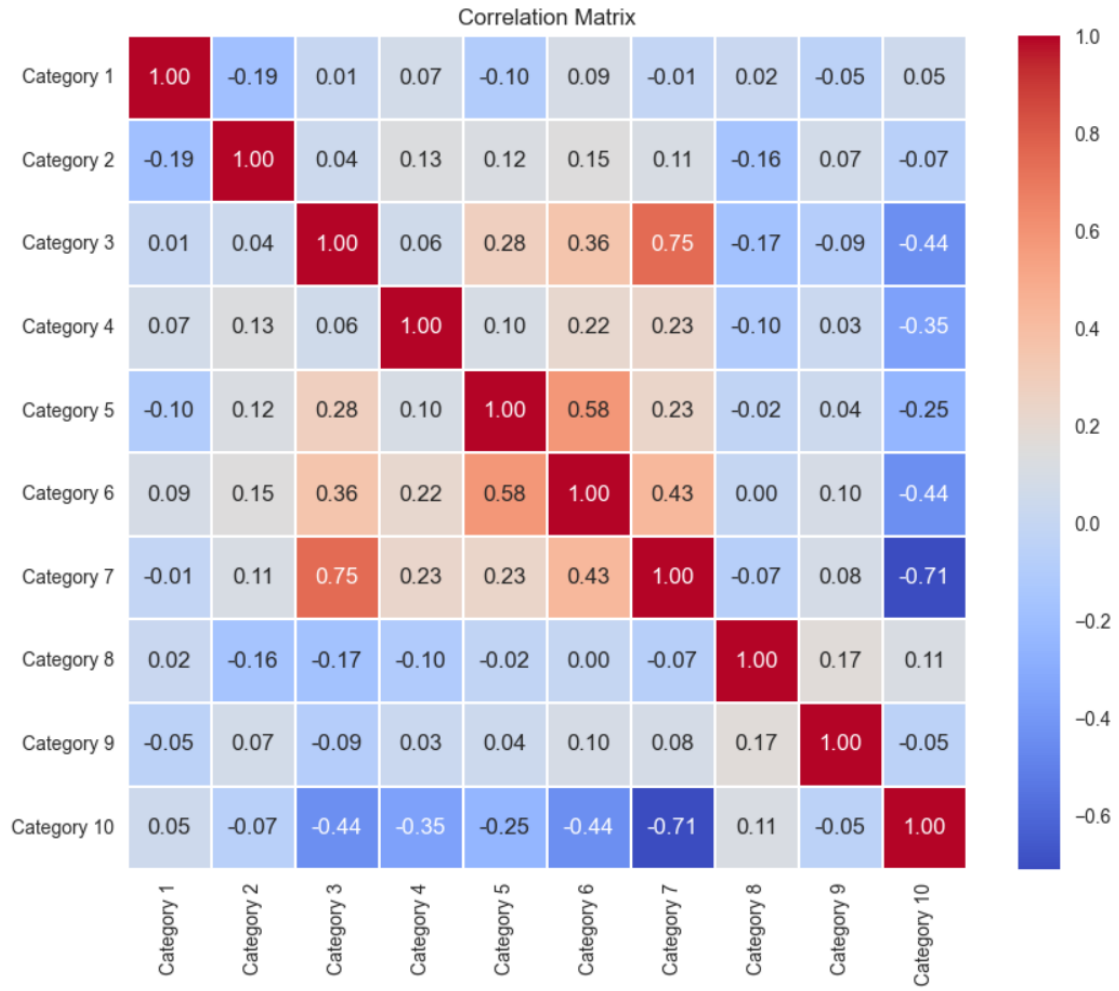
Category 7: all values are around 3.18, indicating very limited variance and no outliers.

Category 8: median is slightly below 3, and there are outliers on both ends.

Category 9: median is near 1.5 with outliers present on the right.

Category 10: median is close to 3. There are outliers on the left.

Correlation:



Category 1: does not show significant correlation with other features as the correlations are close to zero.

Category 2: does not show significant correlation with other features as the correlations are close to zero.

Category 3: shows a strong positive correlation with Category 7.

Category 4: does not show significant correlation with other features as the correlations are close to zero.

Category 5: does not show significant correlation with other features as the correlations are close to zero.

Category 6: does not show significant correlation with other features as the correlations are close to zero.

Category 7: shows a strong negative correlation with Category 10.

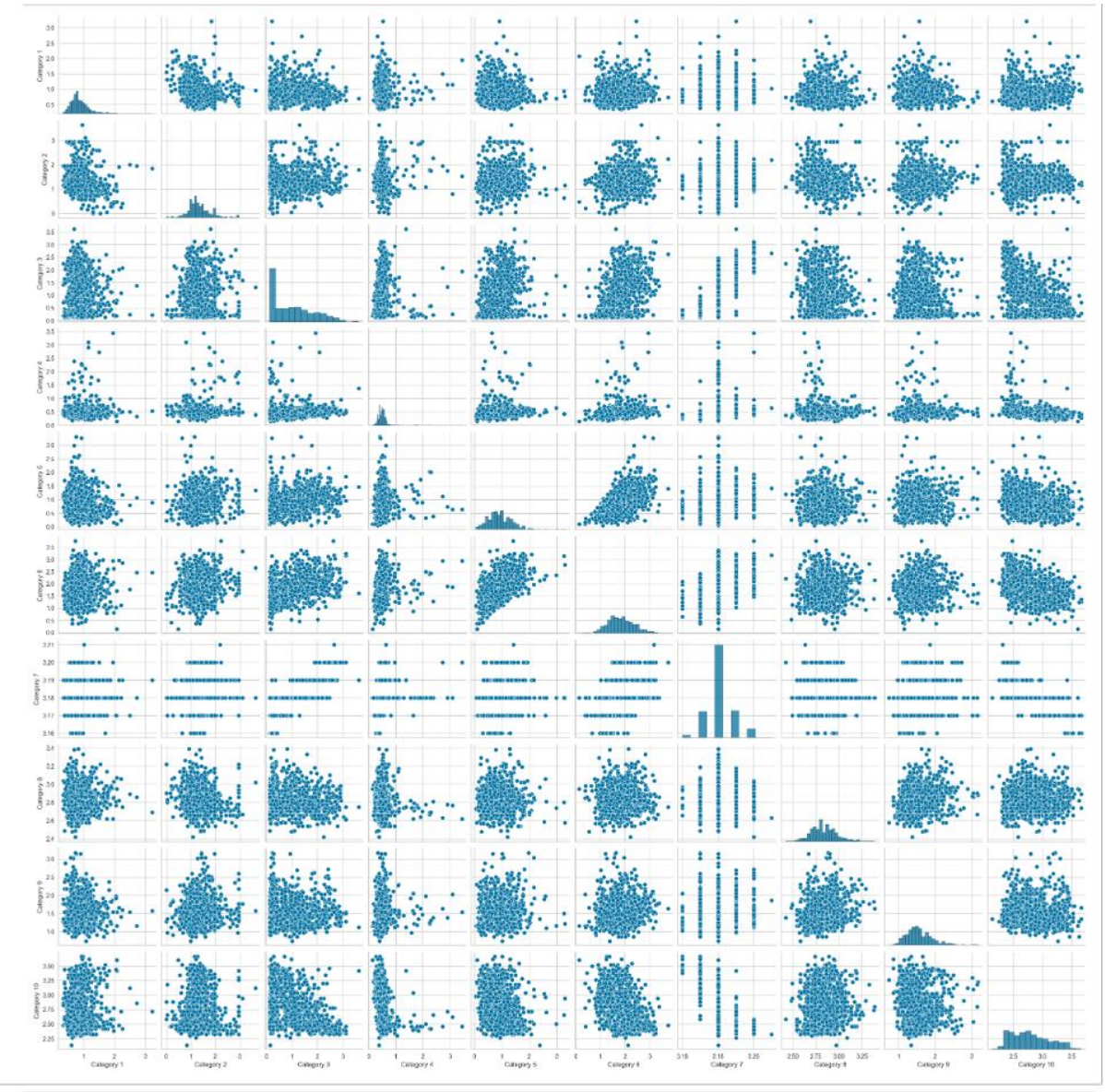
Category 8: does not show significant correlation with other features as the correlations are close to zero.

Category 9: does not show significant correlation with other features as the correlations are close to zero.

Category 10: does not show significant correlation with other features as the correlations are close to zero.

Strong Positive correlation pair: ('Category 3', 'Category 7')
Strong Positive correlation value: 0.7506509353574906
Strong Negative correlation pair: ('Category 7', 'Category 10')
Strong Negative correlation value: -0.7107309440160328

Pairplot:



Category 1 vs Category 2: has a strong positive correlation between these two features.

Category 2 vs Category 3: also has a positive correlation.

Category 3 vs Category 1: has a negative correlation.

Category 4: has a positive correlation with other features, but relationships seem to be weak.

Category 5 and Category 6: have no correlations with other features.

Category 10: does not have a strong correlation with any of the other features.

UMAP:

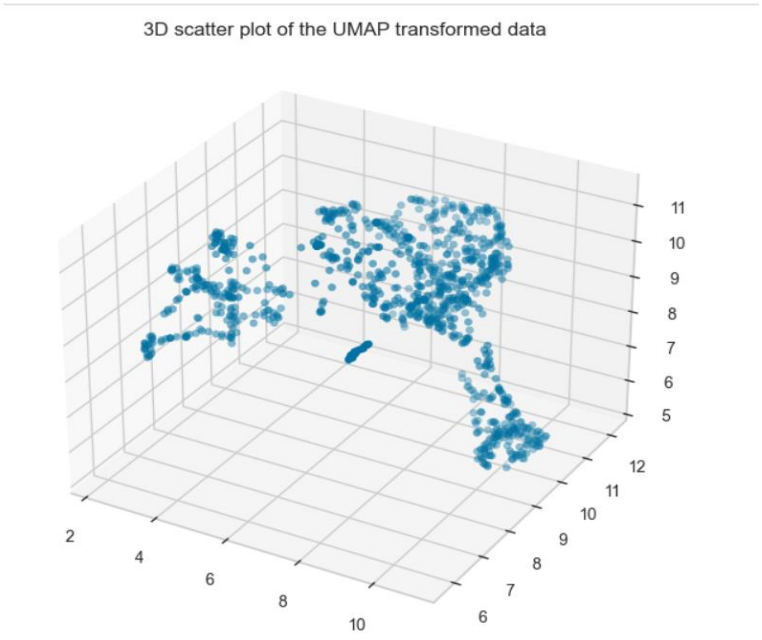
UMAP Uniform Manifold Approximation and Projection. It is a dimensionality reduction technique similar to t-SNEbut is more scalable and can be used in feature engineering and data visualization.

Feature Engineering: The transformed coordinates from UMAP can be used as features for machine learning models.

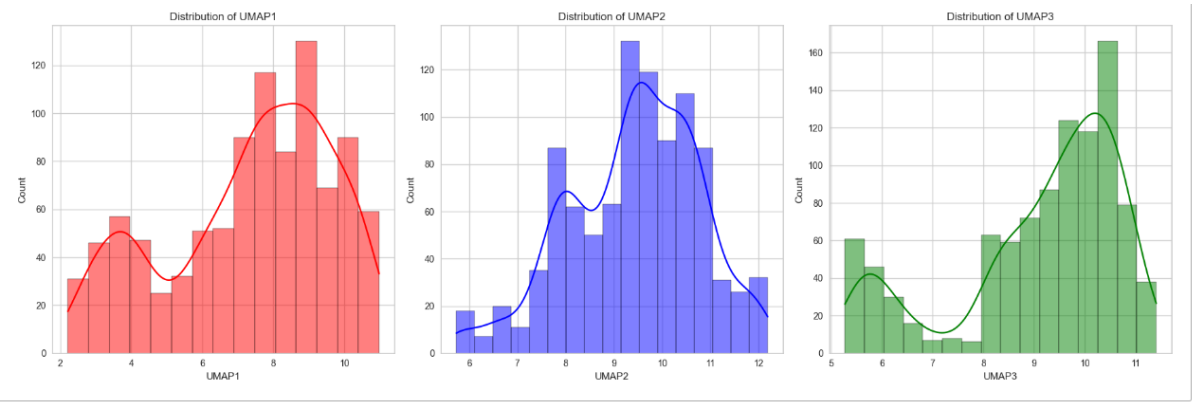
Visualization: If datasets has many features, it is challenging to visualize the data in a meaningful way. UMAP can reduce these high dimensions into 2 or 3, which can be plotted and understood visually.

Noise Reduction: By reducing dimensionality, UMAP can filter out noise and help in highlighting the underlying patterns in the data.

Improved Machine Learning Performance: High-dimensional data can lead to overfitting. Reducing the dimensionality can sometimes help in building more robust models.



The ***3D UMAP*** visualization represents the data for patterns to be visually identified



UMAP1: The distribution appears to be bimodal with two prominent peaks.

UMAP2: The distribution is somewhat uniform with a slight peak around the middle.

UMAP3: The distribution has a clear peak around the middle with tails on both sides.

Trustworthiness of the UMAP embedding: 0.9559

Score is close to 1, indicating that the low-dimensional representation UMAP features preserves much of the structure of the original columns.

	Category 1	Category 2	Category 3	Category 4	Category 5	Category 6	Category 7	Category 8	Category 9	Category 10
UMAP1	-0.025589	-0.129303	-0.801759	-0.110304	-0.414352	-0.564118	-0.843666	0.114128	0.025496	0.682479
UMAP2	0.156336	-0.343743	-0.358625	-0.490076	-0.231702	-0.326923	-0.533970	-0.085126	-0.260666	0.477892
UMAP3	-0.177594	0.287440	0.268050	0.256087	0.270449	0.382501	0.487117	-0.106977	0.045740	-0.535482

Correlation Analysis between ***UMAP*** Columns and ***Original*** Features

Category 3, Category 5, Category 6, and Category 7 have strong correlations with the UMAP components mostly with UMAP1 indicate these categories are important variance of the data.

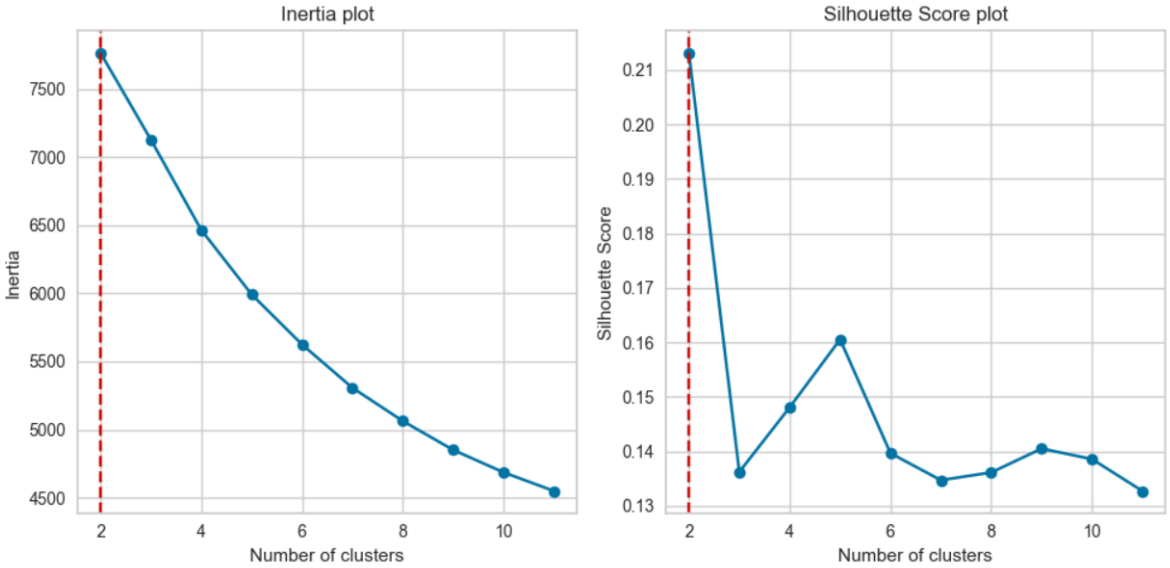
Category 10 also have strong correlations with all three UMAP components indicate importance in the dataset.

The other categories have milder correlations, but they still contribute to the UMAP components to varying degrees.

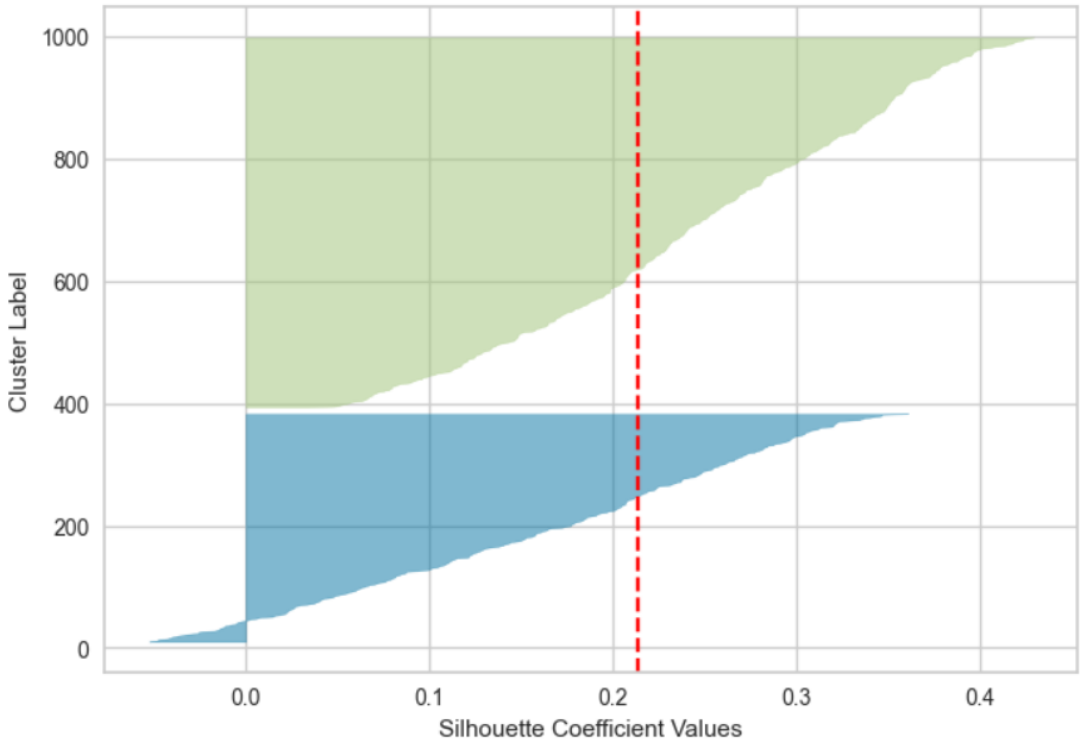
Correlations insights helped to understand original features are most closely associated with each UMAP component.

Prove UMAP feature space improve the grouping of travellers compared to original dataset:

K-means with Original Features:

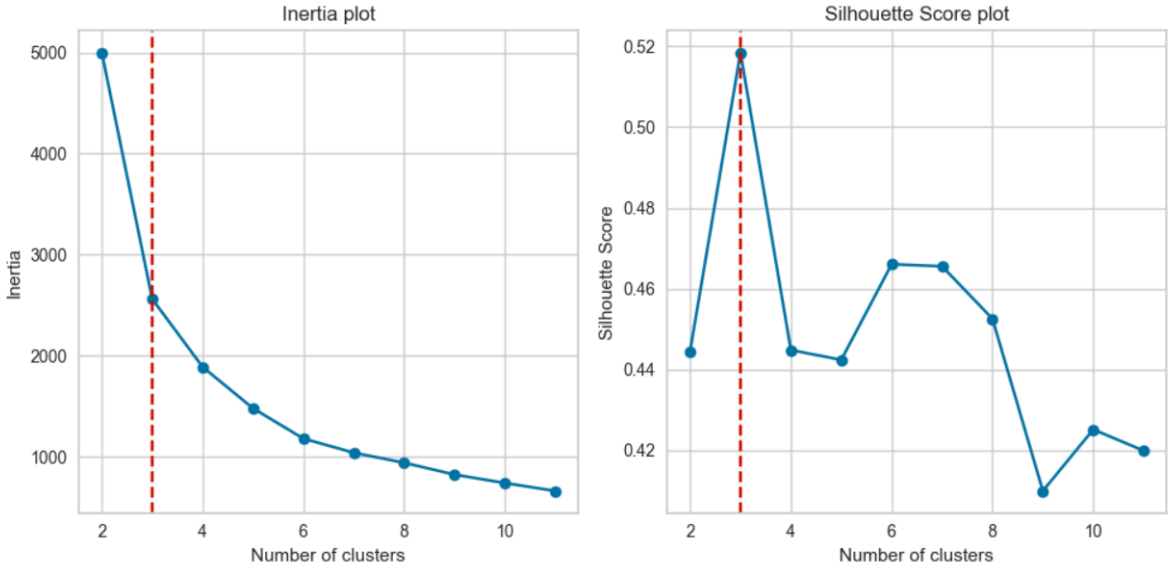


Clusters:

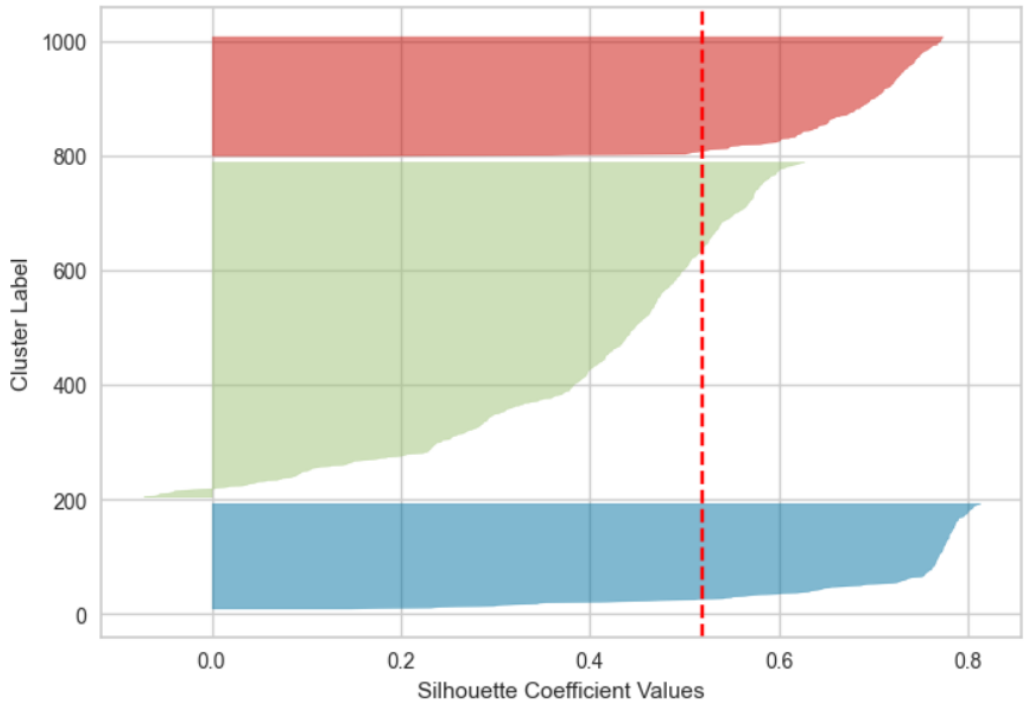


Number of Clusters: 2
Silhouette Score: 0.2132
Davies-Bouldin Index: 1.7957

K-means with UMAP:

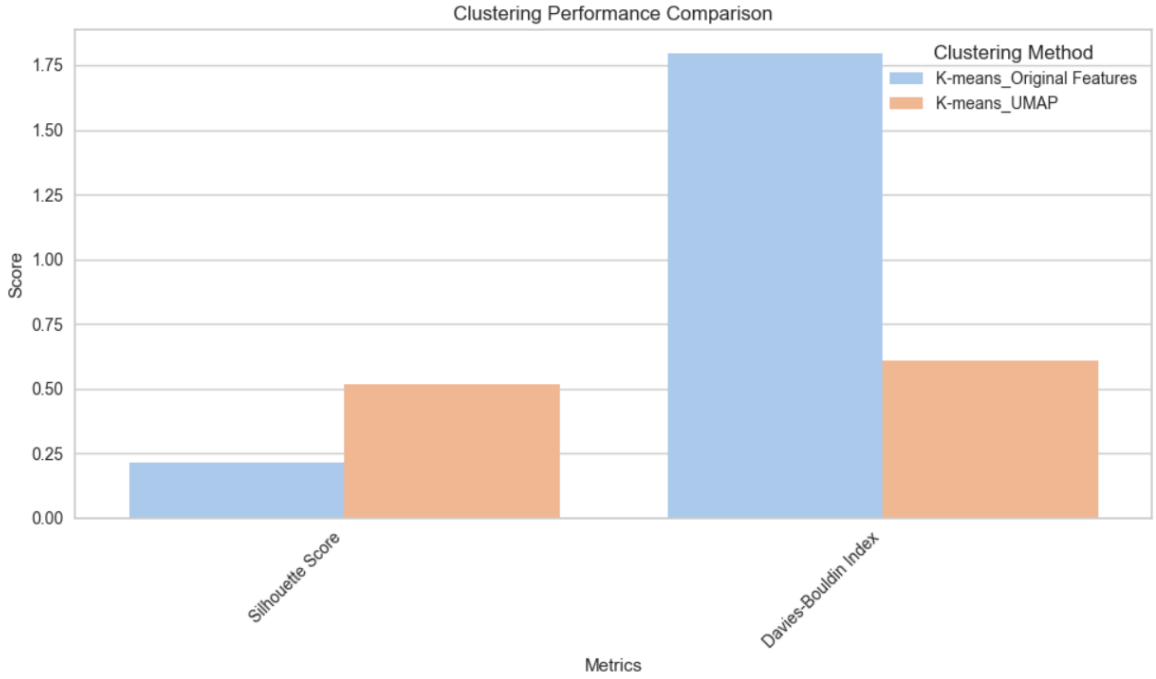


Clusters



Number of Clusters: 3
Silhouette Score: 0.5184
Davies-Bouldin Index: 0.6087

Conclusion:



Silhouette Score:

Original features with 2 clusters, the Silhouette Score is 0.2132 indicates moderate level of cluster separation.

UMAP-transformed data with 3 clusters, the Silhouette Score is significantly higher at 0.5184 indicates that the clusters in the UMAP data are much more better separated than original data.

Davies-Bouldin Index:

Original features with 2 clusters, the DBI is 1.7957 indicates clusters have a moderate level of similarity.

UMAP-transformed data with 3 clusters, the DBI is much lower at 0.6087 indicates clusters are more distinct and less similar to each other.

UMAP-transformed data provides better clustering results compared to the original features, by both Silhouette Score and Davies-Bouldin Index. Higher Silhouette Score and lower DBI for UMAP data indicates UMAP Features are effectively capturing the structure of the data & have more distinct 3 clusters.

DBSCAN Clustering for Picture Data:

Let's consider the data shown in the Figure 1.

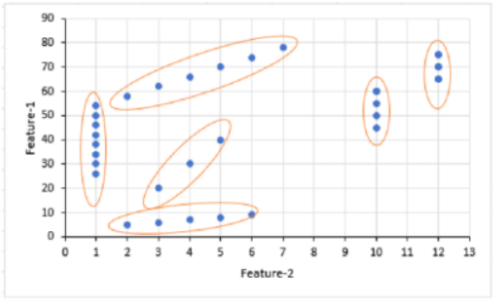
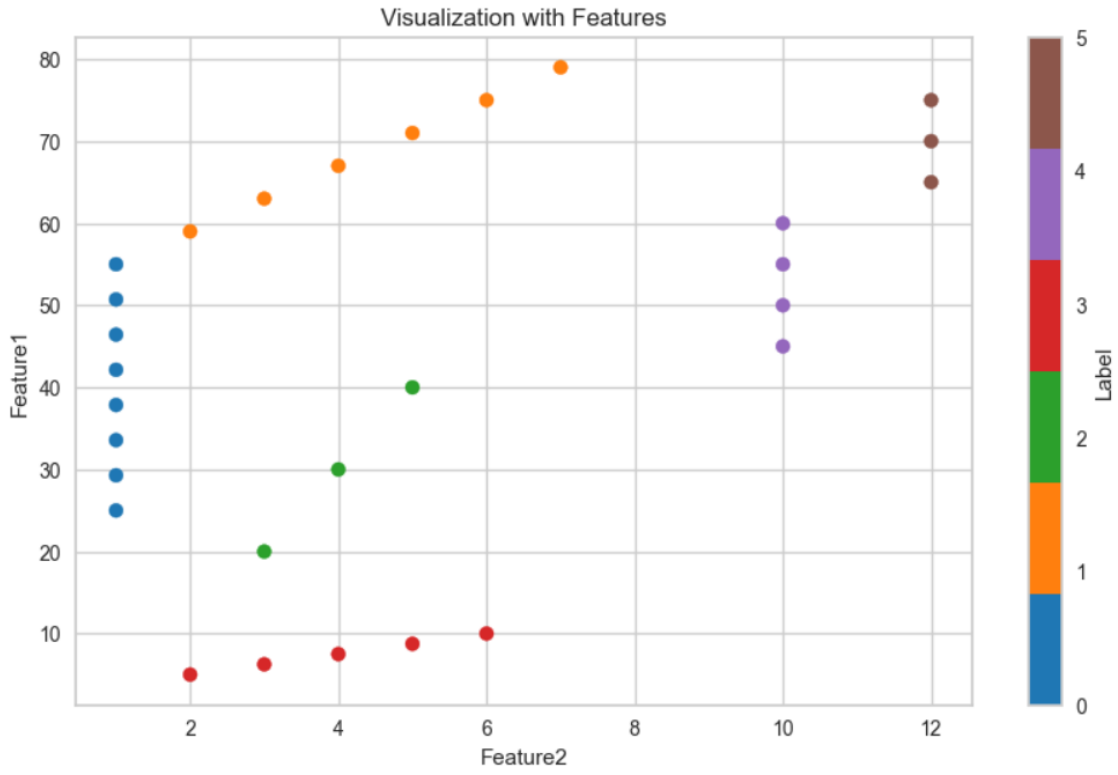


Figure 1: Scatter plot with expected clusters (elliptical shapes)

As the data is given in the image I have tried to read the data using CV2 by filtering the data points in blue but after multiple failures. Attempted to creating using numpy by reading hte values manually from the chart provided.

Replicated the picture data using the numpy library & illustrated in the below chart with six clusters.



Data from Image:

Tried to get data points from an image using OpenCV. This didn't work well, so we used numpy to create the data based on points in the image.

Creating Features:

Created Feature1 and Feature2, from sequences of numbers. These numbers were chosen to match the clusters in the image.

Labels:

Labeled the data points with numbers from 0 to 5 to match the 6 clusters in image.

Plotted the data created using numpy with the colors in the plot to replicate the images & ensured the data points mataches the image provided.

Scaling Data:

Used StandardScaler to scale the numpy data.

Six Clusters:

Used DBSCAN clustering to group our data into 6 clusters, like in the original image. Tested different ireations using below:

Different numbers of nearest neighbors.
Multiple Eps values to determine how close points should be to form a cluster.
Different min_samples values to determine the minimum number of data points to form a dense region.

Results for 6 Clusters:

All this combination using DBSCAN could not find a set of parameters that could give 6 clusters indicates DBSCAN might not be the best choice for this specific data to represent the cluster in the original image.

Hence tried the best next option to check whether we could achive 5 clusters

5 Clusters:
Attempted with the same parameter iterations to recreate the clusters to look like original image

Though initial goal is to create 6 clusters as in image we could not achieve it but managed to achieve a close representation with 5 clusters using DBSCAN. Used the same parameter combinations as before but adjusted to represent 5 clusters.

Evaluation Metrics:

ARI (Adjusted Rand Index): 0.6615

A measure of the similarity between two sets of clusters. A value of 1 indicates perfect agreement, while a value of 0 indicates that clusterings are independent of one another.

Silhouette Score: 0.3031

Measures the quality of clusters, with values ranging between -1 and 1. A higher value indicates better-defined clusters.

Purity: 7.6667

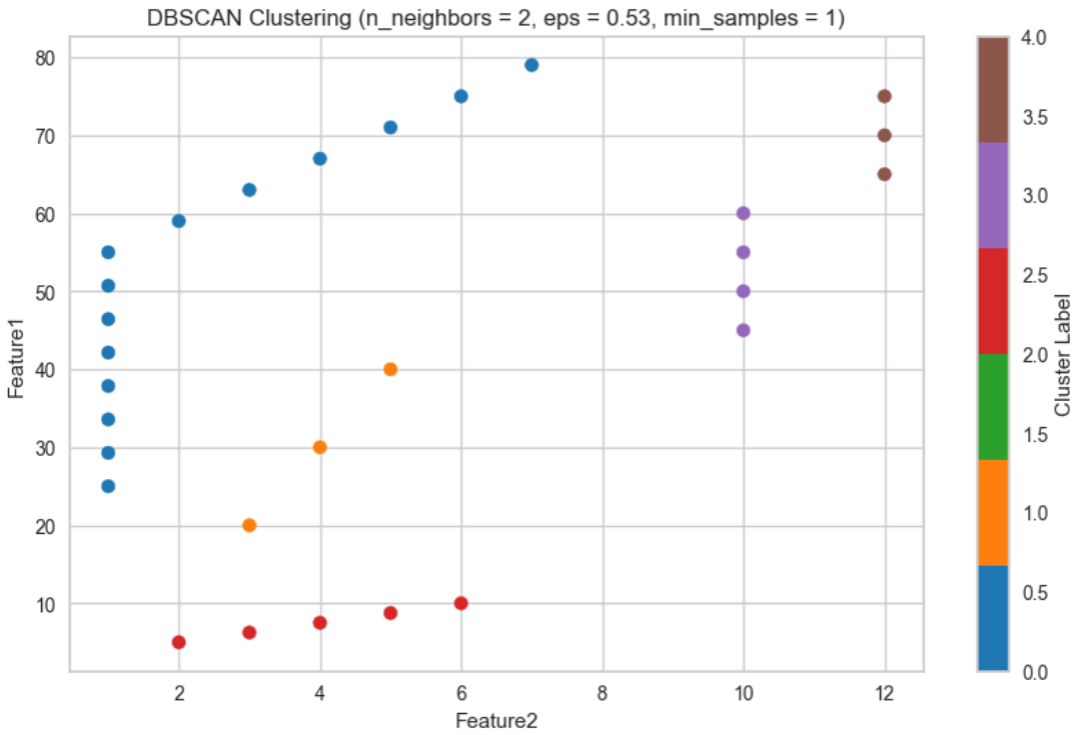
The values indicate the number of samples that are clustered correctly in each cluster. Higher values are better, but the metric doesn't account for the number of clusters and can be misleading when the number of clusters increases.

Davies-Bouldin Index: 0.7578

A value closer to 0 indicates better partitioning. It's the average similarity ratio of each cluster with its most similar cluster.

Mutual Information: 1.3973

Measures the agreement of the original labels and the new cluster assignments, ignoring permutations and with higher values indicating more agreement.



Limitations of DBSCAN:

Density Dependence:

DBSCAN relies on a density-based notion of clusters and aims to discover clusters of varying shapes in areas of different densities. This can pose problems when the dataset has clusters with different densities. If two distinct clusters have similar densities and are close to each other, DBSCAN may merge them.

Context: Within our dataset, we have some groupings are closer together or have overlapping densities, DBSCAN may struggle to differentiate them, leading to fewer clusters.

Parameter Sensitivity:

eps and min_samples parameters play a important role in the performance of DBSCAN. Finding the optimal values for these parameters is not always straightforward and can differ significantly depending on the dataset.

Context: It is possible that our dataset, the inherent cluster structures are sensitive to the choice of eps and min_samples. No combination of these parameters within the explored range could capture the desired 6-cluster structure.

Noisy Points:

DBSCAN classifies certain points as noise. In datasets where clusters are in close proximity, DBSCAN may classify points on the edges of these clusters as noise, leading to a reduction in cluster count.

Context: If some of our clusters are not dense enough or are close to other clusters, some points might be classified as noise.

Difficulty with Clusters of Different Densities:

While DBSCAN can identify clusters of varying shapes, it might struggle when there is a mix of dense and sparse clusters.

Context: Wiht out dataset we have a mix of dense and sparse clusters, a single eps value might either merge multiple sparse clusters or split a dense cluster, preventing the formation of 6 distinct clusters.

Assumption of Uniform Cluster Sizes:

If there are smaller natural clusters in the dataset, DBSCAN might treat them as noise or merge them with larger clusters.

Context: Clusters in our data have significantly different sizes, the smaller ones might not be detected by DBSCAN.

Conclusion:

The data was successfully clustered into 5 groups using DBSCAN which is close to original image 6 clusters indicates that DBSCAN cannot recreate the exact original clusters but provided a close representation of orginal image.

while DBSCAN is a powerful clustering algorithm, especially for data with clusters of varying shapes, it has its limitations. The nature of our dataset, particularly in terms of cluster density, proximity, and size, might have made it challenging for DBSCAN to identify the desired 6 clusters.

State of Art:

Spectral Clustering (originated in the 1970s): Became more popular and refined in the 2000s and 2010s.

Spectral Clustering is a variant of the clustering algorithm that uses the connectivity between the data points to form the clustering. It uses eigenvalues and eigenvectors of the data matrix to forecast the data into lower dimensions space to cluster the data points. It is based on the idea of a

graph representation of data where the data point are represented as nodes and the similarity between the data points are represented by an edge.

Ensemble Clustering (around 2009-2012): Combining multiple clustering results for improved accuracy and robustness.

Ensemble clustering, also called consensus clustering, has been attracting much attention in recent years, aiming to combine multiple base clustering algorithms into a better and more consensus clustering. Due to its good performance, ensemble clustering plays a vital role in many research areas, such as community detection and bioinformatics

OPTICS Clustering: *Ordering Points To Identify the Clustering Structure (Introduced in 1999).*

OPTICS is a density-based clustering algorithm that extends the capabilities of DBSCAN. Unlike DBSCAN, which requires a single density threshold, OPTICS produces an ordered list of points based on their reachability and density, facilitating the extraction of multiple clusters at varying density levels. The algorithm is particularly adept at discovering clusters of differing shapes and densities within a single dataset.

Ensemble Clustering:

Ensemble Clustering (around 2009-2012): Combining multiple clustering results for improved accuracy and robustness.

Ensemble clustering, also called consensus clustering, has been attracting much attention in recent years, aiming to combine multiple base clustering algorithms into a better and more consensus clustering. Due to its good performance, ensemble clustering plays a vital role in many research areas, such as community detection and bioinformatics

Base Clusterings K-means:

K-means clustering is performed multiple times using varying numbers of clusters from 3 to 6 clusters for each iterations co-association matrix is updated based on which data points are clustered together.

Applying Hierarchical Clustering:

Hierarchical clustering is applied on the co-association matrix using different linkage methods: single, complete, average, and ward. Dendrogram produced by hierarchical clustering is then cut to achieve exactly six clusters. Only stored the result when we get target six clusters.

Also, Visualized all possible 6 clusters above.

Ensemble methods is used to produce more stable clustering by averaging across multiple base clusterings & helped to reduce the variance and improve the final clusters.

ARI (Adjusted Rand Index): Reflects the similarity between the true labels and the clustering labels. Giger the better, The values range from 0.436 to 0.595 for top four configurations indicates a moderate similarity between the true labels and clustered labels for these configurations.

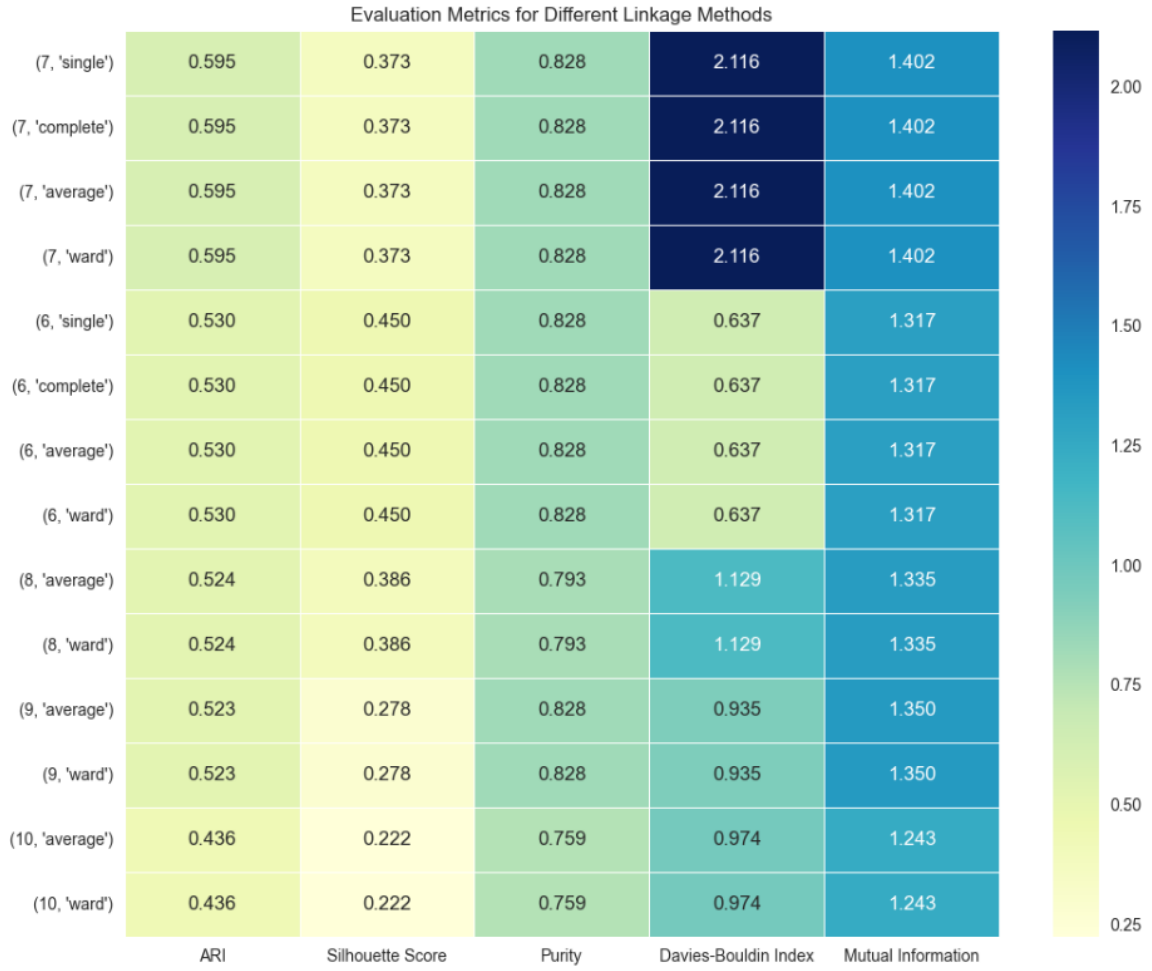
Silhouette Score: Represents the quality of clusters. higer the better & top score is 0.449 indicate clusters are reasonably separated.

Purity: Measures the proportion of the dominant class in each cluster. Higer the better & high purity for various combinations is 0.828 & indicates proportion of the data points in each cluster belong to the dominant class.

Davies-Bouldin Index: Evaluates the cluster's validity based on the average similarity ratio of each cluster with its most similar cluster. A lower value indicates more distinct and well-defined clusters. lower the better. The lowest score is approximately 0.637 indicates well-separated clusters.

Mutual Information: Determines the shared information amount between the true labels and the clustering labels. higer the better. The scores range from 1.243 to 1.402 & top configurations having the highest mutual information indicates strong amount of shared information between the true labels and the clustering labels.

Top four configurations with highest ARI use 7 base clusters and vary only in their linkage methods. Ensemble clustering configurations with 7 base clusters (7, single), (7, complete), (7, average), (7, ward) has performed better across all metrics followed by 6 base clusters

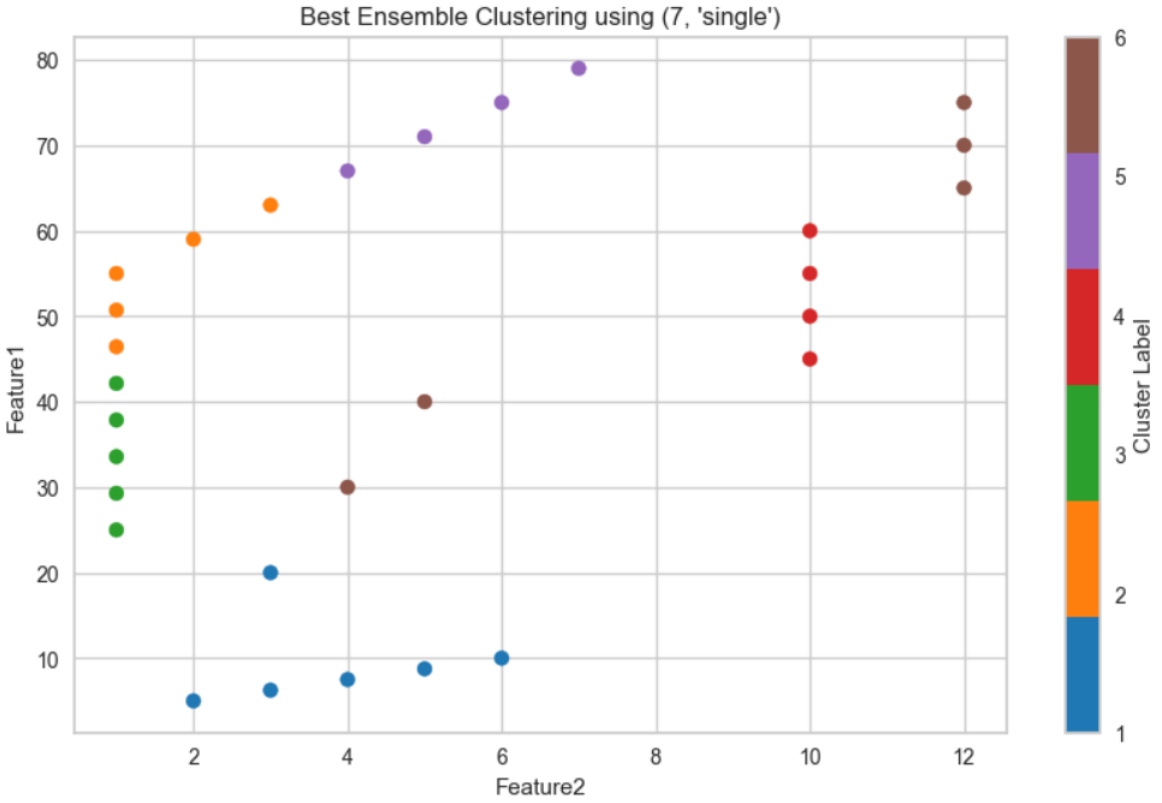


Final selection of the model which is close to Original image

Best Method to nearly match the original image are: (7, 'single')

ARI (Adjusted Rand Index): 0.5952
Silhouette Score: 0.3729
Purity: 0.8276
Davies-Bouldin Index: 2.1156
Mutual Information: 1.4017

Conclusion:



Ensemble clustering using single linkage method performed well to match the original cluster which is represented the chart above.

Spectral Clustering:

Spectral Clustering (originated in the 1970s): Became more popular and refined in the 2000s and 2010s.

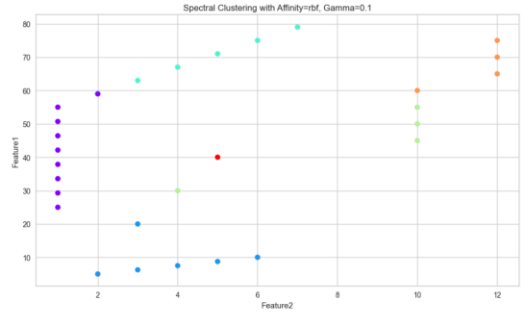
Spectral Clustering is a variant of the clustering algorithm that uses the connectivity between the data points to form the clustering. It uses eigenvalues and eigenvectors of the data matrix to forecast the data into lower dimensions space to cluster the data points. It is based on the idea of a graph representation of data where the data point are represented as nodes and the similarity between the data points are represented by an edge.

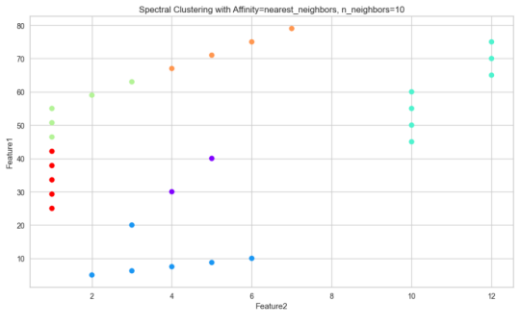
Clustering:

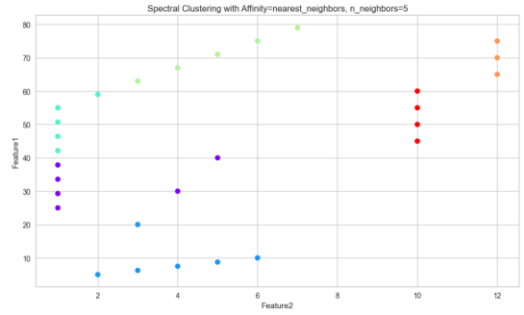
Iterates over different combinations of gamma values and n_neighbors values for Spectral Clustering

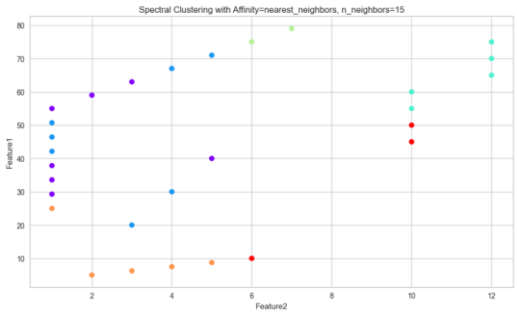
Affinity as 'rbf' (Radial basis function) and varying gamma.
Affinity as 'nearest_neighbors' with varying n_neighbors.
Also plotted the results when we have 6 clusters are formed.

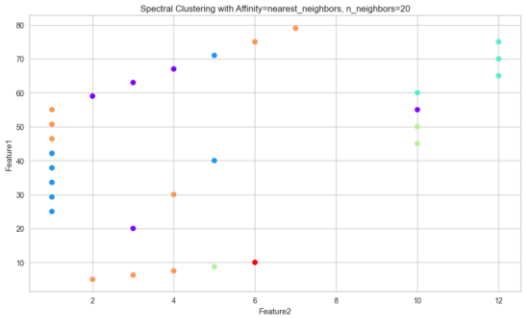
Also, displayed all possible 6 clusters:

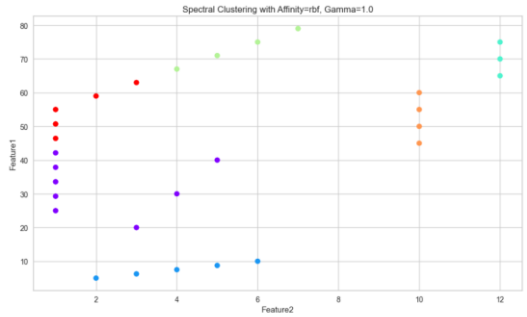

Spectral Clustering with Affinity=rbf, Gamma=0.1
Feature1
Feature2

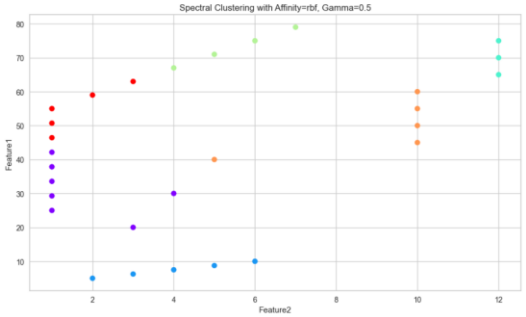

Spectral Clustering with Affinity=nearest_neighbors, n_neighbors=10
Feature1
Feature2

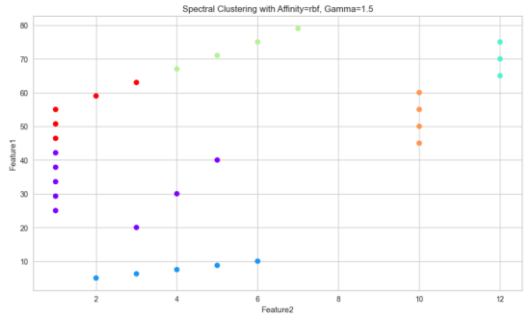

Spectral Clustering with Affinity=nearest_neighbors, n_neighbors=5
Feature1
Feature2


Spectral Clustering with Affinity=nearest_neighbors, n_neighbors=15
Feature1
Feature2

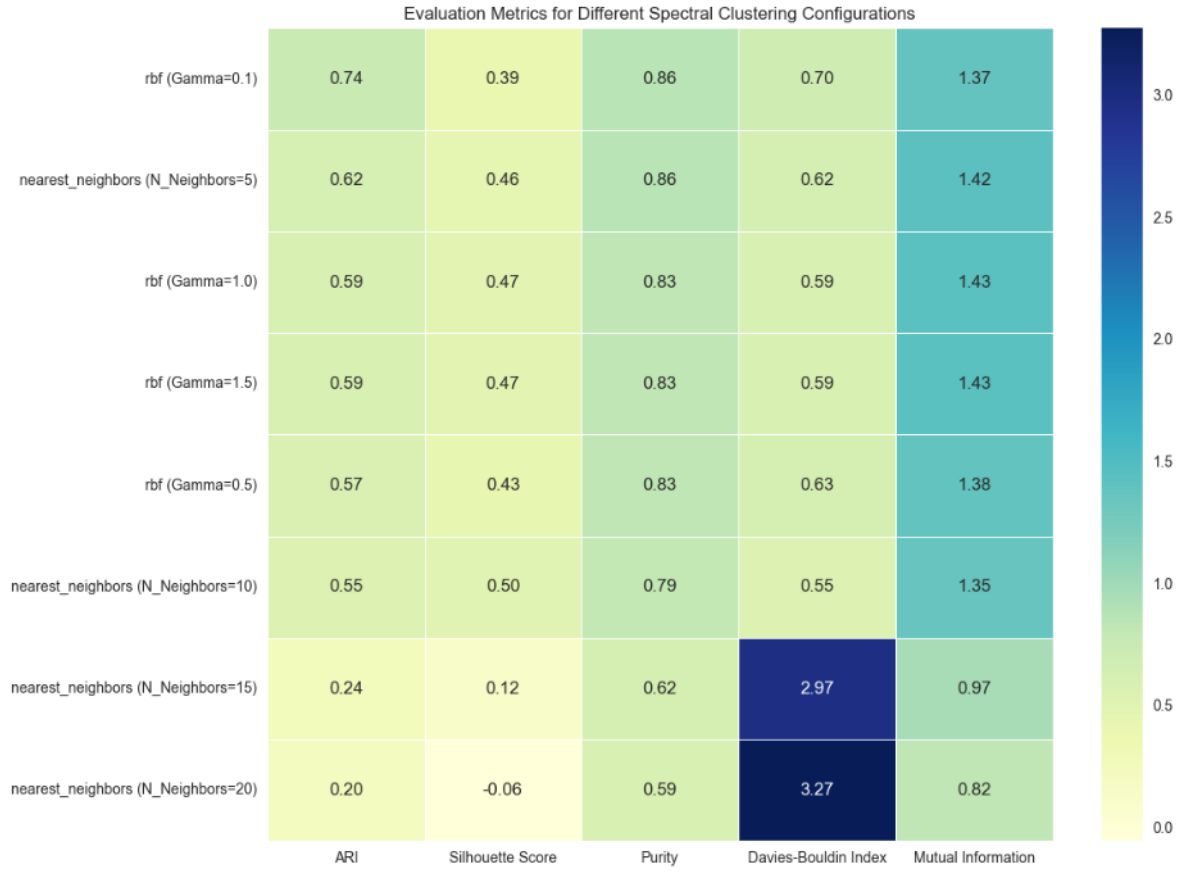

Spectral Clustering with Affinity=nearest_neighbors, n_neighbors=20
Feature1
Feature2


Spectral Clustering with Affinity=rbf, Gamma=1.0
Feature1
Feature2


Spectral Clustering with Affinity=rbf, Gamma=0.5
Feature1
Feature2


Spectral Clustering with Affinity=rbf, Gamma=1.5
Feature1
Feature2

Evaluation Metrics:



ARI (Adjusted Rand Index): indicates how close the true labels and the clustering labels are. Higher is better. Values go from 0.436 to 0.595. The top four configurations say that there's a decent match between the true labels and the cluster labels.

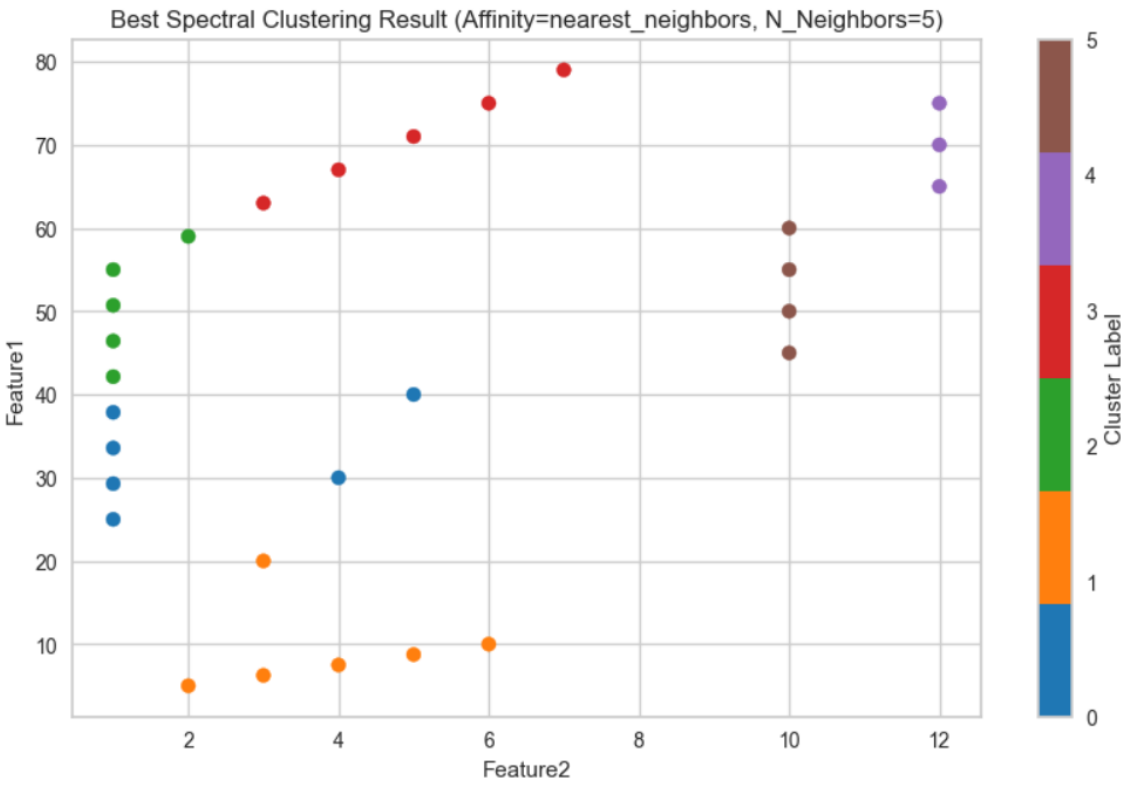
Silhouette Score: Indicates cluster quality. higher number, better cluster. The highest score is 0.449, which means the clusters are fairly apart from each other.

Purity: indicates how much of the main class is in each cluster. higher number is better. The highest purity in the data is 0.828. This says that most data points in the cluster are from the main class.

Davies-Bouldin Index: Checks how good the clusters are based on how similar clusters are to each other. Smaller number, better cluster. The smallest score is about 0.637. This says the clusters are pretty different from each other.

Mutual Information: Checks at the shared info between true labels and clustering labels. Bigger number is better. Scores are between 1.243 to 1.402. The best configurations have the most shared info between true and cluster labels.

Final selection of the model which is close to Original image



Best Method to nearly match the original image are:

Affinity nearest_neighbors
Gamma NaN
N_Neighbors 5.0
Average Score 0.93383
ARI (Adjusted Rand Index): 0.619440
Silhouette Score: 0.4639
Purity: 0.8620
Davies-Bouldin Index: 0.6233
Mutual Information: 1.4157

Conclusion:
Compared to Ensemble clustering & DBScan spectral clustering performed better to match the original clustering in the image but we could not match original cluster which is represented the chart above & almost matched the original clustering with 86% purity.

OPTICS Clustering:

OPTICS (Ordering Points To Identify the Clustering Structure) is a density-based clustering algorithm similar to DBSCAN. It automatically identifies the number of clusters based on the data's density distribution, making it suitable for datasets with varying densities.

Clustering:

Iterates over different combinations of min_samples_values values and xi values for Optics Clustering

min_samples_values specifies the minimum number of samples required to form a dense region

xi values sets the steepness criterion for identifying clusters within the reachability plot

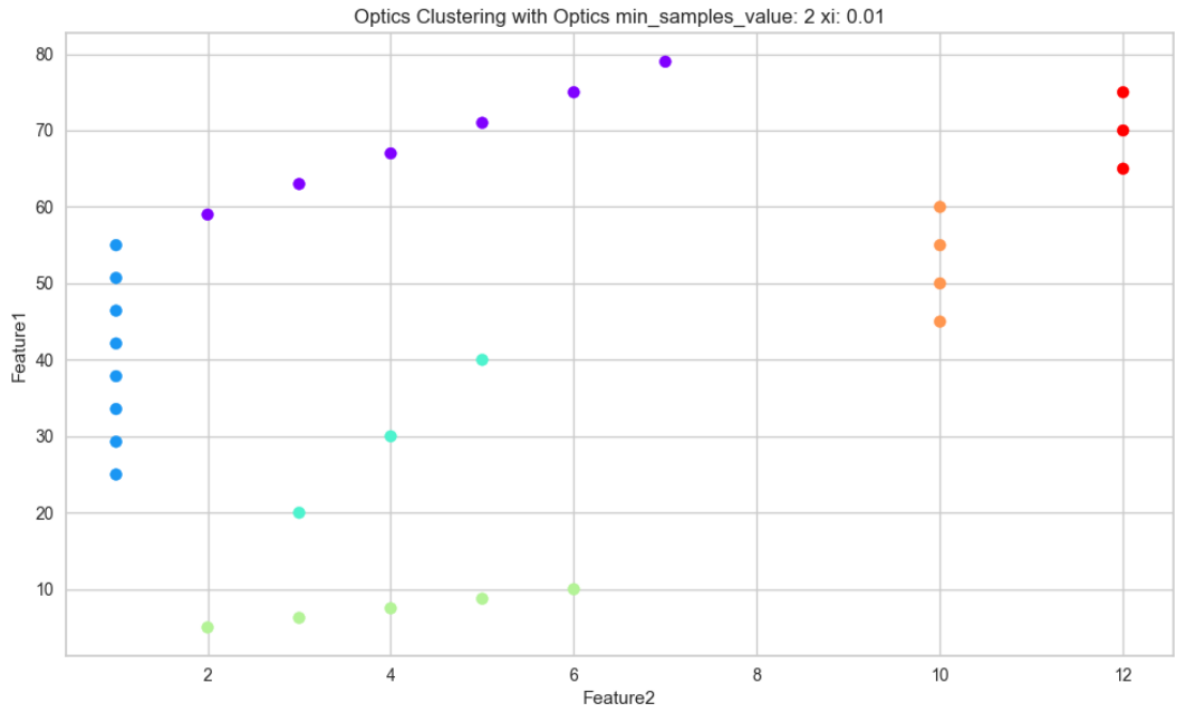
Also plotted the results when we have 6 clusters are formed & All 6 cluters charts has matched upto original clusters with noise points.

OPTICS clustering algorithm was successful in identifying six different labels, including noise, which is denoted as '-1'. It's important to note that while OPTICS could replicate the 6-cluster structure present in the original y labels, one of these clusters is consistently labeled as noise.

This is a significant aspect of density-based algorithms like OPTICS: they have the ability to identify noise or outliers in the data. In the context of this dataset, it suggests that while five clusters closely match the original groups, the sixth group is considered by OPTICS as more of a 'noise' rather than a distinct cluster.

The data points in this "noise" cluster are too sparse and do not meet the density criteria to form a valid cluster.

We have 6 clusters results for 10 instances with different parameters values but all the time we had the clusters matched to the original clusters but with noise as one cluster.



Evaluation Metrics:

ARI (Adjusted Rand Index): A perfect ARI score of 1.0 suggests that the clustering perfectly matches the true labels.

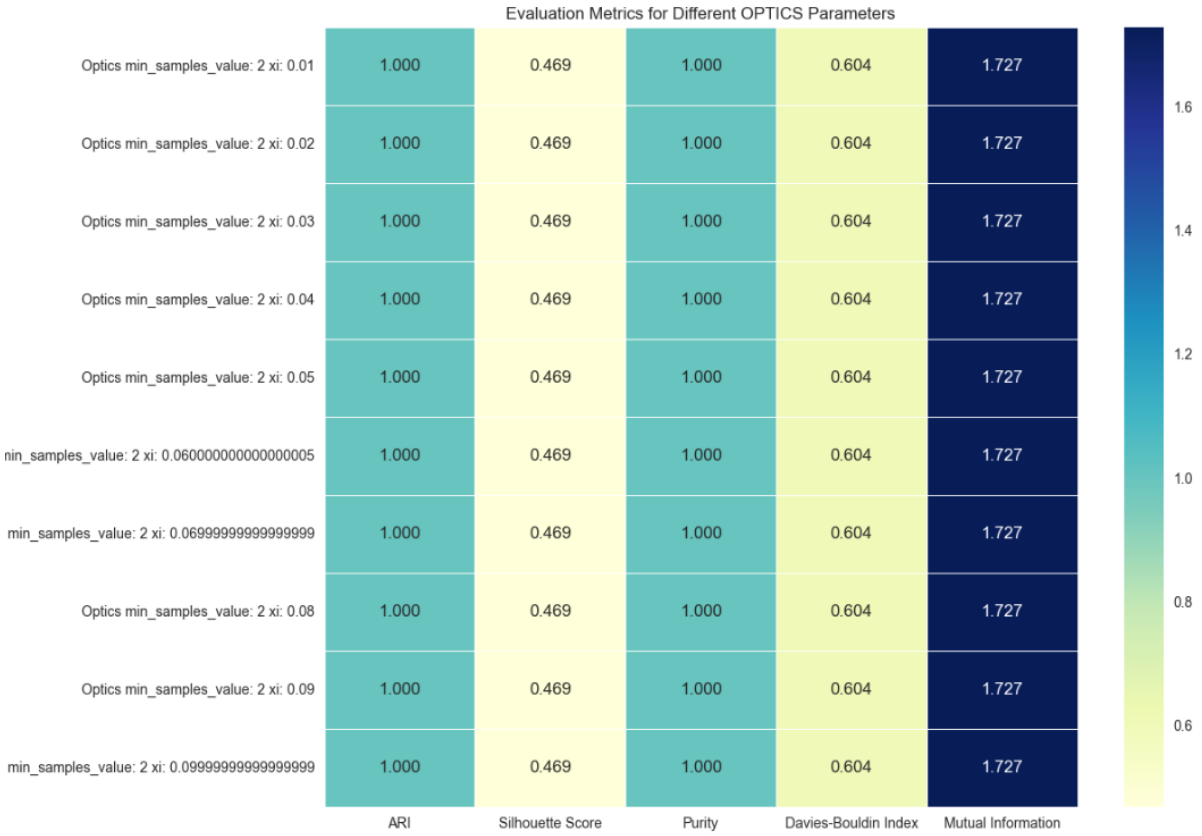
Silhouette Score: A maximum score of approximately 0.47 implies that the data points are closer to other data points in the same cluster compared to those in other clusters.

Purity: The purity score, although not fully displayed, seems to suggest high purity in the clusters, meaning most data points in each cluster belong to a single class.

Davies-Bouldin Index: The score of around 0.60 indicates that each cluster is well-separated from other clusters. Lower values are better, but this is fairly good.

Mutual Information: A high score of around 1.73 suggests a large amount of information is shared between the true labels and the clustering labels.

The high ARI and mutual information scores suggest a strong correlation between the true labels and the clustering labels, but the presence of a 'noise' cluster also points to some limitations in fully replicating the original clustering structure.



Final selection of the model:

Best Method to matched the original image are:

Optics min_samples_value: 2 xi: 0.01

ARI 1.0

Silhouette Score 0.46865

Purity 1.0

Davies-Bouldin Index 0.603897

Mutual Information 1.726947

Evaluation metrics has same values

Data Insensitivity to Parameters: The dataset inherently have very well-defined clusters and have mixed that varying parameters for OPTICS does not yield different results. The clusters are too distinct to be affected by small parameter changes.

Algorithm Limitations: data do not have varying densities for OPTICS to effectively separate clusters with different parameter values. This could be why we see one cluster always labeled as ***noise***.

Conclusion:

OPTICS outperformed Spectral and Ensemble clustering methods in replicating the original 6-cluster structure with a perfect ARI score of 1.0. The model not only found the six clusters but also showed high purity and mutual information scores, indicating that the clusters closely match the original labels. The consistency across different hyperparameters also suggests that the clustering is robust and less sensitive to the choice of hyperparameters for this particular dataset.

State-of-the-Art Conclusion and Comparative Analysis

Explored three different clustering methods: Ensemble Clustering, Spectral Clustering, and OPTICS. The aim was to achieve a clustering that closely matches the original image clusters.

Table summarizing the evaluation metrics for the best parameters for each clustering method:

Clustering Method	Best Parameters	ARI	Silhouette Score	Purity	Davies-Bouldin Index	Mutual Information
Ensemble	(7, 'single')	0.5952	0.3729	0.8276	2.1156	1.4017
Spectral	Affinity: nearest_neighbors, N_Neighbors: 5.0, Gamma: NaN	0.6194	0.4639	0.8620	0.6233	1.4157
OPTICS	min_samples: 2, xi: 0.01	1.0	0.46865	1.0	0.6039	1.7269

Key Takeaways:

Optimal Performance: OPTICS outperformed the other two methods in all evaluation metrics. It achieved a perfect ARI and purity score of 1.0, which suggests that it was able to almost recreate the original clusters perfectly.

Handling Noise: Despite its excellent performance, OPTICS did identify one cluster as noise (-1) & it's the limitation of the algorithm for this dataset.

Silhouette Score: Spectral clustering and OPTICS have similar Silhouette Scores, indicating that the clusters are well apart from each other in both cases.

Davies-Bouldin Index: OPTICS has the lowest Davies-Bouldin Index, suggesting that its clusters are the most distinct compared to the other methods.

Mutual Information: OPTICS has the highest Mutual Information, indicating the most amount of shared information between the true labels and the cluster labels.

Purity: Spectral and OPTICS both have high purity, but OPTICS reaches a perfect score, indicating that it could recreate the original clusters most accurately.

Final Conclusion:

Based on the evaluation metrics, OPTICS seems to be best algorithm for clustering for this dataset. It not only achieved the highest scores in all evaluation metrics but also matched in recreating the original clusters. The only drawback was the presence of a noise cluster, which is due to characteristic of the dataset

Yeast Dataset:

Background

The recently started human and other genome projects are likely to change the situation of molecular biology. Comprehensive analyses of whole genomic sequences will enable us to understand the general mechanisms of how protein and nucleic acid functions are encoded in the sequence data.

Dataset filename: yeast2vs4.csv

Dataset description: There are 8 features and one target in the dataset. All the features are in a numerical format, and the target is in text format. For further information about the attributes,

Attribute Information:

1. mcg: McGeoch's method for signal sequence recognition.
2. gvh: von Heijne's method for signal sequence recognition.
3. alm: Score of the ALOM membrane spanning region prediction program.
4. mit: Score of discriminant analysis of the amino acid content of the N-terminal region (20 residues long) of mitochondrial and non-mitochondrial proteins.
5. erl: Presence of "HDEL" substring (thought to act as a signal for retention in the endoplasmic reticulum lumen). Binary attribute.
6. pox: Peroxisomal targeting signal in the C-terminus.
7. vac: Score of discriminant analysis of the amino acid content of vacuolar and extracellular proteins.
8. nuc: Score of discriminant analysis of nuclear localization signals of nuclear and non-nuclear proteins.
9. class: Presence or absence of protein {positive, negative}.

Data Understanding:

The dataset contains 514 rows and 9 columns with 8 columns represent features and last column class is the target variable. All the features are numerical except of target variable is categorical.

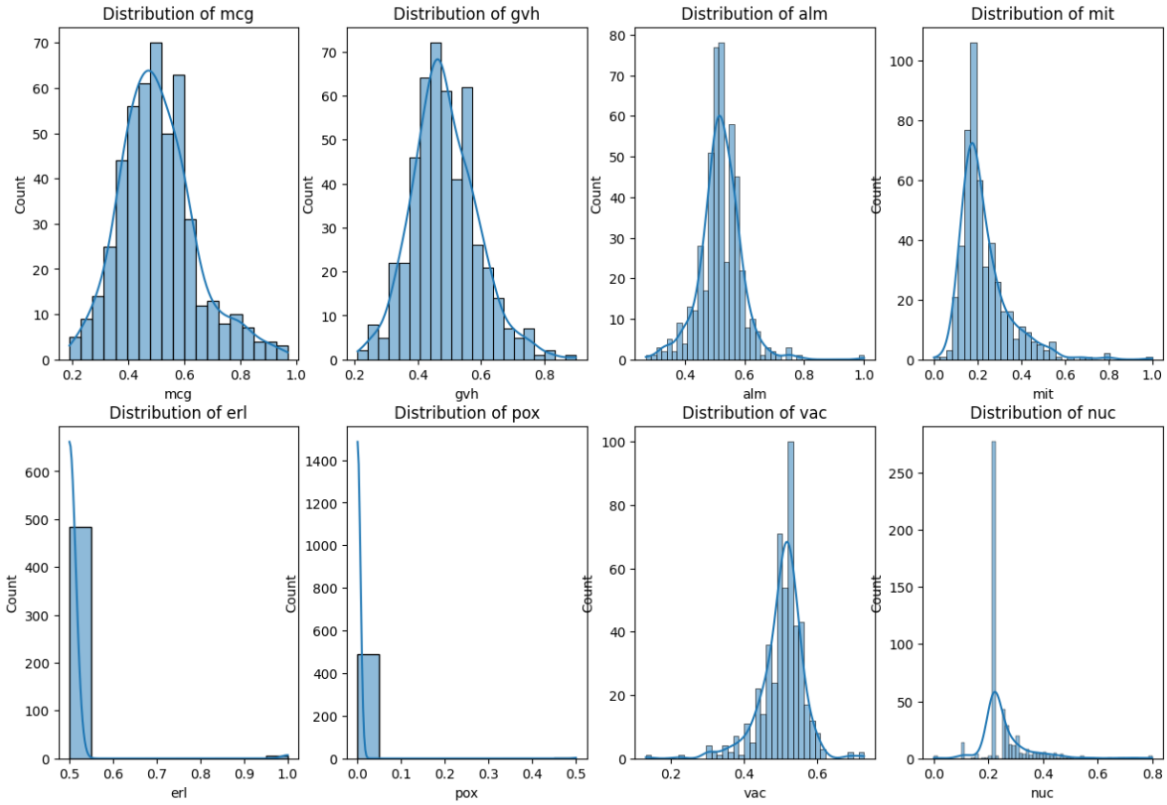
25 Duplicates entries were found in the dataset & They are removed & there are no missing values.

All numerical features range from 0 to 1, except for the 'erl' column, which has a minimum value of 0.5.

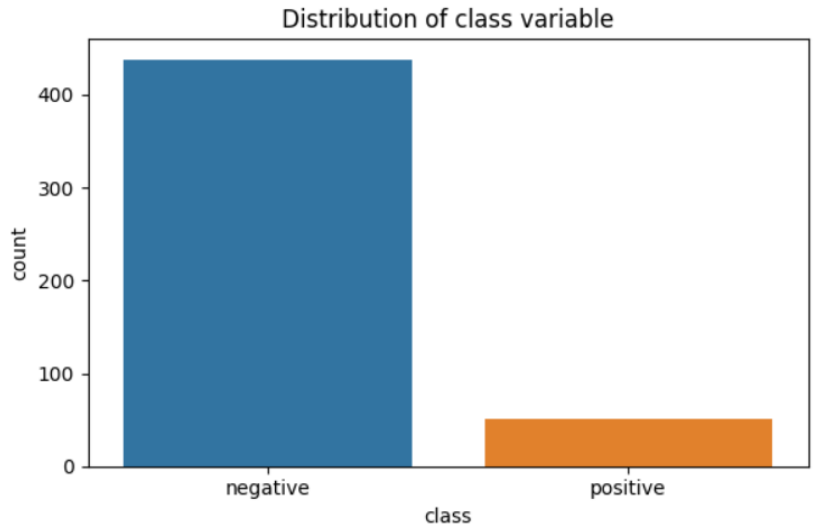
Columns and Datatypes of each column
mcg float64
gvh float64
alm float64
mit float64
erl float64
pox float64
vac float64
nuc float64
class object
Dataset after removing duplicates are 489 rows and 9 columns & Size of dataset is 4401.

Univatite Analysis:
Descibe:
mcg: has a mean of 0.51 & standard deviation of 0.14. Values range from 0.19 to 0.97 & slightly right skewed
gvh: has a mean of 0.49 & standard deviation of 0.10. Values range from 0.21 to 0.90 & normally distributed.
alm: has a mean of 0.52 & standard deviation of 0.07. Values range from 0.27 to 1.0 & normally distributed.
mit: has a mean of 0.23 & standard deviation of 0.11. Values range from 0 to 1 & right skewed.
erl: most of values 0.5 hence mean, min, max, and all percentiles are 0.5 & has little variance.
pox: most of values 0s hence mean, min, max, and all percentiles being 0 & has little variance.
vac: has a mean of 0.50 &standard deviation of 0.06. Values range from 0.13 to 0.73 & normally distributed.
nuc: has a mean of 0.26 & standard deviation of 0.09. Values range from 0 to 0.8 & right skewed.
class: is a target categorical variable which is binary 0 and 1. Class 0 is more frequent 438 compared to the class 1 & the data is imbalanced.

Histogram:

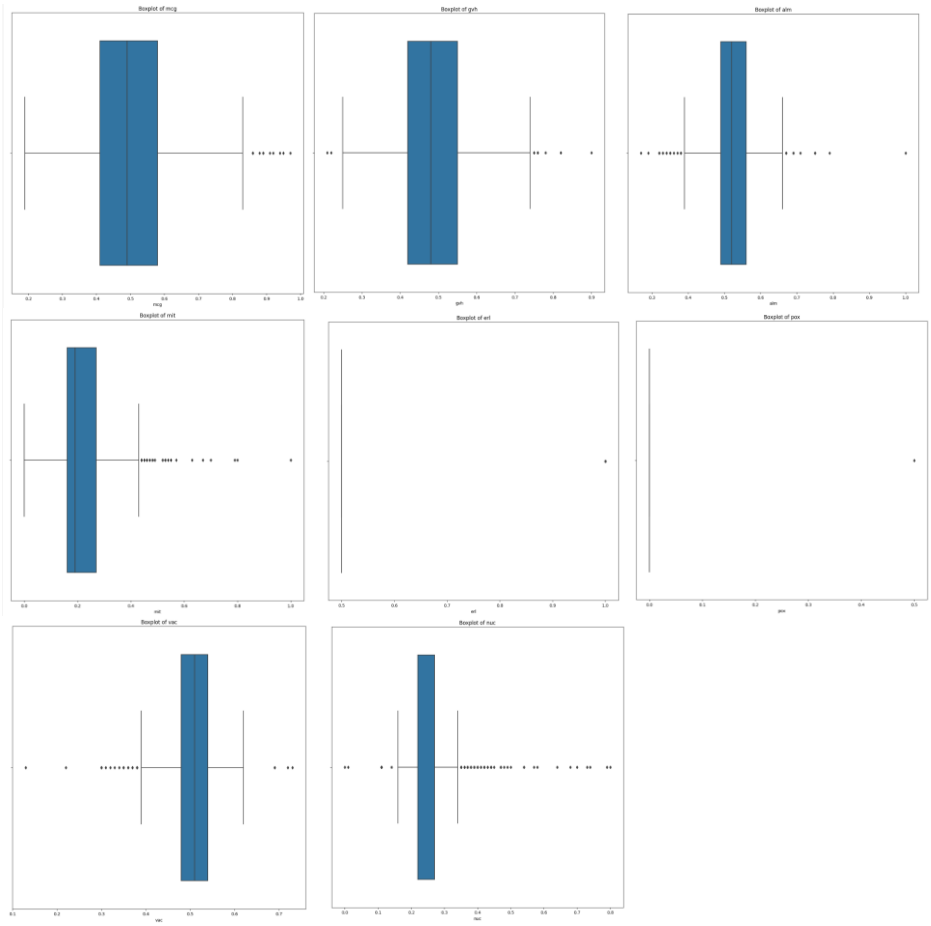


mcg: is symmetric distribution & slightly right skewed.
gvh: is normal distributed.
alm: is right skewed.
mit: is heavily right-skewed. values.
erl: all values around 0.5 and 1.0 and no values in between. pox: most of values are zero & its may not be important feature for the model
vac: is slightly left-skewed.
nuc: is right-skewed. Most of the values around 0.2.



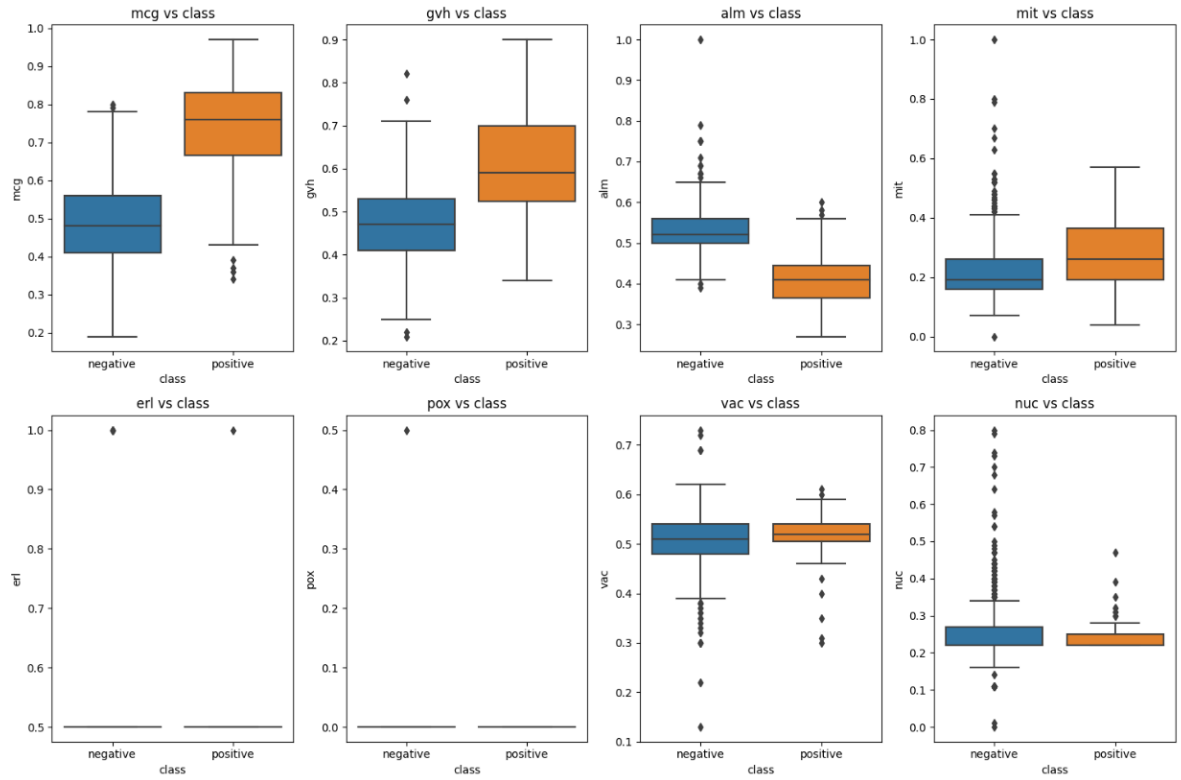
class: is a target categorical variable which is binary 0 and 1. Class 0 is more frequent 438 compared to the class 1 & the data is imbalanced.

Boxplot



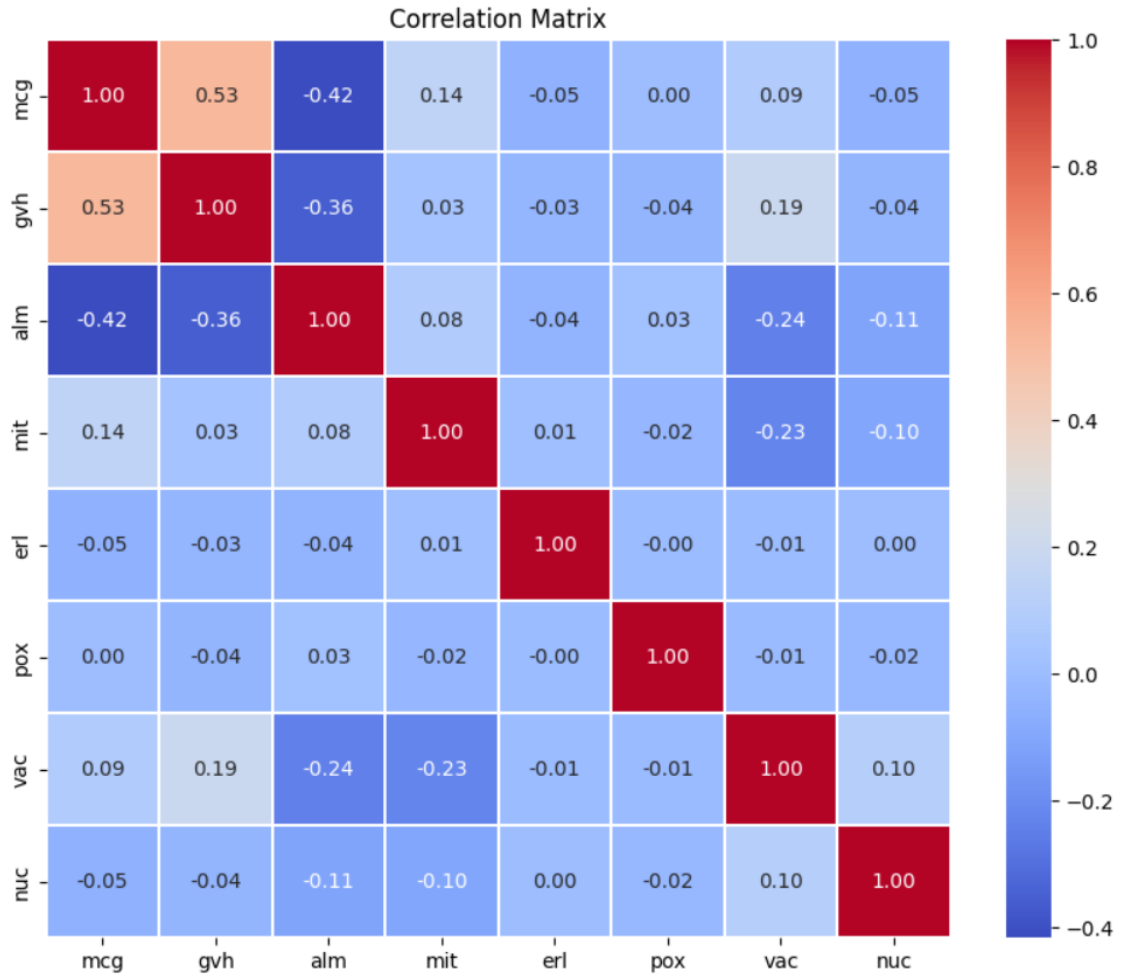
mcg: median is nearly at 0.5 with few outliers.
gvh: median slightly below 0.5 & has few outliers on both end
alm: median value close to 0.5 with outliers on both end.
mit: median slightly below 0.2 & has more outliers on right.
erl: its two groups of data one at 0.5 and 1 & should be a categorical variable.
pox: its two groups of data one at 0 and 1 & most of the values are at 0 and hence it may not be much useful for the model.
vac: median close to 0.5 & has outliers on both end and has more outliers on left.
nuc: most of value falls between 0 and 0.4 & has more outliers on right and few on left.

Boxplot With Class:



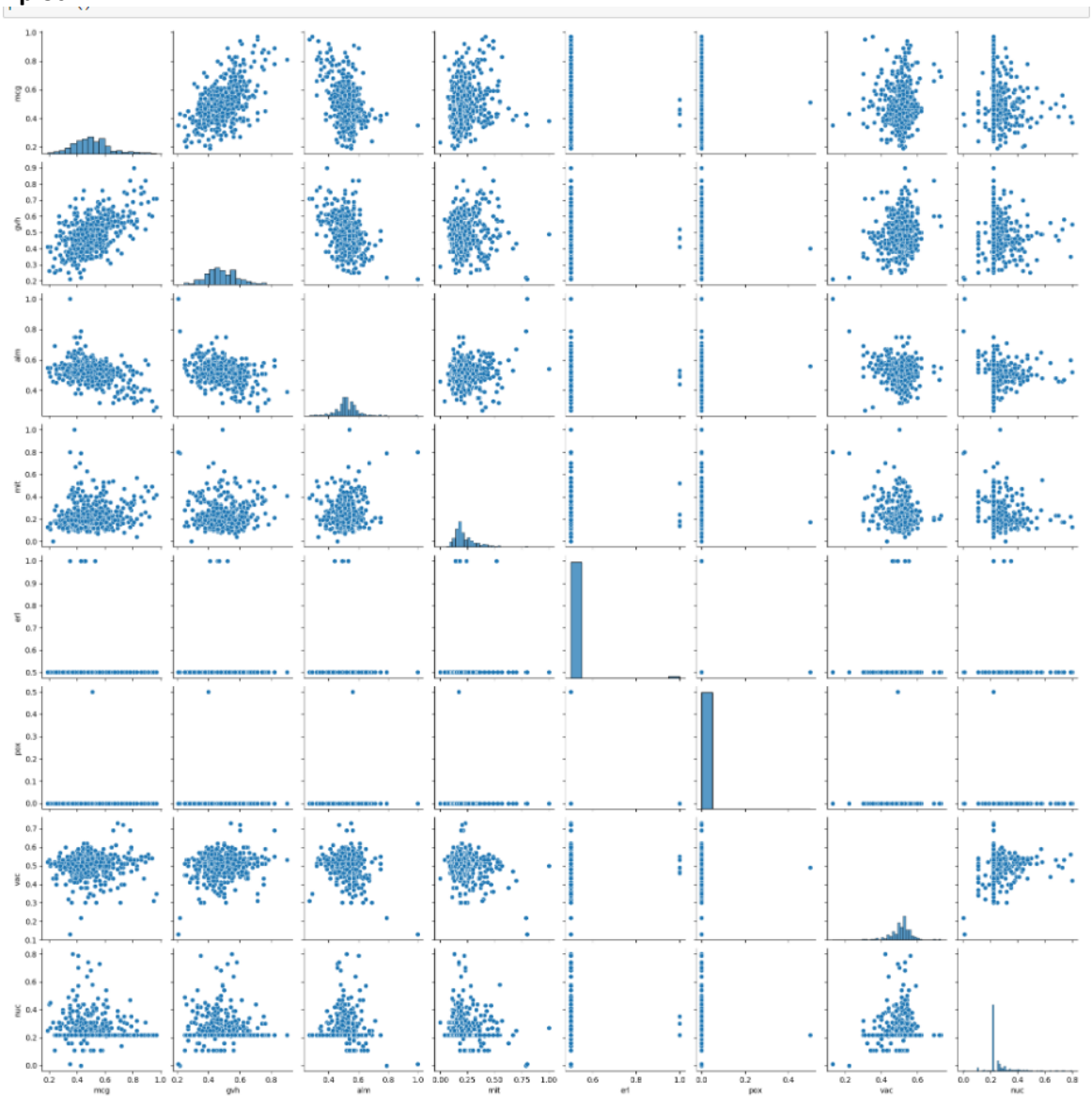
mcg vs class: median of negative class is around 0.5 and positive class is around 0.8 & has difference between positive & negative classes.
gvh vs class: positive class has slightly higher median and wider spread compared to negative class.
alm vs class: negative class has slightly lower median compared to positive class.
mit vs class: negative class has more outliers but positive class has no outliers.
erl vs class: most values are around 0.5 with fewer at 1 for positive class
pox vs class: is zero for both classes & indicates it may not useful for model.
vac vs class: has a similar median for both classes with slightly wider for the positive class.
nuc vs class: has slightly higher median and less outlier for positive class compared to negative class.

Correlation



erl do not have significant correlation with any features as the correlations are close to zero.
pox do not have significant correlation with any features as the correlations are close to zero.
Strong Positive correlation pair: ('mcg', 'gvh')
Strong Positive correlation value: 0.5293766268349265
Strong Negative correlation pair: ('mcg', 'alm')
Strong Negative correlation value: -0.4163643951285503

Pairplot:



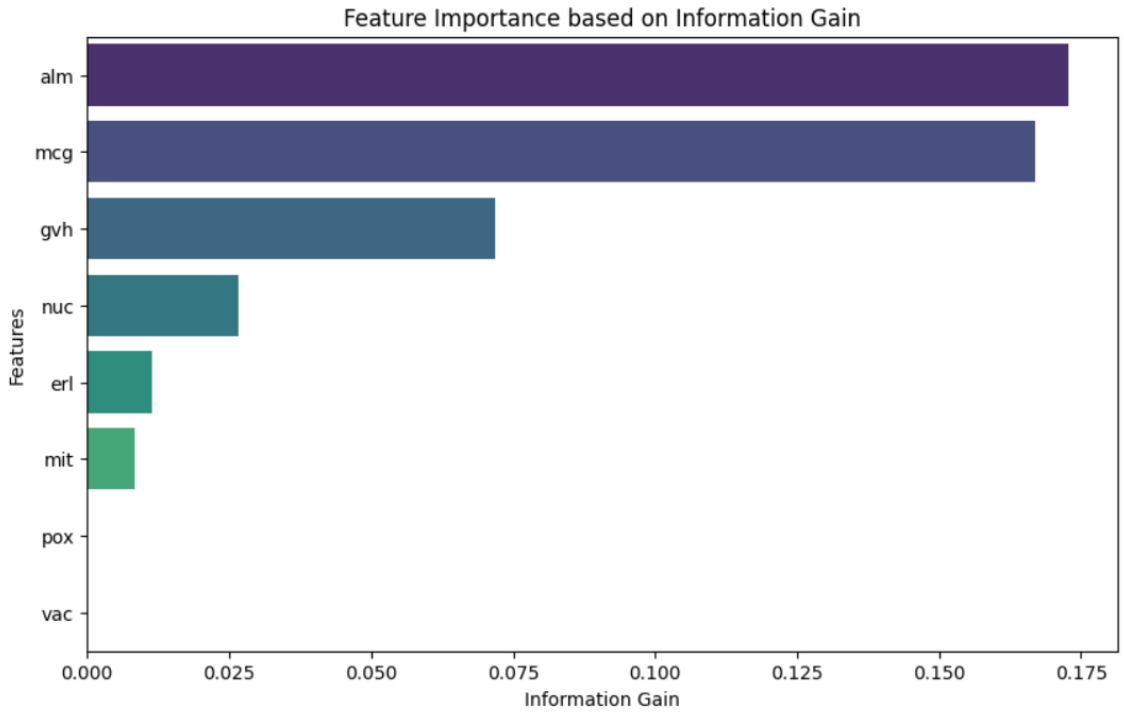
mcg vs. gvh: has a stong positive correlation between these two features
gvh vs. alm: also has a positive correlation.
alm vs. mcg: has a negative correlation.
mit has positive correlation with other features but relationships seem to be weak.
erl and pox has no correlations with other features.
nuc do not have strong correlation with any of the other features.

Feature Importance:

Feature importance is a practice of assigning scores to the independent features for a machine learning model based on their influence in predicting the target variable, scores would be indicated which features are more significant in model's predictions and which features could potentially be ignored without loosing too much predictive power.
Types of Feature Importance:
There are many types of feature importance scores, including statistical correlation scores as we seen above erl and pox do not have any correlation or relationship to any features and also to target.
Coefficients from linear models, decision tree importance, and permutation importance scores.

Each type provides different perspective on feature importance and scores would be different.
Role of Feature Importance:
Feature importance vital in predictive modeling. It offers insights into the data and the model & helps also help in dimensionality reduction and feature selection. These processes can improve the efficiency and effectiveness of a predictive model.

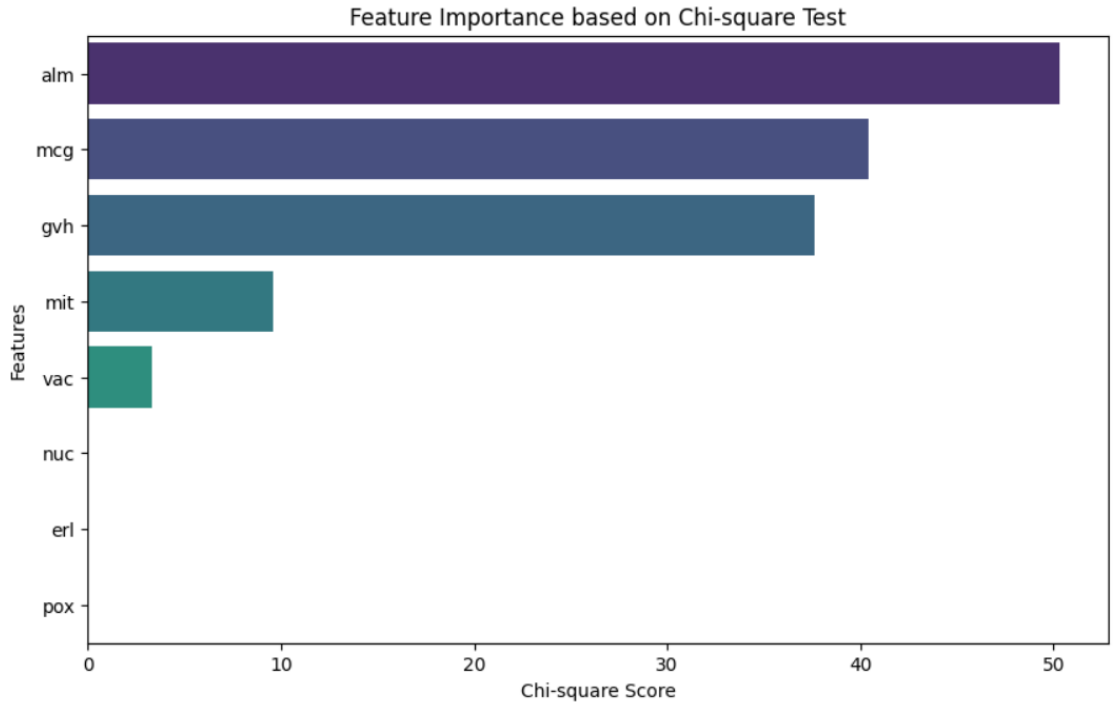
Information Gain (IG) is a measure helps to identify which features are the most informative for predicting the target variable & used with decision tree algorithms and is based on the concept of entropy from information theory



alm: has higest Information Gain score of 0.171 & its important feature for predicting presence or absence of protein.
mcg: has second highest Information Gain score of 0.166 & its important for the model.
gvh: With a score of 0.078 & provides some information for the prediction.
pox: With a score of 0.020 & influence small amount of information for the prediction.
erl: with Information Gain score of 0.019 & relatively small amount of information for the prediction.
mit: has very little information for the prediction as its score is 0.017.
vac, nuc: These features have Information Gain scores of 0 & do not provide any information about the presence or absence of protein.

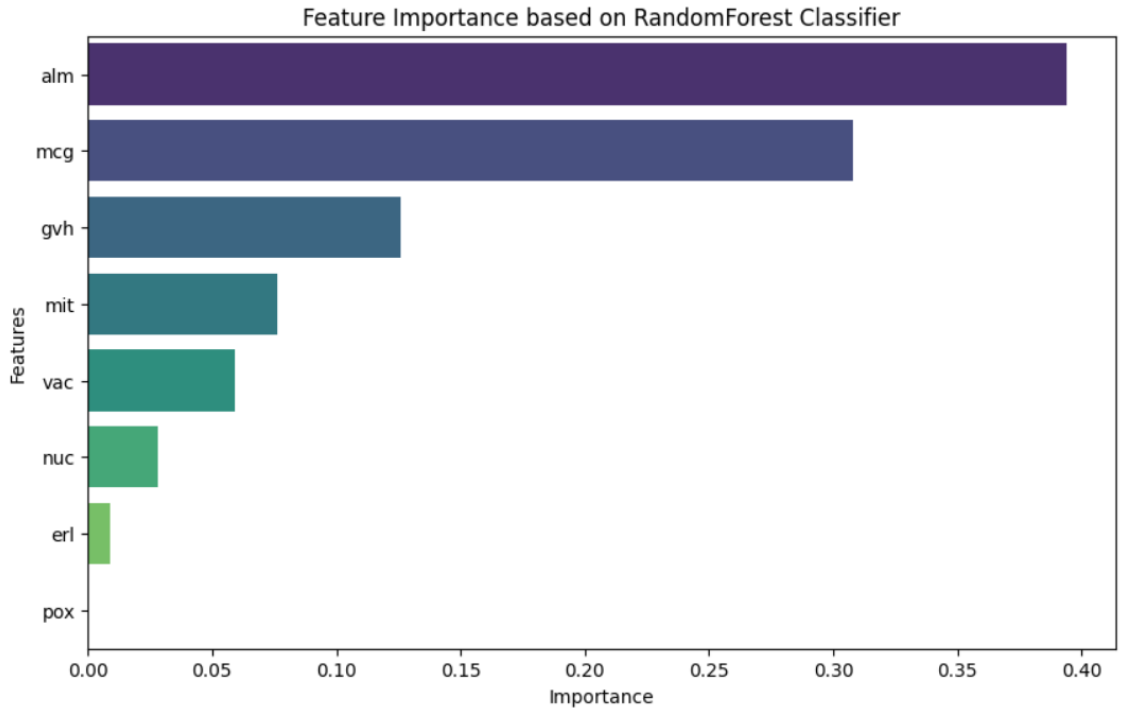
Chi-square test is statistical test to determine if there's a significant association between two categorical variables. In terms of feature selection, it can be used to test the relationship between each feature and the target variable. If the test shows that the feature and target are independent, the feature is not informative for predicting the target and could potentially be ignored.

Chi-square test requires that the input data be categorical. If the features or target are not categorical we would need to binned. Chi-square test cannot be used with features or targets that can take on a large number of categories, as the test's performance deteriorates in this case.



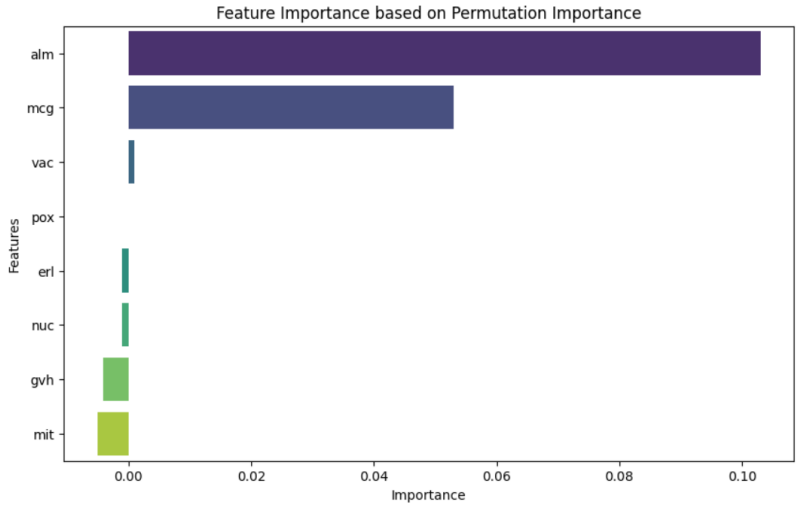
alm: Chi-square score of 50.34 and a p-value close to 0 & alm feature has high significance with target variable.
mcg: Second Chi-square score of 40.46 and a p-value close to 0 & its important for the prediction.
gvh: With a score of 37.65 and a p-value close to 0 & its important for the prediction.
mit: Chi-square score of 9.57 and a p-value of 0.002 & indicates statistically significant association with the target.
vac: Chi-square score of 3.30 and a p-value of 0.069 & as the p-value is larger than the commonly used significance level of 0.05 hence not consider as statistically significant.
nuc: has a very low Chi-square score 0.025 and a p-value of 0.873 & indicates no significance with the target.
erl, pox: Chi-square scores and p-values for these features are not defined as these features were constantafter binning, and the Chi-square test cannot be applied in this case.

Random Forest uses a measure based on the decrease in node impurity & is Gini impurity or entropy in case of classification, and variance in case of regression. Every time a feature is used to split data, the impurity of the child nodes is calculated. The difference between the impurity of the parent node and the weighted impurity of the child nodes gives us the impurity decrease. The feature importance for a feature is calculated as the average impurity decrease for that feature across all trees in the forest.



alm: is the most important as contributing 39.4% to the prediction of the target variable.
mcg: is the second most important, contributing 30.8% to the prediction.
gvh: is contributing 12.6% to the prediction.
mit: is contributing 7.6% to the prediction.
vac: is contributing 5.9% to the prediction.
nuc: is contributing 2.8% to the prediction.
erl: is contributing 0.9% to the prediction.
pox: does not contribute to the prediction.

Permutation Importance: is also a random forest model. This method involves randomly shuffling one feature in the validation dataset and calculating how much the model's performance decreases & is that an important feature, when shuffled, will drastically decrease the performance of the model, because the model relies on the feature for the prediction.



alm: is the most important featurecontributing approximately 10.3% to the prediction of the target variable.
mcg: is the second most important & contributing approximately 5.3% to the prediction.
vac: contributes only 0.1% to the prediction.
pox: do not contribute to the prediction.
erl: is slightly decreases the model performance when permuted.
nuc: is also slightly decreases the model performance when permuted.
gvh: is decreases the model performance by about 0.4% when permuted.
mit: is decreases the model performance by about 0.5% when permuted.

Similarity and Diffeences of Feature Importance

Feature importance analysis has been performed using four different methods

Information Gain
Chi-square Test
Random Forest Decrease in Node Impurity and
Permutation Importance

alm: has consistently high importance across all four methods, making it the most important feaure in predicting the presence or absence of protein.
mcg: is also consistently important across all methods & ranks second most important feature.
gvh: shows significant influence in the Chi-square Test and Random Forest methods but has less score in Information Gain and it even slightly decreases model performance in the Permutation Importance. The importance of this feature might be modetate.
mit: shows moderate importance in Random Forest method and has a significant importance with the target in the Chi-square Test. But contributes very less in Information Gain method and decreases the model performance in the Permutation Importance method.
vac: shows a small contribution in the Random Forest method and a very small influence in the Permutation Importance method. Also shows any significant importance with the target in Chi-square Test and doesn't contribute at all in the Information Gain method.
nuc: do not contribute in Information Gain method & Chi-square Test. It contributes a small amount in the Random Forest method but slightly decreases model performance in the

Permutation Importance method.
pox: do not contribute in the Information Gain, Random Forest, and Permutation Importance methods & in Chi-square Test score and p-value couldn't be calculated due to constant values after binning.
erl: contributes very little in the Random Forest method and slightly decreases model performance in the Permutation Importance method & do not contribute in the Information Gain method and its Chi-square Test score and p-value couldn't be calculated due to constant values after binning.

Conclusion:

alm and mcg are the most informative features for predicting the presence or absence of protein, pox do not contribute to the model in the Information Gain, Random Forest, and Permutation Importance methods. Its Chi-square Test score and p-value couldn't be calculated due to constant values after binning. This could be dropped.
erl contributed very little in the Random Forest method and slightly decreased model performance in the Permutation Importance method. It didn't contribute in the Information Gain method and its

Chi-square Test score and p-value couldn't be calculated due to constant values after binning. This column could be dropped too.

Ensemble Models:

Random Forest:
Train F1: 1.0 | Test F1: 0.88
Train ROC AUC: 1.0 | Test ROC AUC: 0.9898
Train Harmonic Mean: 1.0 | Test Harmonic Mean: 0.88
Train Geometric Mean: 1.0 | Test Geometric Mean: 0.8807

Confusion Matrix

Model: [[84 2]
[1 11]]

Classification report

Model:	precision	recall	f1-score	support
0	0.99	0.98	0.98	86
1	0.85	0.92	0.88	12
accuracy			0.97	98
macro avg	0.92	0.95	0.93	98
weighted avg	0.97	0.97	0.97	98

Overfitting Analysis: As the training scores are at 100% indicates the model might be overfitting to the training data & slight drop in test scores indicates model performs well with unseen data.

Performance: has the highest performance metrics among all models for test dataset & making it the more suitable model for this dataset.

Gradient Boosting:

Train F1: 1.0 | Test F1: 0.6957
Train ROC AUC: 1.0 | Test ROC AUC: 0.9608
Train Harmonic Mean: 1.0 | Test Harmonic Mean: 0.6957
Train Geometric Mean: 1.0 | Test Geometric Mean: 0.6963

Confusion Matrix

Model: [[83 3]
[3 9]]

Classification report

Model	precision	recall	f1-score	support
0	0.97	0.97	0.97	86
1	0.75	0.75	0.75	12

accuracy			0.94	98
macro avg	0.86	0.86	0.86	98
weighted avg	0.94	0.94	0.94	98

Overfitting Analysis: More significant drop in the test scores compared to Random Forest, especially in the F1 Score and Harmonic Mean, which indicates overfitting.

Performance: The model's ROC AUC is good but the drop in F1 score means that it might not classify the positive samples as accurately as Random Forest.

AdaBoost:

Train F1: 1.0 | Test F1: 0.72
Train ROC AUC: 1.0 | Test ROC AUC: 0.8779
Train Harmonic Mean: 1.0 | Test Harmonic Mean: 0.72
Train Geometric Mean: 1.0 | Test Geometric Mean: 0.7206

Confusion Matrix

Model: [[82 4]
[3 9]]

Classification report

Model:

	precision	recall	f1-score	support
0	0.96	0.95	0.96	86
1	0.69	0.75	0.72	12
accuracy			0.93	98
macro avg	0.83	0.85	0.84	98
weighted avg	0.93	0.93	0.93	98

Overfitting Analysis: The disparity between training and testing scores indicates overfitting.

Performance: recall on the test set is better than Gradient Boosting but trade-off is a reduced precision hence it misclassifies more negative samples as positive compared to Gradient Boosting.

Voting Classifier:

Train F1: 1.0 | Test F1: 0.75
Train ROC AUC: 1.0 | Test ROC AUC: 0.9787
Train Harmonic Mean: 1.0 | Test Harmonic Mean: 0.75
Train Geometric Mean: 1.0 | Test Geometric Mean: 0.75

Confusion Matrix

Model: [[83 3]
[3 9]]

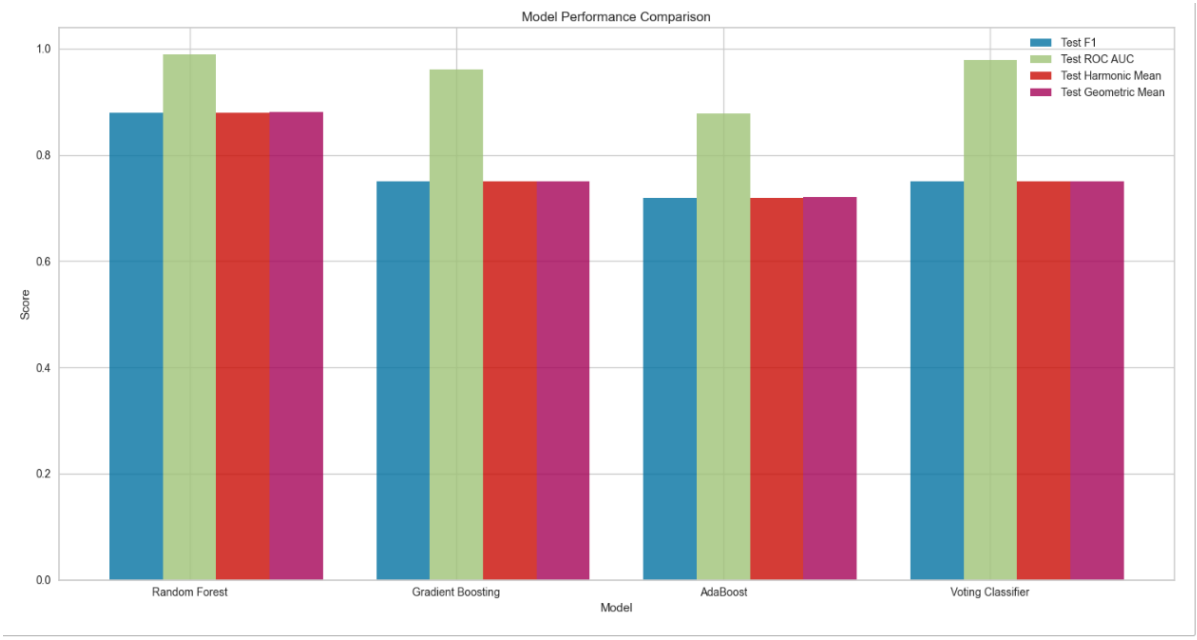
Classification report

Model	precision	recall	f1-score	support
0	0.97	0.97	0.97	86
1	0.75	0.75	0.75	12
accuracy			0.94	98
macro avg	0.86	0.86	0.86	98
weighted avg	0.94	0.94	0.94	98

Overfitting Analysis: The Voting Classifier shows some overfitting, but it generalizes better to the test data than Gradient Boosting and AdaBoost.

Performance: The Voting Classifier, by design, aims to get the best of its constituent models. Its balanced performance is commendable, but it still doesn't outperform Random Forest.

Base Model comparison Chart:



Next Steps

Based on all metrics, Random Forest is performing good but may be overfitting fine tuning the models with hyper parameter as next steps.

Hyperparameters:

Trained the models with below hyper parameters

Random Forest:
class_weight: Set to 'balanced' or 'balanced_subsample'. This adjusts weights inversely proportional to class frequencies.
n_estimators: Number of trees in the forest.
max_features: The number of features to consider when looking for the best split.
max_depth: The maximum depth of the tree.

Best Hyperparameter for random forest are: {'n_estimators': 200, 'max_features': 'auto', 'max_depth': 40, 'class_weight': 'balanced_subsample'}

Gradient Boosting:

n_estimators: Number of boosting stages to be run.
learning_rate: Shrinks the contribution of each tree.
subsample: The fraction of samples used for fitting the individual base learners.

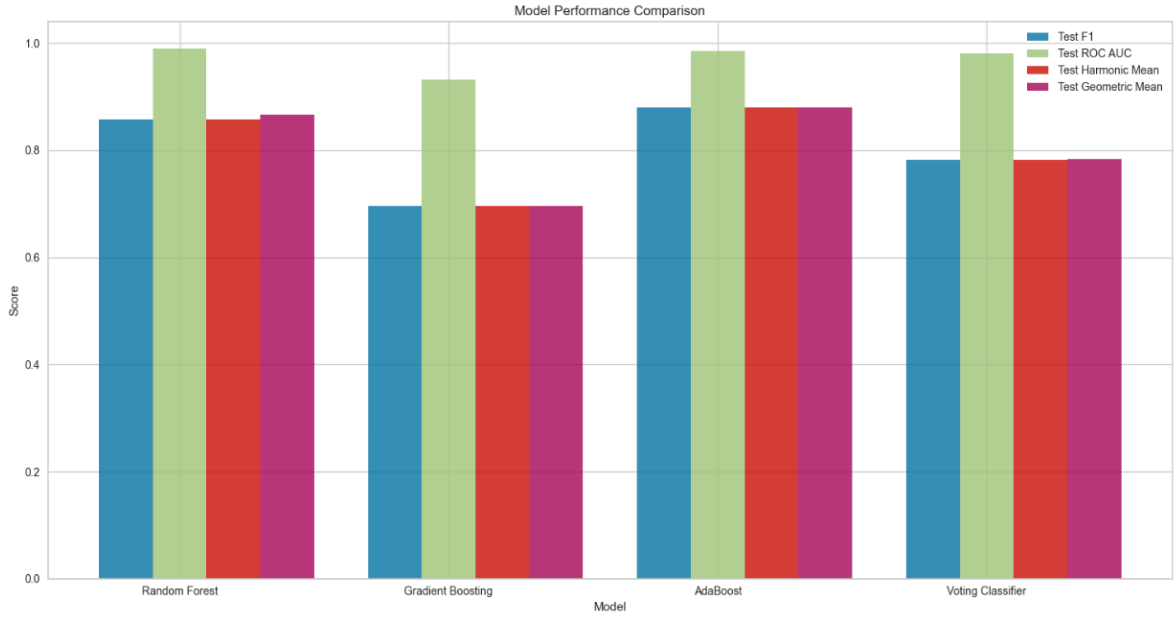
Best Hyperparameter for gradient boosting are: {'subsample': 1.0, 'n_estimators': 50, 'learning_rate': 0.1}

AdaBoost:
n_estimators: Maximum number of estimators (base classifiers) at which boosting is terminated.
learning_rate: Shrinks the contribution of each classifier.

Best Hyperparameter for Ada Boosting are: {'n_estimators': 50, 'learning_rate': 0.1}

Voting Classifier:
Using the best hyperparameters from the individual models.

Hyperparameter Model Comparison



Random Forest:

Train F1: 1.0 | Test F1: 0.8571
Train ROC AUC: 1.0 | Test ROC AUC: 0.9903
Train Harmonic Mean: 1.0 | Test Harmonic Mean: 0.8571
Train Geometric Mean: 1.0 | Test Geometric Mean: 0.8660

Overfitting Analysis: Training scores are perfect & with slight decline in the test scores model is still performing well to unseen data. Minor difference between the training and test scores suggests slight overfitting.

Performance: Even with hyperparameter tuning, the Random Forest model remains the best-performing model among all in terms of the test dataset, particularly in the F1 Score and ROC AUC.

Gradient Boosting:

Train F1: 1.0 | Test F1: 0.6957
Train ROC AUC: 1.0 | Test ROC AUC: 0.9302
Train Harmonic Mean: 1.0 | Test Harmonic Mean: 0.6957
Train Geometric Mean: 1.0 | Test Geometric Mean: 0.6963

Overfitting Analysis: There's a significant drop in the test scores compared to the training scores, indicating overfitting. The gap has remained similar even after tuning.

Performance: The ROC AUC for Gradient Boosting is good but decline in F1 score on the test dataset indicates it is not reliable in classifying positive samples compared to Random Forest.

AdaBoost:

Train F1: 0.8286 | Test F1: 0.8800
Train ROC AUC: 0.9924 | Test ROC AUC: 0.9860
Train Harmonic Mean: 0.8286 | Test Harmonic Mean: 0.8800
Train Geometric Mean: 0.8340 | Test Geometric Mean: 0.8807

Overfitting Analysis: The AdaBoost model has shown improvement with hyperparameter tuning. The gap between training and test scores is relatively small, suggesting the model is generalizing well without much overfitting.

Performance: recall on the test set is impressive, and its overall performance makes it a strong contender, nearly matching the performance of the Random Forest.

Voting Classifier:

Train F1: 1.0 | Test F1: 0.7826
Train ROC AUC: 1.0 | Test ROC AUC: 0.9806
Train Harmonic Mean: 1.0 | Test Harmonic Mean: 0.7826
Train Geometric Mean: 1.0 | Test Geometric Mean: 0.7833

Overfitting Analysis: small overfitting observed as training scores are perfect while the test scores are lower but model performance on the test set is more balanced than Gradient Boosting.

Performance: By design, Voting Classifier aims to achieve the best from its constituent models & performance is good but still do not surpass Random Forest.

Conclusion:
AdaBoost with optimized hyperparameters is final model of choice for this dataset considering the challenge of class imbalance. Its high recall rate ensures that it captures almost all positive instances.

Ensemble models over other ML models:

1. Model Performance Improvement:

Ensemble models combine the strengths of multiple models to achieve better performance than any single model could achieve alone. By aggregating the results of multiple models, ensembles often reduce both bias and variance in context of this dataset & the imbalanced nature of the dataset, individual models might exhibit bias towards the majority class. Ensemble methods like Random Forest or AdaBoost inherently handle imbalances and often yield better performance metrics, as observed in the results.

2. Reduction of Overfitting:

Ensemble models bagging methods like Random Forest, reduce overfitting by introducing randomness like bootstrapping samples or selecting random subsets of features. In context of dataset the ensemble methods used here particularly AdaBoost exhibited less overfitting than some other models. This is perform better with the unseen data.

3. Model Stability:

Individual models might be sensitive to minor changes in the data & leading to large variances in predictions. Ensemble models averages out the predictions of multiple models & provide more stable and consistent predictions. In context of this biological datasets like protein presence or absence can have inherent variability & using ensemble models can help in achieving more consistent predictions across different samples or experiments.

4. Handling High Dimensionality:

Ensemble models particularly tree-based ensembles like Random Forest can handle datasets with a large number of features effectively & in context of dataset has multiple features, some of which might be redundant or less informative. Ensemble models can inherently rank feature importance and focus on the most informative ones, improving predictive performance.

5. Versatility:

Ensemble methods can be used for classification, regression, and even unsupervised learning tasks. They can be constructed using various base models making them adaptable to different problems & in context of this dataset I utilized ensemble methods for a binary classification problem. However, if the problem evolves multi-class protein classification ensemble methods like Random Forest or Gradient Boosting can be easily adapted.

Conclusion:

Individual machine learning models can be powerful and efficient, ensemble models bring together the collective strength of multiple models, offering enhanced performance, stability, and versatility & in the context of predicting the presence or absence of protein the ensemble models especially AdaBoost, showcased strong predictive capabilities while effectively handling the challenges posed by the imbalanced dataset.

Similarities or differences between these models:

Similarities:

Type of Models: All four models Random Forest, Gradient Boosting, AdaBoost, and Voting Classifier are ensemble techniques. This means they consolidate predictions from multiple base models to improve generalization and robustness.

Handling of Imbalance: Random Forest and AdaBoost offer intrinsic mechanisms, such as the class_weight parameter, for handling class imbalance. This is particularly useful in scenarios where the target classes are imbalanced, enhancing the model's predictive power.

Capability for Multi-purpose Modeling: These models are versatile and can be applied to a wide range of tasks, including both classification and regression. This makes them universally applicable across various domains and use-cases.

Base Learners: Except for Gradient Boosting, which can use different types of base learners, the other models predominantly use decision trees. Decision trees are known for their simplicity and ability to handle both categorical and numerical variables.

Flexibility in Hyperparameter Tuning: All four algorithms provide numerous hyperparameters that can be fine-tuned to optimize performance. This level of customization allows for targeted model optimization based on specific evaluation metrics.

Robustness to Overfitting: Ensemble methods, by their very nature, are generally robust to overfitting. This is especially true when they are configured with a moderate number of weak learners.

Interpretability and Feature Importance: While not as interpretable as linear models, tools and techniques exist for these ensemble models to interpret feature importance. This is crucial for understanding which variables are driving the predictions.

Scalability: These models are generally well-suited for large datasets and high-dimensional spaces, thanks to their ensemble nature and the use of decision trees as base learners, which are computationally efficient.

Tolerance to Missing Data: Ensemble models, especially those based on decision trees like Random Forest, are relatively robust to missing or erroneous data. This is because the decision trees within the ensemble can handle missing values naturally during the partitioning process.

Cross-Validation Compatibility: All these models can be readily used with cross-validation techniques for hyperparameter tuning and model evaluation, enhancing their robustness and generalizability.

Differences:

Model Construction:

Random Forest: It is a bagging model that builds multiple decision trees and aggregates their results. Each tree is trained on a random subset of the data and uses a random subset of features.

Gradient Boosting: It is a boosting model where trees are built sequentially. Each tree tries to correct the errors of its predecessor.

AdaBoost: It is also a boosting model. It adjusts the weights of misclassified instances after each tree is built, making the next tree focus more on them.

Voting Classifier: It combines the predictions of multiple models in this case, Random Forest, Gradient Boosting, and AdaBoost and makes a final prediction based on majority voting or average probabilities.

Model Performance:

Random Forest: Has the highest Test ROC AUC (0.9903) but a slightly lower Test F1 score (0.8571) compared to AdaBoost.

Gradient Boosting: Shows signs of overfitting with perfect training scores but reduced test scores. Test F1 is 0.6957, and Test ROC AUC is 0.9322.

AdaBoost: Exhibits a well-balanced performance with a Test F1 score of 0.8800 and a Test ROC AUC of 0.9860. It achieves a higher Test Recall of 0.9167.

Voting Classifier: Achieves a balance between its constituent models with a Test F1 score of 0.7826 and a Test ROC AUC of 0.9806.

Hyperparameter Sensitivity:

Random Forest: Less sensitive due to the random nature of tree building.

Gradient Boosting & AdaBoost: More sensitive as the performance can vary significantly with changes in learning rate or number of trees.

Voting Classifier: Depends on the hyperparameters of its constituent models.

Challenges:

Computational Complexity: Random Forest generally requires more computational power and memory as the number of trees increases, while Gradient Boosting is computationally expensive because trees are built sequentially.

Risk of Overfitting: Gradient Boosting has a higher risk of overfitting, especially if the data is noisy, compared to Random Forest and Voting Classifier.

Conclusion:

AdaBoost stands out due to its balanced performance on the test dataset. It combines the strengths of boosting to focus on misclassified instances and achieves a compelling balance between precision and recall. However, it's crucial to consider the computational costs and the risk of overfitting when choosing a model. Gradient Boosting, despite its high training scores, fails to generalize well on the test data. Random Forest provides robustness but at the cost of computational efficiency. The Voting Classifier offers a balanced performance and serves as a reliable choice for this datasets.

Preferable scenario for using any specific model:

Random Forest:

Scenarios:
Suitable for datasets with a mix of numerical features and imbalances.
For capturing complex feature interactions without manual engineering.
Valuable when feature interpretability is essential.

Advantages:
Robust to overfitting due to bagging and randomness.
Good performance without extensive tuning.

Gradient Boosting:

Scenarios:
Useful when minute patterns in the data can enhance prediction accuracy.
When aiming for the highest performance, even if computational resources are intensive.

Advantages:
Often the most accurate among ensemble methods.
Provides insight with feature importance scores.

AdaBoost:

Scenarios:
Tailored for binary classification like protein presence prediction.
When a balance between interpretability and performance is needed.

Advantages:
Focuses on hard-to-classify instances.
Faster training compared to Gradient Boosting.

Voting Classifier:

Scenarios:
Combines strengths of multiple models for a robust prediction.
Aims to balance trade-offs of individual models.

Advantages:
Reduces overfitting by aggregating predictions.
Offers model combination flexibility.

General Points:

Data Size: Boosting is preferable for not-so-large datasets, but requires careful tuning.

Training Time: Random Forest's parallel training is beneficial for large datasets; Gradient Boosting is more demanding.

Noise and Outliers: Random Forest is robust in scenarios with significant outliers.

Interpretability: In biological contexts, Random Forest and Gradient Boosting's feature importance scores are invaluable.

In Essence for predicting protein presence, ensemble model choice should weigh data characteristics, biological intricacies, and interpretability alongside accuracy.

Possible to build ensemble model using ML classifiers:

Is it possible to build ensemble models using ML classifiers other than decision trees? If yes, then explain with an example.

yes, ensemble models can be constructed using a variety of base classifiers, not limited to decision trees.

Illustration:
In the above code, ensemble model that combines the predictions from a Logistic Regression, KNN and SVM classifiers. This ensemble is built using the VotingClassifier from scikit-learn.

Logistic Regression: A statistical method that predicts the probability of a binary outcome based on one or more predictor variables.

KNN: A non-parametric method that classifies a data point based on the majority class of its 'k' nearest neighbors in the training set. SVM: A method that aims to find the optimal hyperplane that best divides the dataset into classes.

Using VotingClassifier, aggregated the predictions of these different models by using soft voting, which predicts the class label based on the argmax of the sums of the predicted probabilities.

This ensemble method allows us to leverage the distinct strengths of each algorithm. Logistic Regression is powerful for its ability to predict probabilities based on linear relationships in the data. KNN captures local patterns, and SVM is effective in high-dimensional spaces or when the data is not linearly separable.

I have demonstrated that ensemble models can seamlessly integrate classifiers other than decision trees to have enhanced performance.

Ensemble Results:
Training Accuracy: 95.9%

Test Accuracy: 94.8%

As we could notice individual base models individually has overfitting and results were biased as we performed in Credit task 2. However, with voting classifier the model seems to be more balanced.

References:

1. Great Learning Olympus. (2023, July). Site for masters in data science Week 2 and Week 3 content. Read, understood & implemented the concepts from (https://olympus.mygreatlearning.com/courses/97556?module_id=628759).

2. Great Learning Olympus. (2023). Site for PGP for Artificial intelligence and machine learning study materials. Read, understood & implemented the concepts from (https://olympus.mygreatlearning.com/courses/74001).
3. Great Learning Olympus. (2023). Site for PGP for Artificial intelligence and machine learning previous project work. Read, understood & implemented the concepts from (https://olympus.mygreatlearning.com/courses/74001/assignments/337472?module_item_id=2852405), (https://olympus.mygreatlearning.com/courses/73999/assignments/315149?module_item_id=2606609), (https://olympus.mygreatlearning.com/courses/74000/assignments/332211?module_item_id=2799576).
4. AlindGupta. (No Date). Read, understood & implemented the concepts from (https://www.geeksforgeeks.org/ml-spectral-clustering/).
5. João Pedro. (May 19, 2022). Read, understood & implemented the concepts from (https://towardsdatascience.com/how-to-ensemble-clustering-algorithms-bf78d7602265).
6. Scikit-learn. (No Date). Read, understood & implemented the concepts from (https://scikit-learn.org/stable/modules/clustering.html).
7. Saul Dobilas. (October 25, 2021). UMAP Dimensionality Reduction — An Incredibly Robust Machine Learning Algorithm. Read, understood & implemented the concepts from (https://towardsdatascience.com/umap-dimensionality-reduction-an-incredibly-robust-machine-learning-algorithm-b5acb01de568).
8. Analytics Vidhya. (2019, May 27). Hierarchical Clustering published by Pulkit Sharma. Read, understood & implemented the concepts from (https://www.analyticsvidhya.com/blog/2019/05/beginners-guide-hierarchical-clustering/).
9. Analytics Vidhya. (2020, October 15). DBScan Clustering by Kavya Gajjar. Read, understood & implemented the concepts from (https://medium.com/analytics-vidhya/cluster-analysis-with-dbscan-density-based-spatial-clustering-of-applications-with-noise-6ade1ec23555).
10. Scikit-Learn. (No Date). DBScan. Read, understood & implemented the concepts from (https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html).
11. Scikit-Learn. (No Date). Hierarchical clustering. Read, understood & implemented the concepts from (https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html).
12. Jason Brownlee. (April 6, 2020). 10 Clustering Algorithms With Python. Read, understood & implemented the concepts from (https://machinelearningmastery.com/clustering-algorithms-with-python/).
13. Anshul Saini. (September 15, 2021). Master the AdaBoost Algorithm: Guide to Implementing & Understanding AdaBoost. Read, understood & implemented the concepts from (https://www.analyticsvidhya.com/blog/2021/09/adaboost-algorithm-a-complete-guide-for-beginners/).
14. Scikit-Learn. (No Date). Hyper parameter tuning. Read, understood & implemented the concepts from (https://scikit-learn.org/stable/modules/grid_search.html).
15. Analytics Vidhya. (2021, June 17). Random Forest: Sruthi E R. Read, understood & implemented the concepts from (https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/).
16. Analytics Vidhya. (2020, December 17). Hierarchical Clustering by Heena Girdher. Read, understood & implemented the concepts from (https://medium.com/analytics-vidhya/hierarchical-clustering-d2d92835280c).
17. Prabowo Yoga Wicaksana. (March 9, 2022). Hyperparameter Optimization on Random Forest Classifier. Read, understood & implemented the concepts from (https://medium.com/@prabowoyogawicaksana/hyperparameter-optimization-random-forest-classifier-550fd5ed8e14).

18. Mohtadi Ben Fraj. (December 24, 2017). Parameter tuning for Gradient Boosting. Read, understood & implemented the concepts from (https://medium.com/all-things-ai/in-depth-parameter-tuning-for-gradient-boosting-3363992e9bae).
19. Brownlee, J. (2020). Tour of Evaluation Metrics for Imbalanced Classification. Read, understood & implemented the concepts from (https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification/).
20. Brownlee, J. (No Date). Metrics for Imbalanced Classification Books. Read, understood & implemented the concepts from (https://machinelearningmastery.com/imbalanced-classification-with-python/).
21. Jain, A. (2018). Hyperparameters and Parameters in Machine Learning: What's the difference? Read, understood & implemented the concepts from (https://www.analyticsvidhya.com/blog/2018/08/difference-between-parameters-and-hyperparameters/).
22. Zhou, Z. (2018). Understanding Hyperparameters and its Optimisation techniques. Read, understood & implemented the concepts from (https://towardsdatascience.com/understanding-hyperparameters-and-its-optimisation-techniques-f0debba07568).
23. Goyal, P. (2018). Feature Importance and Why It's Important. Read, understood & implemented the concepts from (https://towardsdatascience.com/feature-importance-and-why-its-important-c46d326e81d2).
24. Sankar, A. (2019). The 5 Classification Evaluation Metrics Every Data Scientist Must Know. Read, understood & implemented the concepts from (https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/).
25. Patel, K. (2018). Why, How and When to Scale your Features. Read, understood & implemented the concepts from (https://medium.com/greyatom/why-how-and-when-to-scale-your-features-4b30ab09db5e).
26. MACHINE LEARNING EXPLAINED. (2020, December 18). Density-Based Spatial Clustering of Applications with Noise (DBSCAN). Read, understood, and updated from https://ml-explained.com/blog/dbscan-explained.
27. SKLearn. (No date). Optics clustering. Read, understood, and implemented from https://scikit-learn.org/stable/modules/generated/sklearn.cluster.OPTICS.html.
28. Rupak (Bob) Roy - II. (2021, June 25). OPTICS CLUSTERING (Intro). Read, understood, and implemented from https://bobrupakroy.medium.com/optics-clustering-intro-76dcdaf94bde.
29. Varsha's Engineering Stuff. (2020, December 29). Optics Clustering Algorithm, Data Mining. Watched video, understood, and implemented from https://www.youtube.com/watch?v=UgCDPRZFj10.
30. Brownlee, J. (2020) "Ensemble Learning Methods for Deep Learning Neural Networks." Read, understood, and implemented from Machine Learning Mastery from https://machinelearningmastery.com/ensemble-learning-methods-for-deep-learning-neural-networks/
31. Zhang, T. (2019) "Ensemble Learning: A Beginner's Guide." Read, understood, and implemented from Towards Data Science from Link: https://towardsdatascience.com/simple-guide-for-ensemble-learning-methods-d87cc68705a2